

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

**«Розробка онлайн-системи для продажу будівельних
матеріалів»**

Виконав студент групи ІПЗ-19-1
ХОМЧУК Антон Олександрович

Керівник роботи:
МАРЧУК Галина Вікторівна

Рецензент:
КРАВЧЕНКО Світлана Миколаївна

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

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«02» серпня 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: ХОМЧУК Антон Олександрович

Керівник роботи: **МАРЧУК Галина Вікторівна**

Тема роботи: **«Розробка онлайн-системи для продажу будівельних матеріалів»**,

затверджена Наказом закладу вищої освіти від **«02» серпня 2023 р., № 326/с**

Вихідні дані для роботи: **використання веб-орієнтованих технологій для розробки онлайн-системи для продажу будматеріалів.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Марчук Г.В.	03.08.2023р.	14.08.2023р.
2	Марчук Г.В.	15.08.2023р.	24.08.2023р.
3	Марчук Г.В.	25.08.2023р.	27.08.2023р.

РЕФЕРАТ

Кваліфікаційна робота бакалавра складається з програмного продукту на тему «Розробка онлайн-системи для продажу будівельних матеріалів» та пояснювальної записки. Пояснювальна записка містить 65 сторінок, 33 ілюстрації та 6 таблиць.

Метою роботи є створення онлайн-системи для продажу будівельних матеріалів з використанням наступного стеку технологій, зокрема HTML, CSS, JavaScript та PHP.

В роботі визначені головні завдання для розробки системи, проаналізовано методи вирішення поставленої задачі, проаналізовано методи обробки та аналізу даних. Наведено алгоритм роботи програмного комплексу, а також архітектуру та взаємодію його основних компонентів.

КЛЮЧОВІ СЛОВА: ОНЛАЙН-СИСТЕМА, PHP, MYSQL, JAVASCRIPT, СЕРВЕР, БД, LARAVEL, XAMPP.

					ІПЗ.КР.Б – 121 – 23 – ПЗ							
Зм.	Арк.	№ докум.	Підпис	Дата								
Розроб.		А.О. Хомчук			Розробка онлайн-системи для продажу будівельних матеріалів Пояснювальна записка			Літ.		Арк.	Аркушів	
Керівник		Г.В. Марчук								3	71	
н. контр.		О.П. Тичина						Житомирська політехніка, група ІПЗ-19-1				
Зав. каф.		А.В. Морозов										

ABSTRACT

Bachelor's graduate work consists of a software product on the topic "Development of an online system for the sale of building materials" and documentation. Documentation for bachelor's graduate work contain 65 pages, 33 illustrations, 6 tables.

The purpose of graduation work to create an online system for the sale of building materials using the following technology stack, in particular HTML, CSS, JavaScript and PHP. These skills will be applied in the process of developing a software product.

Graduation work contains main tasks for system development, analyzed the methods for solving the task, i.e., the development of an online system, and analyzed the methods of data processing and analysis. The algorithm of the software package, as well as the architecture and interaction of its main components are given.

KEYWORDS: ONLINE SYSTEM, PHP, MYSQL, JAVASCRIPT, SERVER, DATABASE, LARAVEL, XAMPP.

					ПЗ.КР.Б – 121 – 23 – ПЗ	
						4

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ІНФОРМАЦІЙНОГО ДОДАТКА.....	9
1.1 Постановка задачі.....	9
1.2 Аналіз аналогів програмного продукту	10
1.3 Обґрунтування вибору архітектури інформаційного додатка	13
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	14
Висновки до першого розділу	16
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ІНФОРМАЦІЙНОГО ДОДАТКА	17
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	17
2.2 Розробка бази даних системи	22
2.3 Проектування та реалізація алгоритмів роботи системи	25
2.4 Реалізація функціоналу онлайн-системи.....	26
Висновки до другого розділу	34
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКА	35
3.1 Порядок встановлення та налаштування системи.....	35
3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з онлайн-системою для продажу будівельних матеріалів	36
3.3 Тестування роботи програмного комплексу	44
Висновки до третього розділу	45
ВИСНОВКИ.....	47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	48
ДОДАТКИ.....	50
ДОДАТОК А	51

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних.

CRUD – Create, Read, Update, Delete (з англ. створити, прочитати, оновити, видалити).

MVC – Model–View–Controller (з англ. Модель–Представлення–Контролер).

ПЗ – програмне забезпечення.

ORM – об'єктно-реляційна проекція.

API – прикладний програмний інтерфейс.

ВСТУП

Актуальність теми. Тема "Онлайн-система для продажу будівельних матеріалів" є актуальною, оскільки такі онлайн-системи стали невід'ємною частиною сучасного електронного бізнесу, а будівельні матеріали є одним з найпопулярніших товарів, які продаються в Інтернеті.

За даними статистики, в Україні будівельний ринок зростає, а споживачі все частіше замовляють будівельні матеріали через Інтернет. Також варто зазначити, що пандемія COVID-19 змусила багатьох споживачів шукати можливості для покупки товарів онлайн, що підвищило популярність інтернет-магазинів в цілому.

Розробка онлайн-системи для продажу будівельних матеріалів дозволить підприємцям залучати нових клієнтів, збільшувати продажі та оптимізувати бізнес-процеси. Також це дозволить споживачам швидко та зручно знайти необхідні будівельні матеріали, порівняти ціни та характеристики, зробити замовлення та отримати їх доставку.

Метою кваліфікаційної роботи є розробка онлайн-системи для продажу будівельних матеріалів.

Встановлена мета обумовлює наступні завдання:

- проведення аналізу вимог;
- аналіз аналогічних програмних продуктів;
- вибір архітектури компонентів системи;
- обґрунтування та вибір засобів реалізації;
- визначення варіантів використання;
- розробка вебресурсу та реалізація всіх визначених вимог до програмного забезпечення;
- проектування алгоритмів системи;
- реалізація функціоналу онлайн-системи;
- опис структури інтерфейсу системи;
- тестування онлайн-системи.

Об’єктом дослідження є технології створення онлайн-систем для продажу.

Предметом дослідження є використання фреймворку Laravel та його використання у створенні у розробці онлайн-системи для продажу будівельних матеріалів.

Методи дослідження: в роботі було використано методи об’єктно-орієнтованого проектування, програмування та дизайну.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ІНФОРМАЦІЙНОГО ДОДАТКА

1.1 Постановка задачі

Метою розробки є створення онлайн-системи, який буде призначений для продажу будівельних матеріалів. За допомогою системи є можливість перегляду товару, додавання в кошик, вибору адресу доставки та способу оплати замовлення. Сервіс дає можливість відвідування так званих «гостей», тобто незареєстрованих користувачів, створення облікових записів для користувачів та адміністраторів, які використовують частковий або повний функціонал системи відповідно. Основна мета системи — продаж будівельних матеріалів. Для реалізації поставленого завдання, основними етапами є:

1. Розробити дизайн користувацького інтерфейсу.

1.1. Створити загальну концепцію дизайну користувацького інтерфейсу онлайн-системи.

1.1.1. Головна сторінка.

1.1.2. Вхід, реєстрація.

1.1.3. Відображення товарів.

1.1.4. Персональний кабінет користувача.

1.1.5. Сторінки для - Головна, Перевірка та Кабінет.

1.2. Створити загальну концепцію дизайну адміністративного інтерфейсу.

1.2.1. Головна сторінка адміністративного кабінету.

1.2.2. Сторінки CRUD для користувачів, товарів, атрибутів, категорій.

2. Проектування та розробка основних модулів.

2.1. Модуль керування користувачами.

2.2. Модуль керування товарами.

2.3. Модуль керування атрибутами.

2.4. Модуль керування категоріями.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						9

1.2 Аналіз аналогів програмного продукту

Потрібно проаналізувати аналоги онлайн-системи для продажу будівельних матеріалів, щоб створити дійсно конкуруючий продукт. Було виділено 3 такі аналоги: БудМарт, САД та Віконт.

1) БудМарт – житомирський магазин будматеріалів, заснований 2010 року (рис. 1.1, рис.1.2).

Переваги:

- Зручний для навігації інтерфейс
- Гарні рекомендації товарів(категорія «хіти»)

Недоліки:

- Немає вибору виробника
- Неможливість онлайн-придбання багатьох товарів(відсутні сторінки з цими товарами)

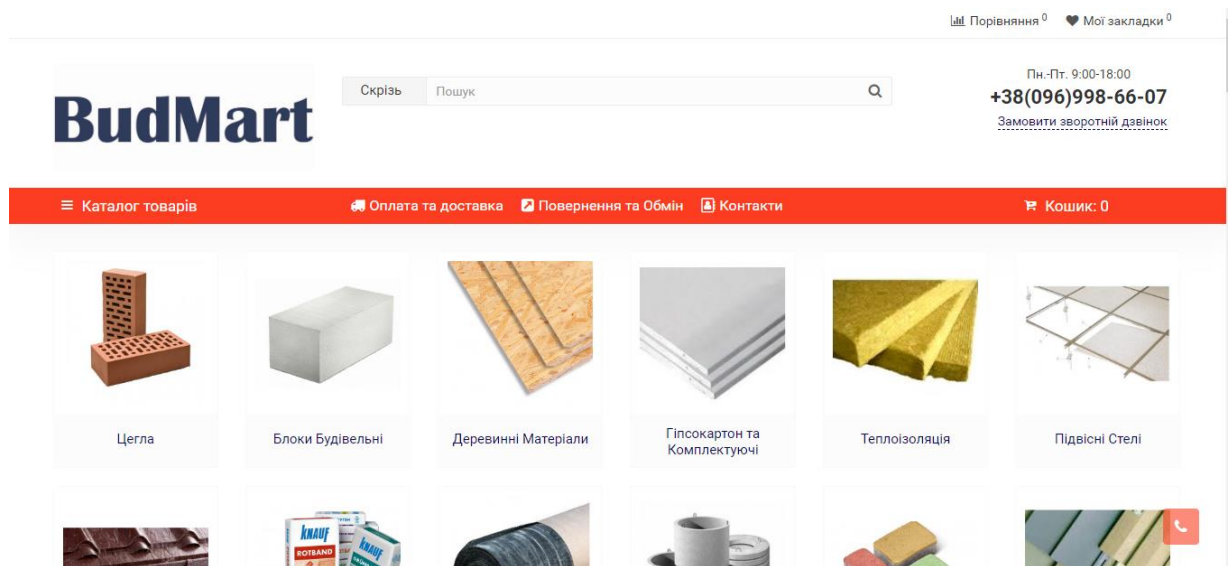


Рис. 1.1. Головна сторінка сайту БудМарт[16]

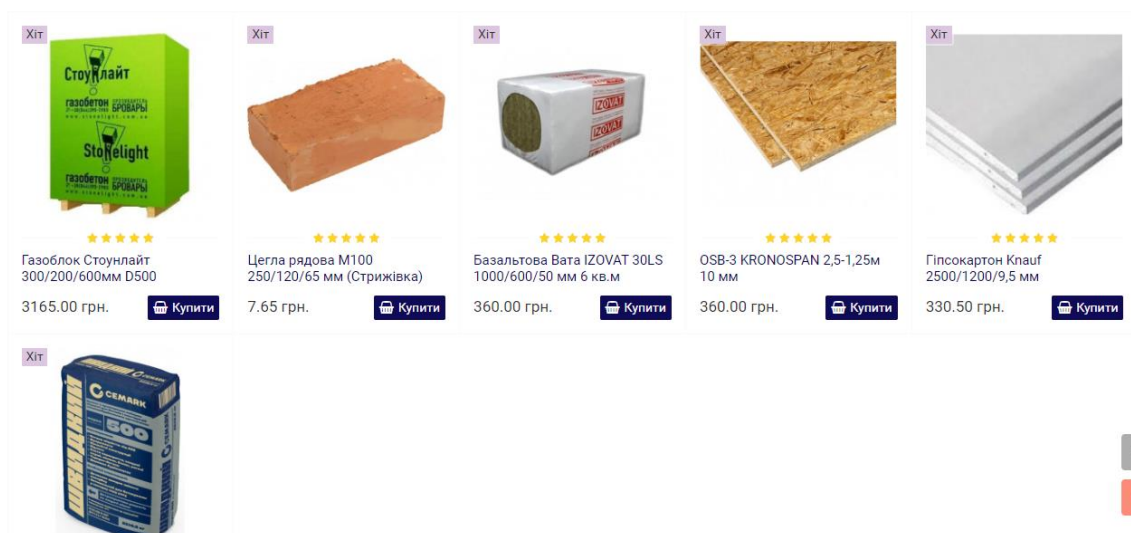


Рис. 1.2. Головна сторінка сайту БудМарт з товарами[16]

2) САД - українська компанія, що спеціалізується на продажу товарів для будівництва(рис. 1.3).

Переваги:

- Добре структурований дизайн
- Зручні переходи на сторінки сайту

Недоліки:

- Відсутність кошика
- Працює тільки по Житомиру та області

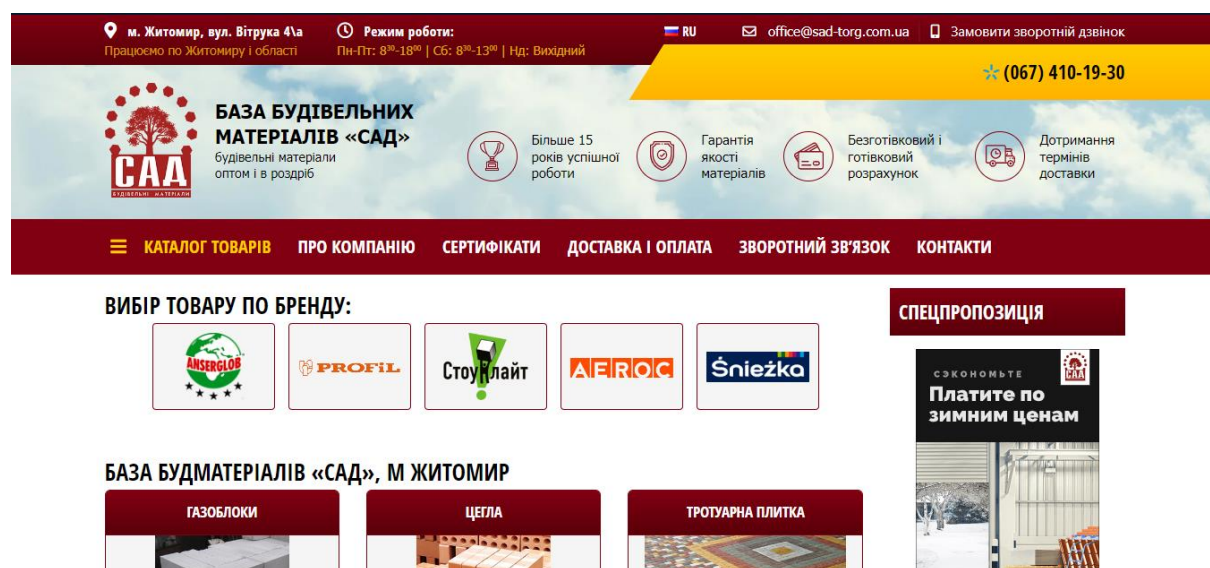


Рис. 1.3. Головна сторінка сайту САД[16]

3) ВІКАНТ – найбільший постачальник металопродукту в Україні в сегменті промислового та інфраструктурного будівництва(рис. 1.4)

Переваги:

- великий асортимент металопродукту;
- Мінімалістичний дизайн сайту

Недоліки:

- Нестабільна робота сайту

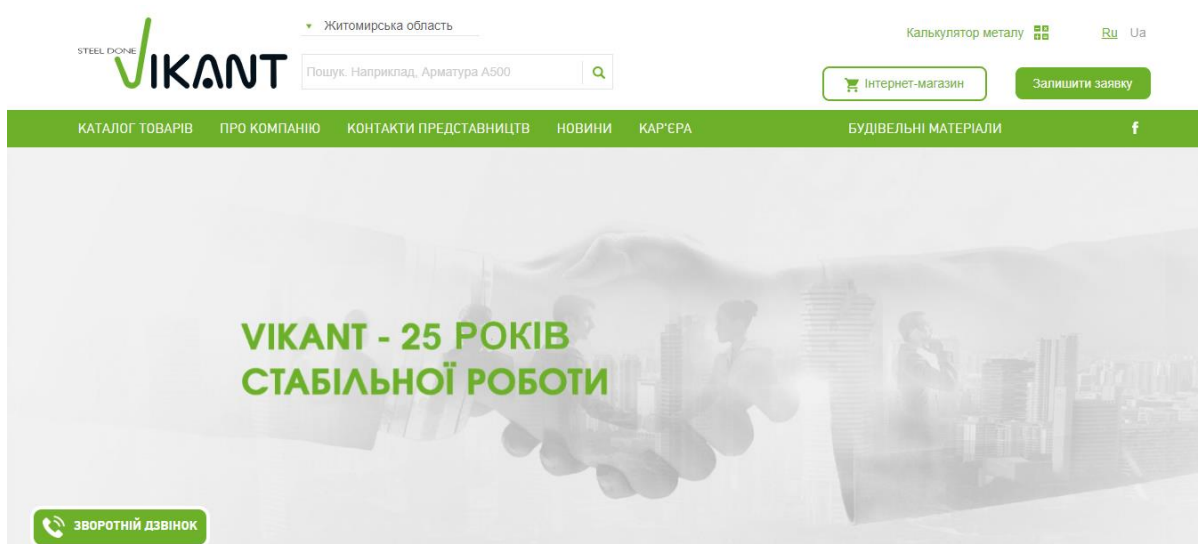


Рис. 1.4. Головна сторінка сайту Вікант[18]

Таблиця 1.1

Характеристики сайтів-аналогів

Характеристика	БудМарт	САД	Вікант
Дизайн	Сучасний	Простий	Сучасний
Навігація	Зручна	Зручна	Зручна
Асортимент	Широкий	Широкий	Великий
Адаптивність	Є	Є	Є
Можливість реєстрації та авторизації	Є	Є	Є

В таблиці 1.1, яка наведена вище порівняно, певні характеристики інтернет-магазинів, які я описував вище. Аналізуючи ці дані можна зробити висновок що в загальному інтернет-магазини дуже схожі та всі вони максимально зручні у використанні.

1.3 Обґрунтування вибору архітектури інформаційного додатка

Вибір архітектурного підходу до розробки онлайн-системи по продажу будівельних матеріалів є важливим етапом проектування. Один з найбільш поширених підходів - це архітектура MVC (Model-View-Controller) [1]. Вона дозволяє розділити логіку магазину на три компоненти: модель, представлення та контролер, що забезпечує чітку структуру проекту, спрощує тестування і підтримку проекту. Крім того, такий підхід сприяє масштабованості проекту та його розвитку.

Одна з ключових переваг архітектури MVC полягає в її гнучкості. Завдяки розділенню логіки на три компоненти, зміни в одному компоненті не впливають на інші компоненти. Це дозволяє більш ефективно розвивати та підтримувати проект у майбутньому. Крім того, такий підхід дозволяє збільшити простоту коду і зменшити залежності між компонентами, що полегшує розробку та тестування проекту.

Хоча існують інші архітектурні підходи, такі як MVP (Model-View-Presenter) [2] та MVVM (Model-View-ViewModel) [3], використання архітектури MVC має свої переваги. Наприклад, MVC дозволяє збільшити продуктивність розробки завдяки можливості паралельної розробки окремих компонентів проекту. Крім того, MVC є досить простим у використанні, із зрозумілим розподілом обов'язків між різними компонентами, що дозволяє розробникам швидко орієнтуватися в коді(рис.1.5.).

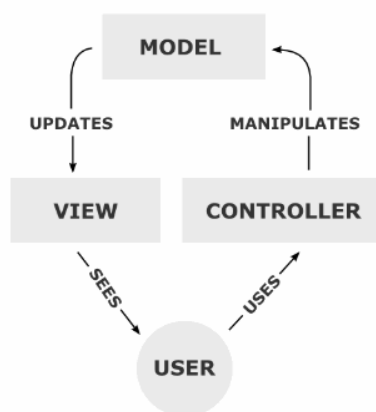


Рис. 1.5 - Архітектура MVC

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для створення онлайн-системи потрібно застосовувати: мову розмітки гіпертекстових документів HTML, каскадні таблиці стилів CSS, мову програмування JavaScript, мову програмування PHP, веб-сервер Apache з підтримкою PHP, MySQL-бази даних та фреймворк Laravel.

HTML (HyperText Markup Language) - це мова розмітки гіпертексту, що використовується для створення веб-сторінок. HTML використовує теги для описування структури документа, таких як заголовки, абзаци, списки та інші елементи. За допомогою HTML можна також вставляти зображення, відео, аудіо та інші медіа-елементи на сторінку. HTML не відповідає за стиль та поведінку сторінки, вона використовується для створення структури та контенту сторінки [4].

CSS (Cascading Style Sheets) - це файли з описом стилів, що використовується для задання вигляду та макету веб-сторінок. CSS дозволяє відокремити стиль від контенту, що дозволяє змінювати вигляд сторінки без зміни її структури та контенту. CSS дозволяє задавати кольори, шрифти, розміри, відступи, межі та інші стилізаційні властивості для елементів на сторінці [5].

JavaScript - це мова програмування, що використовується для додавання інтерактивності та динаміки на веб-сторінки. JavaScript дозволяє змінювати сторінку без перезавантаження, взаємодіяти з користувачем, валідувати форми та інші функції. Ця мова тісно пов'язана з HTML та CSS, і використовується для зміни стилів та DOM-елементів на сторінці [6].

PHP (Hypertext Preprocessor) - це мова програмування, що використовується для створення динамічних веб-сторінок та веб-додатків. PHP використовується для збору та обробки даних, взаємодії з базами даних, створення функцій, обробки форм та інші функції, що дозволяють створювати більш складні та функціональні веб-додатки. PHP використовується в поєднанні з HTML та CSS для створення динамічних веб-сторінок, які можуть

змінюватися в залежності від даних та дій користувача. Актуальна версія - 8.2 [7].

Laravel - безкоштовний, з відкритим кодом PHP-фреймворк, призначений для розробки вебдодатків відповідно до шаблону model–view–controller (MVC) [14].

JetBrains PhpStorm - універсальне середовище розробки, але з нахилом на розробку на мові PHP. Середовище, яке допомагає працювати з кодом за допомогою інтелектуального редактора, рефакторингу, підтримки генераторів і поліпшення синтаксису, підтримка фреймворків на PHP, гнучке налаштування плагінів, візуальна стилізація під користувача, редактор мови розмітки(HTML, CSS), слідування за стилями коду (PSR1/PSR2) [20], потужні інструменти для роботи з базами даних [8].

Було вирішено скористатись MySQL як реляційною базою даних, оскільки вона має велику швидкодію та ефективність, що дозволяє оптимально обробляти великі обсяги даних [9]. Крім того, MySQL підтримує розподілені транзакції та має потужну систему кешування для покращення продуктивності. Нам також сподобалась велика спільнота користувачів та широка підтримка, що робить MySQL легким у використанні та навчанні.

Розглянувши варіанти технологій було обрано:

- Мову розмітки документів HTML5;
- стилі CSS3;
- мову програмування JavaScript та бібліотеку jQuery;
- фреймворк Laravel 10 (PHP8.1);
- середовище керування базами даних MySQL.

Для того щоб мати змогу зайти на дану онлайн-систему, система користувача повинна задовольняти вимогам на комп'ютері, що знаходиться в наступній мінімальній комплектації:

- Windows 10, Linux, MacOS;
- 32-розрядний або 64-розрядний процесор з тактовою частотою 1 ГГц та вище;

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						15

- 4 Гбайти ОЗП;
- 24 Гбайти, за можливості більше вільного місця на жорсткому диску;
- комп'ютер користувача виконує роль сервера.

Висновки до першого розділу

Проведений аналіз дозволив визначити базові напрямки для реалізації онлайн-системи.

Було розглянуте аналогічне програмне забезпечення, а саме БудМарт, САД, Вікант, де можна помітити, що основними функціями є: головна сторінка з товарами, можливість перегляду товарів за категорією, можливість реєстрації та авторизації всередині додатку.

Визначившись з базовою архітектурою, був обраний достатній набір стеку технологій для реалізації поставлених задач.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ІНФОРМАЦІЙНОГО ДОДАТКА

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

Онлайн-система створюється для того, щоб в подальшому його можна було використовувати для продажу будівельних матеріалів, що дасть можливість для збільшення кількості клієнтів.

Вимоги користувачів.

Гість має доступ до наступних функцій (рис. 2.1):

1. до інформації з головної сторінки;
2. до перегляду окремих товарів;
3. до написання відгуків до товарів;
4. до перегляду всіх прохань про допомогу та їх детальну інформацію;
5. до реєстрації на сайті;
6. до авторизації на сайті.

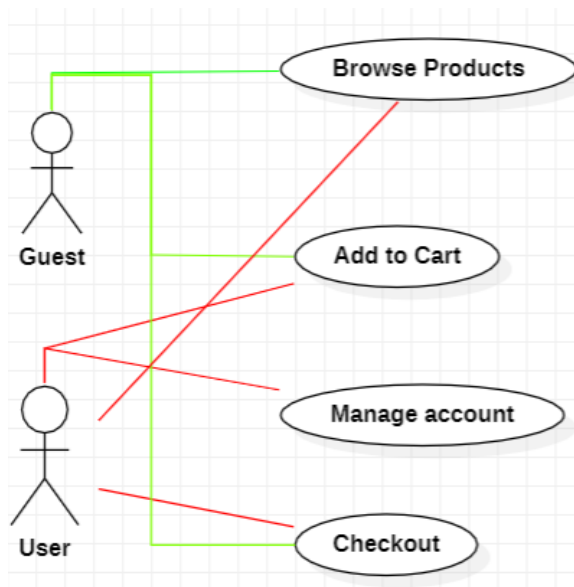


Рис. 2.1. Діаграма варіантів використання гостя

Авторизований користувач має доступ до наступних функцій (рис. 2.2):

1. до інформації з головної сторінки;
2. до перегляду окремого товару та інформації про неї;

3. до коментарів до товарів;
4. до можливості виставлення певного адресу куди будуть йти замовлення по замовчуванню в особистому кабінеті;
5. до особистого кабінету;
6. до CRUD з товарами;
7. до відгуків;
8. до виходу с облікового запису.

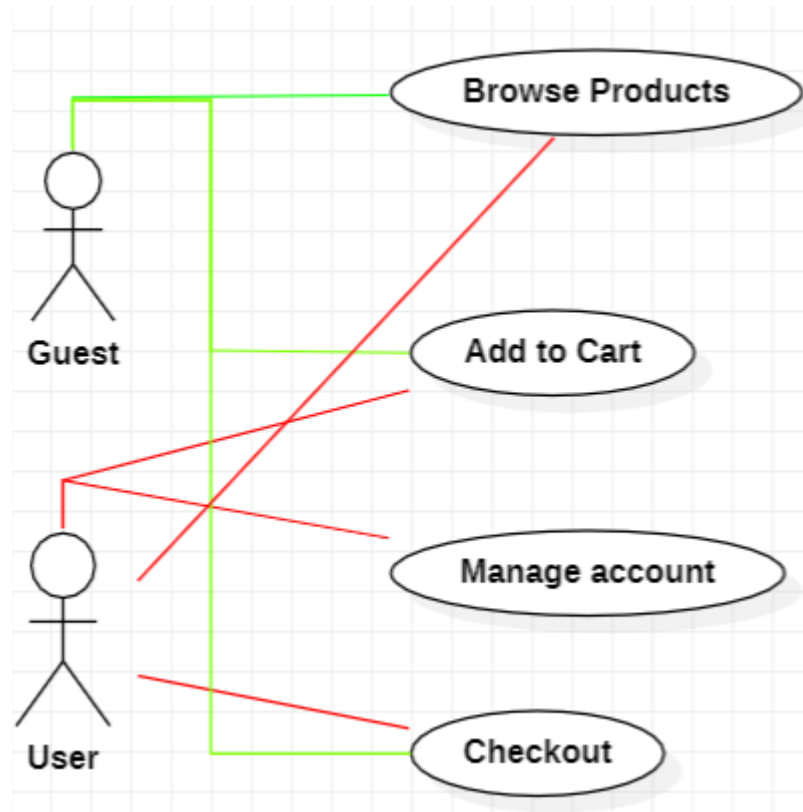


Рис. 2.2. Діаграма варіантів використання авторизованого користувача
Адміністратор має доступ до наступних функцій(рис. 2.3):

1. до CRUD з категорій;
2. до CRUD з замовлень;
3. до CRUD з товарів;
4. до CRUD з користувачів.

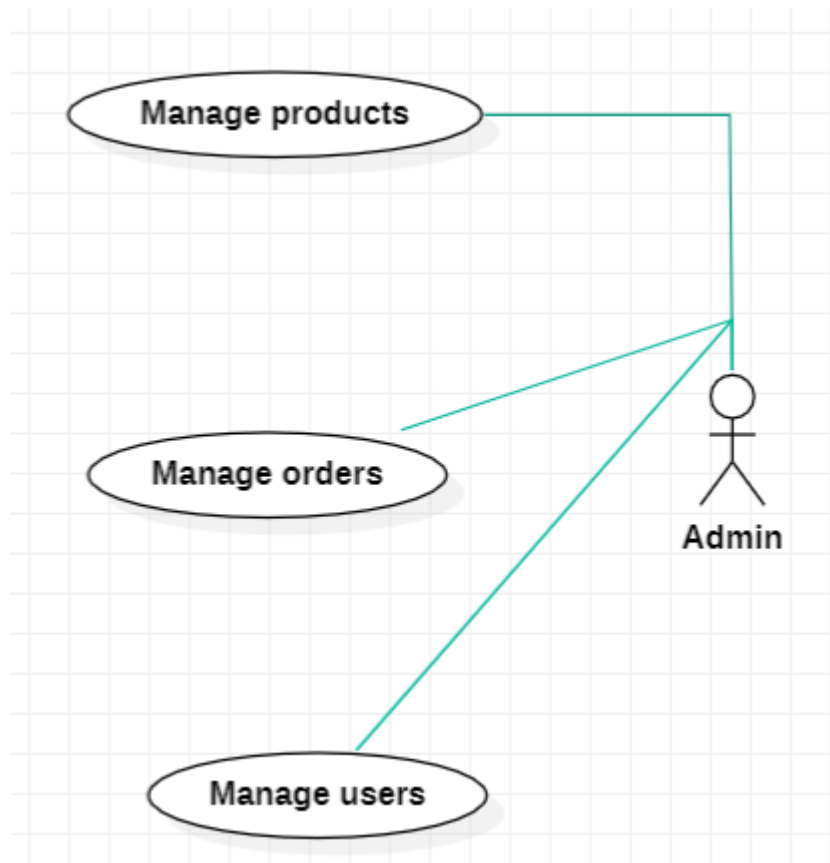


Рис. 2.3. Діаграма варіантів використання адміністратора
Функціональні вимоги:

1. Реєстрація та авторизація. Для користувачів необхідна можливість реєстрації або входу в уже створений акаунт. Необхідна реалізація можливості додавання користувачів з адміністративного кабінету.
2. Валідація: усі дані, які вводить користувач повинні проходити валідацію.
3. Відображення результатів виконання операцій: користувач повинен отримувати повідомлення про результат виконання будь-яких операцій, таких як додавання товару в кошик, видалення товару зі списку бажаного та інші. Ці повідомлення повинні містити інформацію про статус виконання операції, який може бути успішним або неуспішним.
4. Створення, редагування, перегляд та видалення програм допомоги, атрибутів, категорій та користувачів з адмін-панелі.
5. Конфіденційність: система повинна зберігати конфіденційність користувачів.

6. Система повинна коректно та автоматично відображати необхідну інформацію.

Нефункціональні вимоги:

Сприйняття

1. Сприйняття:

- Час, потрібний для початку роботи - 3 хв.
- Час відповіді системи для звичайних запитів не повинен перевищувати 0.540 мс, а для більш складних запитів – 5 с.
- Інтерфейс повинен бути зрозумілим для користувачів.

2. Надійність:

- Середній час безперервної роботи - 55 год.
- Максимальна норма помилок та дефектів в роботі програми - 1 помилка на 3000 запитів користувача.

3. Швидкість. Сайт повинен швидко виконувати запити користувачів.

4. Можливості використання:

- Масштабування - після додавання функціонала не повинно бути фатальних помилок.

Аналіз функціональних вимог дозволив виявити наступні ключові елементи, що забезпечать розробку програмного комплексу для цієї системи. На рисунку 2.4 наведена діаграма класів.

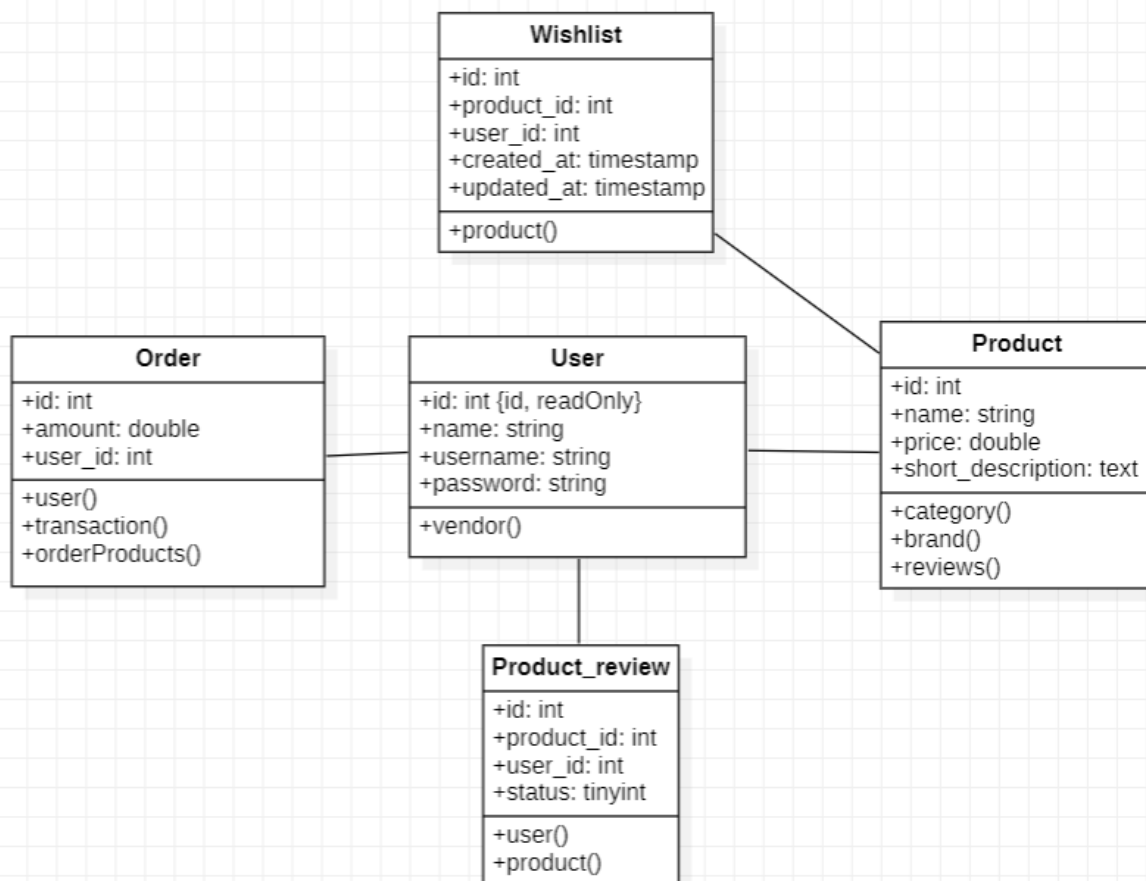


Рис. 2.4. Діаграма класів

На рисунку 2.4 можна виділити наступні класи:

Product – клас, який містить методи для роботи зі всіма сутностями товару.

Поля:

- id (тип int) – унікальний ідентифікатор;
- name (тип string) – назва товару;
- short_description (text) – опис товару;
- price (double) – ціна товару;

Методи:

- category(): Зв’язка товару з категорією.
- brand(): Зв’язка товару з його брендом.
- reviews(): Зв’язка товару з оглядом на нього.

Product_review – клас, який містить методи для роботи із описом товару.

Поля:

- id (тип int) – унікальний ідентифікатор;
- status (tinyint): Статус огляду товару.
- product_id (int): Ідентифікатор товару, до якого належить огляд.
- user_id (int): Ідентифікатор користувача, якому належить огляд.

Методи:

- user(): Зв'язка огляду з користувачем який його залишив.
- product(): Зв'язка огляду з товаром на який він був залишений.

2.2 Розробка бази даних системи

Кожна онлайн-система потребує бази даних для зберігання необхідної інформації, яка може змінюватися в залежності від взаємодії користувачів з системою. База даних дозволяє зберігати, оновлювати та видаляти дані, а також забезпечує доступ до цих даних для різних компонентів системи, що дозволяє реалізувати багато функцій та можливостей для користувачів додатку.

В онлайн-системі продажу будівельних матеріалів було використано ХАМРР[19] для керування базою даних.

Першим кроком при розробці бази даних було її створення та підключення до проекту. Для цього я створив базу даних в ХАМРР напямую, і після цього виконав певні налаштування в файлі .env для того, щоб під'єднати базу даних.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=diploma
DB_USERNAME=root
DB_PASSWORD=
```

Рис. 2.5. Фрагмент з файлу .env, який відповідає за підключення до бази даних

Кожен рядок відповідає за певний параметр підключення бази даних.

DB_CONNECTION=mysql – вказує на тип підключення до бази даних, у цьому конкретному випадку MySQL.

DB_HOST=127.0.0.1 – вказує на адресу бази даних. Через те, що локальний пристрій виступає в ролі сервера, тому IP-адреса така - 127.0.0.1

DB_PORT=3306 – вказує на порт, через який проходить підключення до бази даних.

DB_DATABASE=diploma – вказує на назву бази даних, яку було зазначено в XAMPP при створенні.

DB_USERNAME=root – вказує на логін користувача для авторизації в XAMPP.

DB_PASSWORD= - вказує на пароль, який повинен ввести користувач, щоб отримати доступ до бази даних.

Після підключення бази даних я створював моделі для компонентів за допомогою команди «php artisan make:model Назва моделі –m» у терміналі. Ця команда створює модель та відповідну таблицю для міграції. Команди виконуються за допомогою вбудованого в Laravel консольного інтерфейсу artisan [15]. Далі можна додати необхідні поля до таблиці та визначити їх типи. Наступним кроком були налаштовані створеної моделі, додавши до неї рядок «protected \$guarded = [];», що дозволяє масове заповнення всіх полів таблиці.

Після створення файлу міграції, додавались необхідні поля для таблиці та вказував їх типи. Для більшості полів, де можливо, використовувались тип

`string`, оскільки це дозволяє зберігати будь-який текстовий контент, такий як рядки, URL-адреси або JSON-об'єкти, і є досить простим для роботи з ним. Однак, для полів, які містять цифрові значення, використовувався тип `int`. Цей тип дозволяє зберігати числа без десяткової частини, тому цей тип добре підходить для зберігання даних стовпців по типу `category\_id` або `brand\_id`, оскільки в цих стовпцях присутні тільки невеликі числа.

```

*/
public function up(): void
{
    Schema::create( table: 'products', function (Blueprint $table) {
        $table->id();
        $table->string( column: 'name');
        $table->string( column: 'slug');
        $table->text( column: 'thumb_image');
        $table->integer( column: 'vendor_id');
        $table->integer( column: 'category_id');
        $table->integer( column: 'sub_category_id')->nullable();
        $table->integer( column: 'child_category_id')->nullable();
        $table->integer( column: 'brand_id');
        $table->integer( column: 'qty');
        $table->text( column: 'short_description');
        $table->text( column: 'long_description');
        $table->text( column: 'video_link')->nullable();
        $table->string( column: 'sku')->nullable();
        $table->double( column: 'price');
        $table->double( column: 'offer_price')->nullable();
        $table->date( column: 'offer_start_date')->nullable();
        $table->date( column: 'offer_end_date')->nullable();
        $table->string( column: 'product_type')->nullable();
        $table->boolean( column: 'status');
        $table->integer( column: 'is_approved')->default( value: 0);
        $table->string( column: 'seo_title')->nullable();
        $table->text( column: 'seo_description')->nullable();

        $table->timestamps();
    });
}

```

Рис. 2.6. Фрагмент коду з файлу, який використовується для міграції (див. додаток А)

Після виконання цих кроків потрібно виконати міграцію до бази даних наступною командою «php artisan migrate». Ця команда виконує міграцію усіх таблиць, яких ще немає в БД, тобто створює таблицю в ХАМРР, яку можна надалі використовувати для записування даних, отримання цих даних, або їх змінювати(рис. 2.7, 2.8).

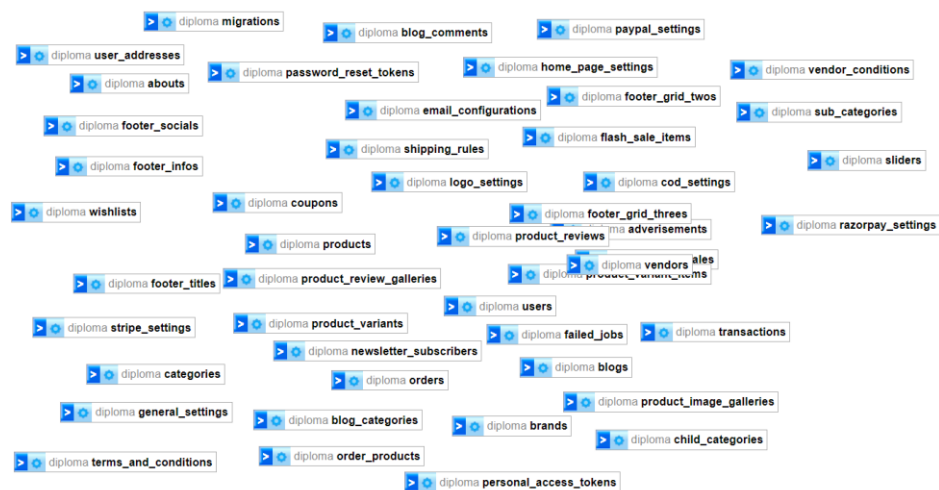


Рис. 2.7. Зменшена діаграма бази даних



Рис. 2.8. Діаграма бази даних

Було вирішено обрати саме такий спосіб створення бази даних, тому що він дуже зручний і економить дуже багато часу.

2.3 Проектування та реалізація алгоритмів роботи системи

У даній системі адміністратор має ключову роль у виконанні дій з

товарами. Розглянемо алгоритм роботи адміністратора(рис.2.9):

- автентифікація в обліковому записі з роллю Адміністратор;
- створення товарів;
- збереження створених товарів у базу даних;
- перегляд, редагування, видалення певних елементів товару або й товару повністю;
- збереження оновленої інформації.



Рис. 2.9. Діаграма активності адміністратора

Було розглянути можливі варіанти взаємодії з додатком зі сторони адміністратора та користувача.

2.4 Реалізація функціоналу онлайн-системи

Для реалізації системи було вибрано фреймворк Laravel. Laravel використовує розширення залежностей, такі як: маршрути, контролери, моделі та представлення. Компоненти взаємодіють між собою за допомогою вбудованих методів.

Першим кроком до побудови системи буде реалізація маршрутів(роутинг). Код представлено у ДОДАТКУ А. Побудувавши ієрархію маршрутів переходимо до створення контролерів. Розглянемо контролери зі сторони адміністратора. Опис кожного методу цих контролерів наведений у таблицях 2.1-2.5.

Контролер AdminController відповідає за відображення головної сторінки адмін-панелі зі швидким доступом до різних функцій.

```
public function dashboard()
{
    $todaysOrder = Order::whereDate('created_at',
Carbon::today())->count();
    $todaysPendingOrder = Order::whereDate('created_at',
Carbon::today())
->where('order_status', 'pending')->count();
    $totalOrders = Order::count();
    $totalPendingOrders = Order::where('order_status',
'pending')->count();
    $totalCanceledOrders = Order::where('order_status',
'canceled')->count();
    $totalCompleteOrders = Order::where('order_status',
'delivered')->count();

    $todaysEarnings = Order::where('order_status','!=',
'canceled')
->where('payment_status',1)
->whereDate('created_at', Carbon::today())
->sum('sub_total');

    $monthEarnings = Order::where('order_status','!=',
'canceled')
->where('payment_status',1)
->whereMonth('created_at', Carbon::now()->month)
->sum('sub_total');

    $yearEarnings = Order::where('order_status','!=',
'canceled')
->where('payment_status',1)
->whereYear('created_at', Carbon::now()->year)
->sum('sub_total');

    $totalReview = ProductReview::count();
}
```

```

$totalBrands = Brand::count();
$totalCategories = Category::count();
$totalBlogs = Blog::count();
$totalSubscriber = NewsletterSubscriber::count();
$totalVendors = User::where('role', 'vendor')->count();
$totalUsers = User::where('role', 'user')->count();

```

```

return view('admin.dashboard', compact(
    'todaysOrder',
    'todaysPendingOrder',
    'totalOrders',
    'totalPendingOrders',
    'totalCanceledOrders',
    'totalCompleteOrders',
    'todaysEarnings',
    'monthEarnings',
    'yearEarnings',
    'totalReview',
    'totalBrands',
    'totalCategories',
    'totalBlogs',
    'totalSubscriber',
    'totalVendors',
    'totalUsers'
));

```

```

}

```

```

public function login()
{
    return view('admin.auth.login');
}

```

```

}

```

Методи DashboardController

Назва	Опис та призначення
1	2
dashboard	Метод для відображення головної сторінки адмін-панелі
login	Метод для відображення сторінки авторизації адміна

Контролер товарів ProductController відповідає за роботу із товарами, їх створення, редагування, перегляд та видалення.

```

public function index(ProductDataTable $dataTable)
{
    return $dataTable->render('admin.product.index');
}

/**
 * Show the form for creating a new resource.
 */
public function create()
{
    $categories = Category::all();
    $brands = Brand::all();
    return view('admin.product.create', compact('categories',
'brands'));
}

/**
 * Store a newly created resource in storage.
 */
public function store(Request $request)
{
    $request->validate([
        'image' => ['required', 'image', 'max:3000'],
        'name' => ['required', 'max:200'],
        'category' => ['required'],
        'brand' => ['required'],
        'price' => ['required'],
        'qty' => ['required'],
        'short_description' => ['required', 'max: 600'],
        'long_description' => ['required'],
        'seo_title' => ['nullable', 'max:200'],
        'seo_description' => ['nullable', 'max:250'],
        'status' => ['required']
    ]);

    /** Handle the image upload */
    $imagePath = $this->uploadImage($request, 'image', 'uploads');

    $product = new Product();
    $product->thumb_image = $imagePath;

```

```

$product->name = $request->name;
$product->slug = Str::slug($request->name);
$product->vendor_id = Auth::user()->vendor->id;
$product->category_id = $request->category;
$product->sub_category_id = $request->sub_category;
$product->child_category_id = $request->child_category;
$product->brand_id = $request->brand;
$product->qty = $request->qty;
$product->short_description = $request->short_description;
$product->long_description = $request->long_description;
$product->video_link = $request->video_link;
$product->sku = $request->sku;
$product->price = $request->price;
$product->offer_price = $request->offer_price;
$product->offer_start_date = $request->offer_start_date;
$product->offer_end_date = $request->offer_end_date;
$product->product_type = $request->product_type;
$product->status = $request->status;
$product->is_approved = 1;
$product->seo_title = $request->seo_title;
$product->seo_description = $request->seo_description;
$product->save();

toastr('Created Successfully!', 'success');

return redirect()->route('admin.products.index');
}

```

Таблиця 2.2

Методи ProductController

Назва	Опис та призначення
1	2
index	Метод для відображення всіх продуктів, які доступні в онлайн-системі
create	Метод, який відображає сторінку для створення нового продукту
store	Метод, який відображає сторінку для збереження інформації про продукт

Ресурсний Контролер CategoryController відповідає за зміни у таблиці бази даних categories.

```

public function index(CategoryDataTable $dataTable)
{
    return $dataTable->render('admin.category.index');
}

/**
 * Show the form for creating a new resource.

```

```

*/
public function create()
{
    return view('admin.category.create');
}

/**
 * Store a newly created resource in storage.
 */
public function store(Request $request)
{
    $request->validate([
        'icon' => ['required', 'not_in:empty'],
        'name' => ['required', 'max:200',
'unique:categories,name'],
        'status' => ['required']
    ]);

    $category = new Category();

    $category->icon = $request->icon;
    $category->name = $request->name;
    $category->slug = Str::slug($request->name);
    $category->status = $request->status;
    $category->save();

    toastr('Created Successfully!', 'success');

    return redirect()->route('admin.category.index');
}

/**
 * Show the form for editing the specified resource.
 */
public function edit(string $id)
{
    $category = Category::findOrFail($id);
    return view('admin.category.edit', compact('category'));
}

```

Таблиця 2.3

Методи CategoryController

Назва	Опис та призначення
1	2
index	Метод для відображення всіх категорій з бази даних
create	Метод для створення нової категорії
store	Метод зберігає нову категорію в базі даних
edit	Метод, який відображає сторінку для редагування інформації про категорію

Контролер UserProfileController відповідає за CRUD користувачів.

```
public function index(CategoryDataTable $dataTable)
{
    return $dataTable->render('admin.category.index');
}

/**
 * Show the form for creating a new resource.
 */
public function create()
{
    return view('admin.category.create');
}

/**
 * Store a newly created resource in storage.
 */
public function store(Request $request)
{
    $request->validate([
        'icon' => ['required', 'not_in:empty'],
        'name' => ['required', 'max:200',
'unique:categories,name'],
        'status' => ['required']
    ]);

    $category = new Category();

    $category->icon = $request->icon;
    $category->name = $request->name;
    $category->slug = Str::slug($request->name);
    $category->status = $request->status;
    $category->save();

    toastr('Created Successfully!', 'success');

    return redirect()->route('admin.category.index');
}
public function edit(string $id)
{
    $category = Category::findOrFail($id);
    return view('admin.category.edit', compact('category'));
}

/**
 * Update the specified resource in storage.
 */
public function update(Request $request, string $id)
{
    $request->validate([
        'icon' => ['required', 'not_in:empty'],
        'name' => ['required', 'max:200',
```

```

'unique:categories,name,'.$id],
  'status' => ['required']
]);

$category = Category::findOrFail($id);

$category->icon = $request->icon;
$category->name = $request->name;
$category->slug = Str::slug($request->name);
$category->status = $request->status;
$category->save();

toastr('Updated Successfully!', 'success');

return redirect()->route('admin.category.index');
}

```

Таблиця 2.4

Методи UserController

Назва	Опис та призначення
1	2
index	Метод для відображення всіх користувачів з бази даних
create	Метод, який відображення сторінки реєстрації нового користувача
store	Метод зберігає дані нового користувача в базі даних
edit	Метод, який відображає сторінку редагування профілю користувача
update	Метод, який оновлює попередні дані у базі даних

Контролер OrderController створений для управління CRUD замовлень.

```

public function index(OrderDataTable $dataTable)
{
    return $dataTable->render('admin.order.index');
}
public function show(string $id)
{
    $order = Order::findOrFail($id);
    return view('admin.order.show', compact('order'));
}
public function destroy(string $id)
{
    $order = Order::findOrFail($id);
    // delete order products
    $order->orderProducts()->delete();
    // delete transaction
    $order->transaction()->delete();
}

```

```

        $order->delete();
        return response(['status' => 'success', 'message' => 'Deleted
successfully!']);
    }

    public function changeOrderStatus(Request $request)
    {
        $order = Order::findOrFail($request->id);
        $order->order_status = $request->status;
        $order->save();
        return response(['status' => 'success', 'message' => 'Updated
Order Status']);
    }

```

Таблиця 2.5

Методи OrderController

Назва	Опис та призначення
1	2
index	Метод для відображення список замовлень які зберігаються в базі даних
changeOrderStatus	Метод використовується для зміни статусу замовлення
destroy	Метод використовується для скасування замовлення
Show	Метод для перегляду певного замовлення

Висновки до другого розділу

Були створені декілька діаграм для опису алгоритму роботи системи. Розпочавши з аналізу діаграм варіантів використання, були розглянуті ролі адміністратора, зареєстрованого користувача, гостя.

Було продемонстровано створення бази даних з необхідним для сайту функціоналом. База даних складається зі 120 таблиць, які взаємодіють між собою, з них було розглянуто основні 7.

Було розглянуто основні методи та технології для продуктивної роботи з додатком.

Було детально розглянута частина функціонала, точніше контролери які відповідають за створення сутностей з адміністративного кабінету.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКА

3.1 Порядок встановлення та налаштування системи

Для роботи з системою на локальному пристрої необхідно, щоб були встановлені певні налаштування комп'ютера. Для коректної роботи Laravel та його вбудованих методів необхідні такі встановлення: npm/yarn, node.js, PHP (версії 8.1), composer, MySQL, PhpStorm. Опишемо детальніше встановлення цих всіх технологій.

1. npm на Windows необхідно завантажити exe-файл з офіційного сайту та йти по пунктам інструкції інсталятора [10].
2. Щоб встановити Node.js, потрібно перейти на сайт для завантаження інсталятора та виконати дії, необхідні для встановлення програми [11].
3. Щоб встановити Composer, необхідно дотримуватись інструкцій, які наведені на офіційному сайті [12].
4. Для встановлення PHP потрібно завантажити архів з офіційного сайту та обрати необхідну версію. Після завантаження архіву, необхідно додати шлях до файлу php.exe у реєстр Windows [7].
5. Для встановлення MySQL потрібно виконати інструкції, які наведені у завантаженому інсталяторі з офіційного сайту [13].

Після налаштування та встановлення необхідних компонентів, наступним кроком є розгортання даного проекту в середовищі розробки, яке в нашому випадку є PhpStorm. Для цього необхідно створити базу даних для проекту, що можна зробити за допомогою терміналу MySQL або візуального середовища MySQL Workbench. Після створення бази даних, необхідно змінити конфігураційний файл .env проекту, де поля, що відповідають за конфігурацію бази даних, необхідно заповнити даними, що були створені:

```
DB_DATABASE=example
DB_USERNAME=root
DB_PASSWORD=
```

Після того, як були зроблені зміни до конфігураційного файлу, необхідно ініціалізувати проект, введенням команди **php artisan serve**. Після успішного встановлення всіх залежностей, необхідно виконати міграції, що можна зробити за допомогою команди **php artisan migrate**.

3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з онлайн-системою для продажу будівельних матеріалів

Головна сторінка знаходиться за адресою <http://127.0.0.1:8000>. Коли користувач перейде за посиланням, тоді він потрапить на головну сторінку сайту і зможе почати користуватись його функціоналом(рис. 3.1).

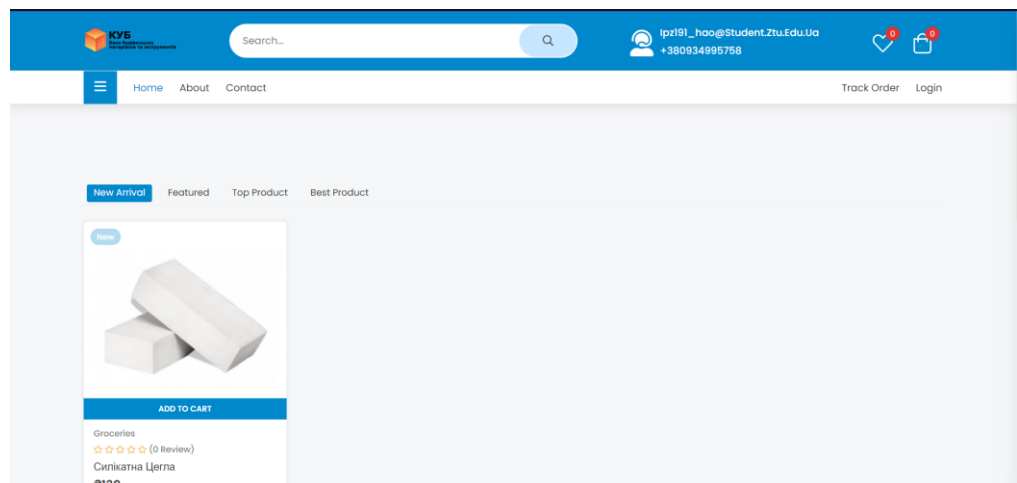


Рис. 3.1. Головна сторінка “Куб”

Якщо прогорнути головну сторінку вниз, користувач побачить товари(рис. 3.2).

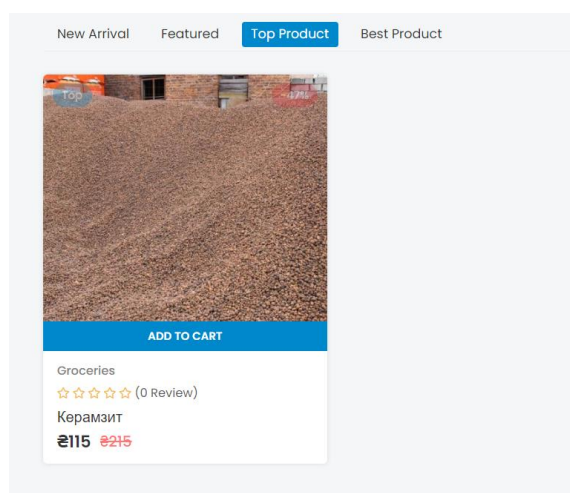


Рис. 3.2. Товари на головній сторінці

Незареєстрований користувач може тільки додати товар в кошик, але щоб далі купувати, потрібно зайти в свій профіль(рис. 3.3).

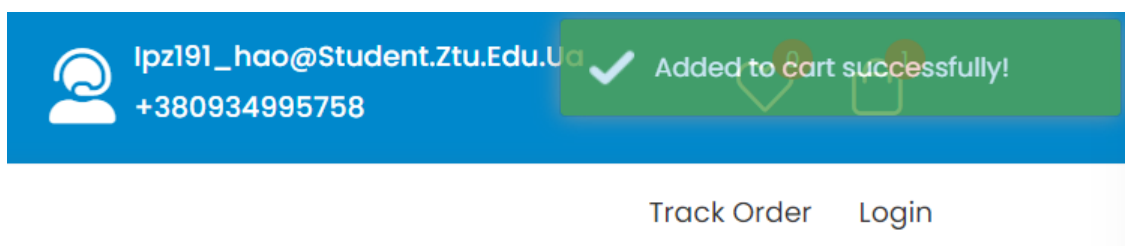


Рис. 3.3. Повідомлення про успішне додавання товару у кошик

Тому користувачу краще перейти по кнопці «Login» в правому верхньому куті сторінки(рис. 3.4).

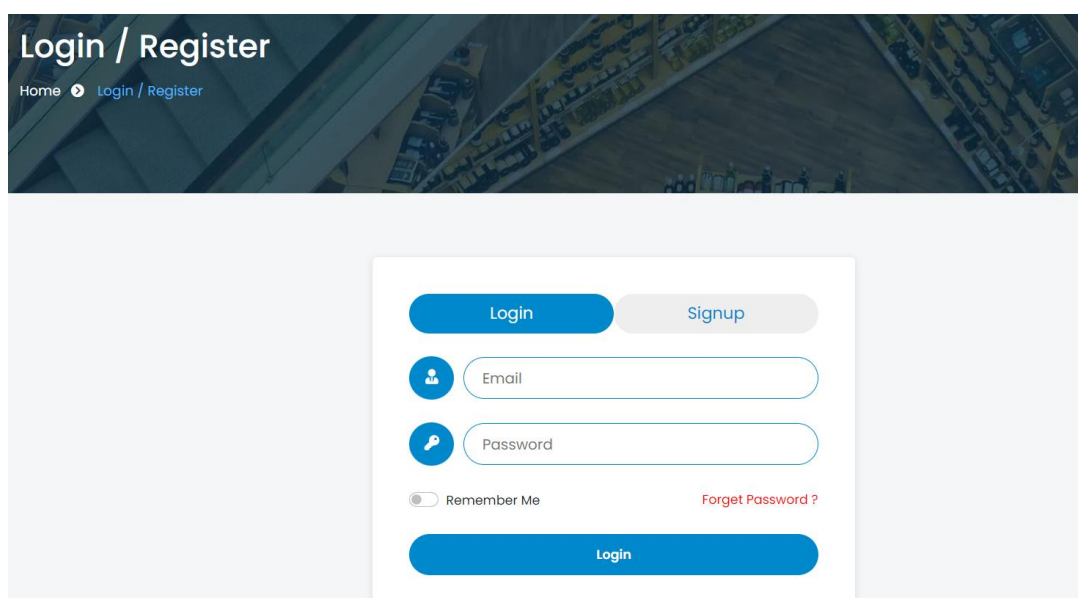


Рис. 3.4. Кнопки «Login» та «Signup»

Відкривши сторінку входу, користувач матиме змогу зайти в свій акаунт. На рисунку відразу видно що валідація рядків форми логіну працює, бо рядки пошти і пароллю не мають бути порожніми(рис. 3.5).

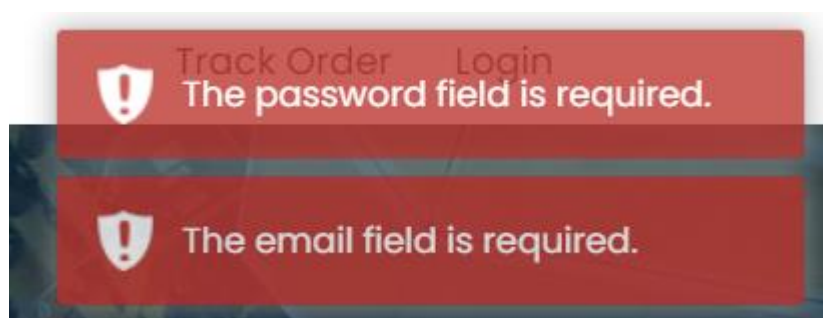


Рис. 3.5. Сторінка логіну

Якщо вхід був успішний, користувач потрапить на сторінку його профілю(рис. 3.6).

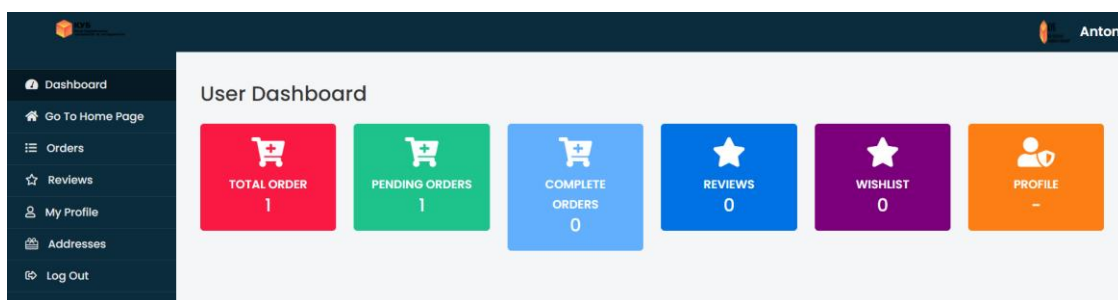


Рис. 3.6. Профіль зареєстрованого користувача

Але, якщо користувач не був зареєстрований раніше, тоді потрібно зареєструватись в системі, тому користувач повинен перейти по кнопці «Signup» (рис. 3.7).

Рис. 3.7. Форма реєстрації

На формі видно, що валідація не дає ввести пусті або неправильні значення в них, тому користувачу нічого не залишається, як ввести все правильно(рис. 3.8).

Рис. 3.8. Користувач правильно ввів всі дані

Тепер, коли користувач авторизований, він перейде на вкладку “Orders”, і тоді побачить перелік своїх замовлень(рис. 3.9).

Orders

Show entries Search:

Id	Invoice Id	Customer	Date	Product Qty	Amount	Order Status	Payment Status	Payment Method	Action
13	927102	Anton	29-Aug-2023	1	€440	pending	pending	COD	

Showing 1 to 1 of 1 entries Previous Next

Рис. 3.9. Сторінка “Orders”

На сторінці замовлень користувач має можливість детально переглянути свої минулі замовлення, також ті що зараз знаходяться в обробці.

Тепер потрібно детально оглянути товар. Користувач бачить основну та детальну інформацію про товар, наявність, ціну, та кнопку «Add to cart» (рис. 3.10).



Рис. 3.10. Детальний огляд товару

При виборі товарів за категорією користувач отримає перелік товарів за його вибором. Виберемо варіант «Рідкі матеріали» (рис. 3.11).

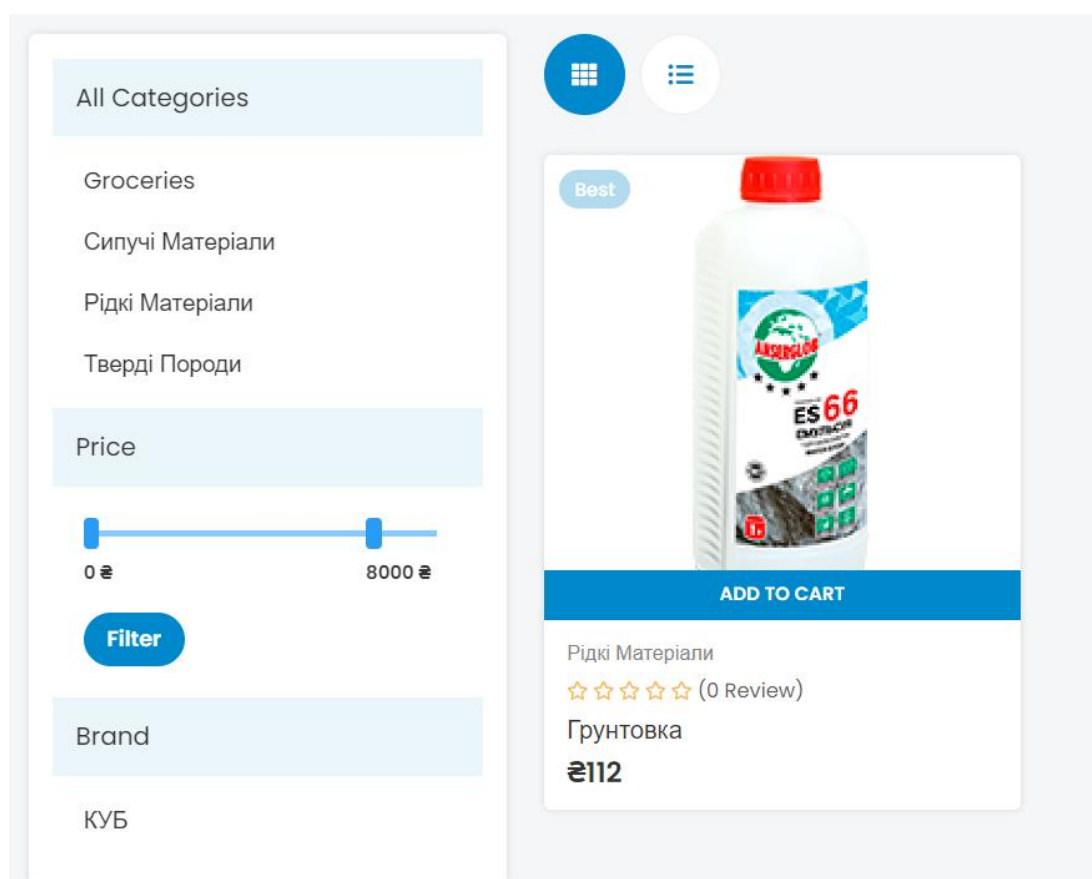


Рис. 3.11. Перелік товарів у категорії «Рідкі матеріали».

Розглянемо можливості та функціонал адміністративного кабінету. При авторизації з логіном і паролем адміністратора, буде відкрита головна сторінка адміністративного кабінету, а саме <http://127.0.0.1:8000/admin/dashboard>. Головна сторінка відображає переходи на функції адмінпанелі(рис. 3.12).

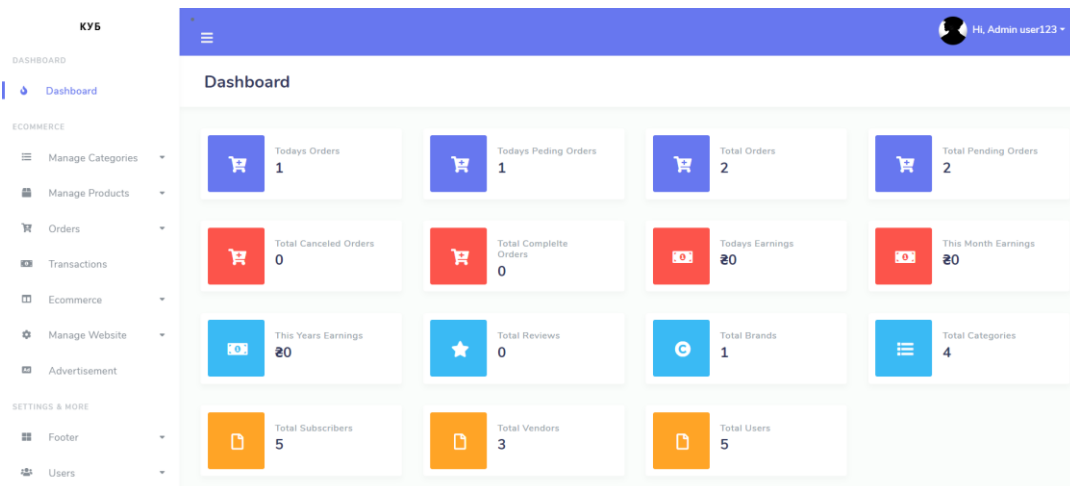


Рис. 3.12. Головна сторінка адмін-панелі

Розглянемо декілька можливих варіантів роботи з сутностями. Наприклад, “Users”. Адміністратор при переході на вкладку “Users” бачить перелік всіх зареєстрованих користувачів. Також має можливість деактивувати обліковий запис користувача(рис. 3.13).

Customer list

All customer			
Show	10	entries	Search:
Id	Name	Email	Status
11	Anton	test@gmail.com	☒
10	User 234	user123@gmail.com	☒
8	new user	newUser@gmail.com	☒
4	test user	testuser@gmail.com	☒
3	Jhon Deo	user@gmail.com	☒
Showing 1 to 5 of 5 entries			Previous 1 Next

Рис. 3.13. Сторінка “Users”

Розглянемо “Товари”. Реалізовано CRUD, на головній сторінці відображаються необхідні посилання для маніпулювання даними(рис. 3.14).

Product

All Products

+

Create New

Show 10 entries

Search:

Id	Image	Name	Price	Type	Status	Action
55		Грунтовка	112	Top Product	☒	
54		Цегла керамічна	111	Featured Product	☒	
53		Силікатна цегла	130	New Arrival	☒	
52		Керамзит	215	Top Product	☒	

Showing 1 to 4 of 4 entries

Previous

1

Next

Рис. 3.14. Сторінка “Products”

При додаванні адміністратор повинен заповнити всі обов’язкові поля(рис. 3.15, 3.16, 3.17).

Product

Create Product

Image

Вибрати файл

Файл не вибрано

Name

Category

Select

Sub Category

Select

Child Category

Select

Brand

Select

SKU

Price

Рис. 3.15. Створення товару

Offer Price

Offer Start Date 2023-08-30 Offer End Date 2023-08-30

Stock Quantity

Video Link

Short Description

Long Description

Product Type

Seo Title

Seo Description

Status

Create

Рис. 3.16. Продовження створення товару

Product Type

Select

Seo Title

Seo Description

Status

Active

Create

Рис. 3.17. Закінчення створення товару

Перейшовши на вкладку “Manage categories”, адміністратор має можливість переглянути, видалити та змінити категорії(рис. 3.18).

Category

All Categories

Show 10 entries

Search:

Id	Icon	Name	Status	Action
22		Терпал погорт	☒	Edit Delete
21		Ріжачі шаблони	☒	Edit Delete
20		Світлові шаблони	☒	Edit Delete
15		Groceries	☒	Edit Delete

Showing 1 to 4 of 4 entries

Previous

Next

Рис. 3.18. Сторінка категорій

Створення категорій проходить по такому ж принципу як і створення товарів, але менше етапів створення(рис. 3.19).

Category

Рис. 3.19. Створення категорії

3.3 Тестування роботи програмного комплексу

Для коректної роботи будь-якого додатка необхідно проводити тестування, що може виявити слабкі місця та проблеми у додатку. Для перевірки працездатності виконаємо дії, які повинен виконувати користувач.

1. Реєстрація облікового запису:

передумови: користувач який потрапив на головну сторінку має можливість зареєструватись;

кроки: натиснувши “Зареєструватись”, користувач заповнює форму та відправляє;

очікуваний результат: користувач отримує лист з верифікацією на пошту, після верифікації обліковий запис ініціалізований в системі;

фактичний результат: користувач зареєструвався без верифікації.

2. Користувач заходить в персональний кабінет:

передумови: користувач переходить до акаунту з головної сторінки;

очікуваний результат: користувач перейшов до особистого кабінету і взаємодіє з його елементами;

фактичний результат: користувач потрапив до особистого кабінету.

3. Взаємодія з сутностями (Збережені, Коментарі, Допомога):

передумови: користувач повинен мати змогу для керування даними які йому потрібно;

кроки: перехід на вкладку сутності (Замовлення, Відгуки, Адреса), де користувач може переглянути, додати, змінити та видалити обрані дані;

очікуваний результат: користувач змінює необхідні йому дані й вони зберігаються;

фактичний результат: користувач має можливість взаємодіяти зі всіма елементами персонального кабінету.

4. Взаємодія з елементами головної сторінки:

передумови: користувач, який знаходиться на головній сторінці, повинен мати змогу: переглядати товари, сторінку “Категорії” та мати можливість взаємодіяти з товарами;

передумови: користувач, який знаходиться на головній сторінці, відкриває посилання на товар;

очікуваний результат: користувачу відкривається повна версія товару з можливістю придбати його;

фактичний результат: користувач має змогу взаємодіяти з необхідними елементами.

5. CRUD сутностей (Товари, Користувачі, Категорії) користувачем з роллю адміністратор:

передумови: користувач (адмін), перейшовши до адміністративного кабінету, повинен мати можливість для маніпулювання необхідними даними;

кроки: обравши додавання користувача, адміністратор побачить форму з необхідними для заповнення полями;

очікуваний результат: після відправлення форми, користувач збережений у базі даних;

фактичний результат: після створення користувач збережений у базі даних.

Висновки до третього розділу

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						45

В даному розділі було розглянуто інструкцію щодо розгортання системи на локальній машині, які технології та дії для цього потрібні.

Було детально розглянуто принцип роботи системи. Її компоненти, сторінки та методи. Розглянуто дії для кожної з ролей (гість, користувач, адміністратор). Сам опис являє собою візуальну інструкцію щодо використання додатка.

Після інформації про роботу інтерфейсу, було проведене тестування. Тестування відбувалося з метою виявлення помилок та їх усунення.

ВИСНОВКИ

При розробці випускної кваліфікаційної роботи бакалавра було реалізовано онлайн-систему для продажу будівельних матеріалів за допомогою PHP фреймворку Laravel. При створенні системи були вирішені наступні питання:

1. Вибір веб-орієнтованої архітектури для розробки системи. Були розглянуті аналогічні програмні продукти (БудМарт, САД, Вікант). Був обраний певний стек технологій який буде використовуватись при розробці онлайн-системи.

2. Було розглянуто набір технологічних інструментів для розробки, обрано мову програмування PHP, базу даних MySQL та фреймворк Laravel.

3. Було реалізовано та розглянуто основні методи та функції, які присутні в онлайн-системі. Було представлено алгоритми роботи на сайті та інші можливості. Відповідно до обраної архітектури та поставлених вимог були обрані інструментальні засоби для реалізації проекту.

4. Було детально розглянуто принцип роботи системи, компоненти, сторінки. Розглянуто дії для кожної з ролей (гість, користувач, адміністратор).

Внаслідок виконання кваліфікаційної роботи бакалавра був реалізована онлайн-система, що функціонує та має потенціал в розвитку додаткових методів та варіантів роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. “MVC патерн”, freeCodeCamp, 2023. Доступ за посиланням <https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/>
2. “MVP патерн”, Brander, 2023. Доступ за посиланням <https://brander.ua/technologies/mvp>
3. “MVVM патерн”, TechTarget, 1999-2023. Доступ за посиланням <https://www.techtarget.com/whatis/definition/Model-View-ViewModel>
4. “HTML Documentation”, MDN, 1998-2023. [Електронний ресурс]. Доступ за посиланням <https://developer.mozilla.org/ru/docs/Web/HTML>
5. “CSS Documentation”, mozilla.org, 1998–2023. [Електронний ресурс]. Доступ за посиланням <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>
6. ”JavaScript”, mozilla.org, 1998–2023. [Електронний ресурс]. Доступ за посиланням <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
7. “PHP Documentation”, The PHP Group, 2001-2023. [Електронний ресурс]. Доступ за посиланням <https://www.php.net/docs.php>
8. “IDE PhpStorm”, JetBrains s.r.o., 2000-2023. Доступ за посиланням <https://www.jetbrains.com/phpstorm/>
9. “Installing MySQL”, Oracle, 1995-2023. [Електронний ресурс]. Доступ за посиланням <https://dev.mysql.com/doc/refman/8.0/en/windows-installation.html>
10. “npm install”, npmjs, 2023. [Електронний ресурс]. Доступ за посиланням <https://www.npmjs.com/package/npm>
11. “nodejs install”, OpenJS Foundation, 2023. [Електронний ресурс]. Доступ за посиланням <https://nodejs.org/en/download>
12. “composer install”, Nils Adermann, Jordi Boggiano, 2023. [Електронний ресурс]. Доступ за посиланням <https://getcomposer.org/doc/00-intro.md>
13. “MySQL Documentation”, Oracle, 1995-2023. [Електронний ресурс]. Доступ за посиланням <https://dev.mysql.com/doc/>

14. “Laravel”, Laravel LLC, 2011-2023. [Електронний ресурс]. Доступ за посиланням <https://laravel.com/>
15. “Artisan”, Laravel LLC, 2011-2023. [Електронний ресурс]. Доступ за посиланням <https://laravel.com/docs/10.x/artisan>
16. БудМарт [Електронний ресурс]. Доступ за посиланням <https://budmart.zt.ua>
17. САД [Електронний ресурс]. Доступ за посиланням <https://sad-torg.com.ua>
18. Вікант [Електронний ресурс]. Доступ за посиланням <https://vikant.com.ua/ua>
19. ХАМРР, Apache Friends, 2014-2023. [Електронний ресурс]. Доступ за посиланням <https://www.apachefriends.org/index.html>
20. PSR-1: Basic Coding Standard, PHP Framework Interop Group, 2023, [Електронний ресурс]. Доступ за посиланням <https://www.php-fig.org/psr/psr-1/>

ДОДАТКИ

					ПЗ.КР.Б – 121 – 23 – ПЗ	
						50

Код маршрутів додатка

web.php

```

<?php

use App\Http\Controllers\Backend\AdminController;
use App\Http\Controllers\Backend\VendorController;
use App\Http\Controllers\Frontend\BlogController;
use App\Http\Controllers\Frontend\CartController;
use App\Http\Controllers\Frontend\FlashSaleController;
use App\Http\Controllers\Frontend\FrontendProductController;
use App\Http\Controllers\Frontend\HomeController;
use App\Http\Controllers\Frontend\UserAddressController;
use App\Http\Controllers\Frontend\UserDashboardController;
use App\Http\Controllers\Frontend\UserProfileController;
use App\Http\Controllers\Frontend\CheckOutController;
use App\Http\Controllers\Frontend\NewsletterController;
use App\Http\Controllers\Frontend\PageController;
use App\Http\Controllers\Frontend\PaymentController;
use App\Http\Controllers\Frontend\ProductTrackController;
use App\Http\Controllers\Frontend\ReviewController;
use App\Http\Controllers\Frontend\UserOrderController;
use App\Http\Controllers\Frontend\UserVendorRegeustController;
use App\Http\Controllers\Frontend\WishlistController;
use App\Http\Controllers\ProfileController;
use Illuminate\Support\Facades\Route;

/*
|-----
| Web Routes
|-----
|
| Here is where you can register web routes for your application.
These
| routes are loaded by the RouteServiceProvider and all of them
will
| be assigned to the "web" middleware group. Make something great!
|
*/

Route::get('/', [HomeController::class, 'index'])->name('home');

Route::middleware('auth')->group(function () {
    Route::get('/profile', [ProfileController::class, 'edit'])->
>name('profile.edit');
    Route::patch('/profile', [ProfileController::class,

```

```

'update']->name('profile.update');
    Route::delete('/profile', [ProfileController::class,
'destroy']->name('profile.destroy');
});

require __DIR__.'/auth.php';

Route::get('admin/login', [AdminController::class, 'login']->name('admin.login');

Route::get('flash-sale', [FlashSaleController::class, 'index']->name('flash-sale');

/** Product route */
Route::get('products', [FrontendProductController::class,
'productsIndex']->name('products.index');
Route::get('product-detail/{slug}',
[FrontendProductController::class, 'showProduct']->name('product-
detail');
Route::get('change-product-list-view',
[FrontendProductController::class, 'chageListView']->name('change-product-list-view');

/** Cart routes */
Route::post('add-to-cart', [CartController::class, 'addToCart']->name('add-to-cart');
Route::get('cart-details', [CartController::class,
'cartDetails']->name('cart-details');
Route::post('cart/update-quantity', [CartController::class,
'updateProductQty']->name('cart.update-quantity');
Route::get('clear-cart', [CartController::class, 'clearCart']->name('clear.cart');
Route::get('cart/remove-product/{rowId}', [CartController::class,
'removeProduct']->name('cart.remove-product');
Route::get('cart-count', [CartController::class, 'getCartCount']->name('cart-count');
Route::get('cart-products', [CartController::class,
'getCartProducts']->name('cart-products');
Route::post('cart/remove-sidebar-product', [CartController::class,
'removeSidebarProduct']->name('cart.remove-sidebar-product');
Route::get('cart/sidebar-product-total', [CartController::class,
'cartTotal']->name('cart.sidebar-product-total');

Route::get('apply-coupon', [CartController::class,
'applyCoupon']->name('apply-coupon');
Route::get('coupon-calculation', [CartController::class,
'couponCalculation']->name('coupon-calculation');

/** Newsletter routes */

Route::post('newsletter-request', [NewsletterController::class,
'newsLetterRequset']->name('newsletter-request');

```

```

Route::get('newsletter-verify/{token}',
[NewsletterController::class, 'newsLetterEmailVarify'])-
>name('newsletter-verify');

/** vendor page routes */
Route::get('vendor', [HomeController::class, 'vendorPage'])-
>name('vendor.index');
Route::get('vendor-product/{id}', [HomeController::class,
'vendorProductsPage'])->name('vendor.products');

/** about page route */
Route::get('about', [PageController::class, 'about'])-
>name('about');
/** terms and conditions page route */
Route::get('terms-and-conditions', [PageController::class,
'termsAndCondition'])->name('terms-and-conditions');
/** contact route */
Route::get('contact', [PageController::class, 'contact'])-
>name('contact');
Route::post('contact', [PageController::class,
'handleContactForm'])->name('handle-contact-form');

/** Product track route */
Route::get('product-traking', [ProductTrackController::class,
'index'])->name('product-traking.index');

/** blog routes */
Route::get('blog-details/{slug}', [BlogController::class,
'blogDetails'])->name('blog-details');
Route::get('blog', [BlogController::class, 'blog'])->name('blog');

Route::group(['middleware' => ['auth', 'verified'], 'prefix' =>
'user', 'as' => 'user.'], function() {
    Route::get('dashboard', [UserDashboardController::class,
'index'])->name('dashboard');
    Route::get('profile', [UserProfileController::class,
'index'])->name('profile'); // user.profile
    Route::put('profile', [UserProfileController::class,
'updateProfile'])->name('profile.update'); // user.profile.update
    Route::post('profile', [UserProfileController::class,
'updatePassword'])->name('profile.update.password');

    /** User Address Route */
    Route::resource('address', UserAddressController::class);
    /** Order Routes */
    Route::get('orders', [UserOrderController::class, 'index'])-

```

```

>name('orders.index');
    Route::get('orders/show/{id}', [UserController::class,
'show'])->name('orders.show');

    /** Wishlist routes */
    Route::get('wishlist', [WishlistController::class, 'index'])-
>name('wishlist.index');
    Route::get('wishlist/add-product', [WishlistController::class,
'addToWishlist'])->name('wishlist.store');
    Route::get('wishlist/remove-product/{id}',
[WishlistController::class, 'destory'])->name('wishlist.destory');

    Route::get('reviews', [ReviewController::class, 'index'])-
>name('review.index');

    /** Vendor request route */
    Route::get('vendor-request',
[UserVendorRegeustController::class, 'index'])->name('vendor-
request.index');
    Route::post('vendor-request',
[UserVendorRegeustController::class, 'create'])->name('vendor-
request.create');

    /** product review routes */
    Route::post('review', [ReviewController::class, 'create'])-
>name('review.create');

    /** blog comment routes */
    Route::post('blog-comment', [BlogController::class,
'comment'])->name('blog-comment');

    /** Checkout routes */
    Route::get('checkout', [CheckOutController::class, 'index'])-
>name('checkout');
    Route::post('checkout/address-create',
[CheckOutController::class, 'createAddress'])-
>name('checkout.address.create');
    Route::post('checkout/form-submit',
[CheckOutController::class, 'checkOutFormSubmit'])-
>name('checkout.form-submit');

    /** Payment Routes */
    Route::get('payment', [PaymentController::class, 'index'])-
>name('payment');
    Route::get('payment-success', [PaymentController::class,
'paymentSuccess'])->name('payment.success');

    /** Paypal routes */
    Route::get('paypal/payment', [PaymentController::class,
'payWithPaypal'])->name('paypal.payment');
    Route::get('paypal/success', [PaymentController::class,

```

```

'paypalSuccess']->name('paypal.success');
    Route::get('paypal/cancel', [PaymentController::class,
'paypalCancel'])->name('paypal.cancel');

    /** Stripe routes */
    Route::post('stripe/payment', [PaymentController::class,
'payWithStripe'])->name('stripe.payment');

    /** Razorpay routes */
    Route::post('razorpay/payment', [PaymentController::class,
'payWithRazorPay'])->name('razorpay.payment');

    /** COD routes */
    Route::get('cod/payment', [PaymentController::class,
'payWithCod'])->name('cod.payment');
});

```

admin.php

```

<?php

use App\Http\Controllers\Backend\AboutController;
use App\Http\Controllers\Backend\AboutController;
use App\Http\Controllers\Backend\AdminController;
use App\Http\Controllers\Backend\AdminListController;
use App\Http\Controllers\Backend\AdminReviewController;
use App\Http\Controllers\Backend\AdminVendorProfileController;
use App\Http\Controllers\Backend\AdvertisementController;
use App\Http\Controllers\Backend\BlogCategoryController;
use App\Http\Controllers\Backend\BlogCommentController;
use App\Http\Controllers\Backend\BlogController;
use App\Http\Controllers\Backend\BrandController;
use App\Http\Controllers\Backend\CategoryController;
use App\Http\Controllers\Backend\ChildCategoryController;
use App\Http\Controllers\Backend\CodSettingController;
use App\Http\Controllers\Backend\CouponController;
use App\Http\Controllers\Backend\CustomerListController;
use App\Http\Controllers\Backend\FlashSaleController;
use App\Http\Controllers\Backend\FooterGridThreeController;
use App\Http\Controllers\Backend\FooterGridTwoController;
use App\Http\Controllers\Backend\FooterInfoController;
use App\Http\Controllers\Backend\FooterSocialController;
use App\Http\Controllers\Backend\HomePageSettingController;
use App\Http\Controllers\Backend\ManageUserController;
use App\Http\Controllers\Backend\OrderController;
use App\Http\Controllers\Backend\PaymentSettingController;
use App\Http\Controllers\Backend\PaypalSettingController;
use App\Http\Controllers\Backend\ProductController;
use App\Http\Controllers\Backend\ProductImageGalleryController;
use App\Http\Controllers\Backend\ProductVariantController;
use App\Http\Controllers\Backend\ProductVariantItemController;
use App\Http\Controllers\Backend\ProfileController;
use App\Http\Controllers\Backend\RazorpaySettingController;

```

```

use App\Http\Controllers\Backend\SellerProductController;
use App\Http\Controllers\Backend\SettingController;
use App\Http\Controllers\Backend\ShippingRuleController;
use App\Http\Controllers\Backend\SliderController;
use App\Http\Controllers\Backend\StripeSettingController;
use App\Http\Controllers\Backend\SubCategoryController;
use App\Http\Controllers\Backend\SubscribersController;
use App\Http\Controllers\Backend\TermsAndConditionController;
use App\Http\Controllers\Backend\TransactionController;
use App\Http\Controllers\Backend\VendorConditionController;
use App\Http\Controllers\Backend\VendorListController;
use App\Http\Controllers\Backend\VendorRequestController;
use App\Models\BlogCategory;
use App\Models\FooterInfo;
use App\Models\VendorCondition;
use Illuminate\Support\Facades\Route;

/** Admin Routes */

Route::get('dashboard', [AdminController::class, 'dashboard'])->
>name('dashbaord');

/** Profile Routes */
Route::get('profile', [ProfileController::class, 'index'])->
>name('profile');
Route::post('profile/update', [ProfileController::class,
'updateProfile'])->>name('profile.update');
Route::post('profile/update/password', [ProfileController::class,
'updatePassword'])->>name('password.update');

/** Slider Route */
Route::resource('slider', SliderController::class);

/** Category Route */
Route::put('change-status', [CategoryController::class,
'changeStatus'])->>name('category.change-status');
Route::resource('category', CategoryController::class);
/** Sub Category Route */
Route::put('subcategory/change-status',
[SubCategoryController::class, 'changeStatus'])->>name('sub-
category.change-status');
Route::resource('sub-category', SubCategoryController::class);
/** Child Category Route */
Route::put('child-category/change-status',
[ChildCategoryController::class, 'changeStatus'])->>name('child-
category.change-status');
Route::get('get-subcategories', [ChildCategoryController::class,
'getSubCategories'])->>name('get-subcategories');
Route::resource('child-category', ChildCategoryController::class);

/** Brand routes */
Route::put('brand/change-status', [BrandController::class,

```

```

'changeStatus']->name('brand.change-status');
Route::resource('brand', BrandController::class);

/** Vendor Profile routes */
Route::resource('vendor-profile',
AdminVendorProfileController::class);

/** Products routes */
Route::get('product/get-subcategories', [ProductController::class,
'getSubCategories']->name('product.get-subcategories');
Route::get('product/get-child-categories',
[ProductController::class, 'getChildCategories']->
>name('product.get-child-categories');
Route::put('product/change-status', [ProductController::class,
'changeStatus']->name('product.change-status');
Route::resource('products', ProductController::class);

/** Products image gallery route */
Route::resource('products-image-gallery',
ProductImageGalleryController::class);

/** Products variant route */
Route::put('products-variant/change-status',
[ProductVariantController::class, 'changeStatus']->
>name('products-variant.change-status');
Route::resource('products-variant',
ProductVariantController::class);

/** Products variant item route */
Route::get('products-variant-item/{productId}/{variantId}',
[ProductVariantItemController::class, 'index']->name('products-
variant-item.index');

Route::get('products-variant-item/create/{productId}/{variantId}',
[ProductVariantItemController::class, 'create']->name('products-
variant-item.create');
Route::post('products-variant-item',
[ProductVariantItemController::class, 'store']->name('products-
variant-item.store');

Route::get('products-variant-item-edit/{variantItemId}',
[ProductVariantItemController::class, 'edit']->name('products-
variant-item.edit');

Route::put('products-variant-item-update/{variantItemId}',
[ProductVariantItemController::class, 'update']->name('products-
variant-item.update');

Route::delete('products-variant-item/{variantItemId}',
[ProductVariantItemController::class, 'destroy']->name('products-
variant-item.destroy');

Route::put('products-variant-item-status',

```

```

[ProductVariantItemController::class, 'chageStatus'])-
>name('products-variant-item.chages-status');

/** reviews routes */
Route::get('reviews', [AdminReviewController::class, 'index'])-
>name('reviews.index');
Route::put('reviews/change-status', [AdminReviewController::class,
'changeStatus'])->name('reviews.change-status');

/** Seller product routes */
Route::get('seller-products', [SellerProductController::class,
'index'])->name('seller-products.index');
Route::get('seller-pending-products',
[SellerProductController::class, 'pendingProducts'])-
>name('seller-pending-products.index');
Route::put('change-approve-status',
[SellerProductController::class, 'changeApproveStatus'])-
>name('change-approve-status');

/** Flash Sale Routes */
Route::get('flash-sale', [FlashSaleController::class, 'index'])-
>name('flash-sale.index');
Route::put('flash-sale', [FlashSaleController::class, 'update'])-
>name('flash-sale.update');
Route::post('flash-sale/add-product', [FlashSaleController::class,
'addProduct'])->name('flash-sale.add-product');
Route::put('flash-sale/show-at-home/status-change',
[FlashSaleController::class, 'chageShowAtHomeStatus'])-
>name('flash-sale.show-at-home.change-status');
Route::put('flash-sale-status', [FlashSaleController::class,
'changeStatus'])->name('flash-sale-status');
Route::delete('flash-sale/{id}', [FlashSaleController::class,
'destory'])->name('flash-sale.destory');

/** Coupon Routes */
Route::put('coupons/change-status', [CouponController::class,
'changeStatus'])->name('coupons.change-status');
Route::resource('coupons', CouponController::class);

/** Coupon Routes */
Route::put('shipping-rule/change-status',
[ShippingRuleController::class, 'changeStatus'])->name('shipping-
rule.change-status');
Route::resource('shipping-rule', ShippingRuleController::class);

/** Order routes */
Route::get('payment-status', [OrderController::class,
'changePaymentStatus'])->name('payment.status');
Route::get('order-status', [OrderController::class,
'changeOrderStatus'])->name('order.status');

Route::get('pending-orders', [OrderController::class,
'pendingOrders'])->name('pending-orders');

```

```

Route::get('processed-orders', [OrderController::class,
'processedOrders'])->name('processed-orders');
Route::get('dropped-off-orders', [OrderController::class,
'droppedOfOrders'])->name('dropped-off-orders');

Route::get('shipped-orders', [OrderController::class,
'shippedOrders'])->name('shipped-orders');
Route::get('out-for-delivery-orders', [OrderController::class,
'outForDeliveryOrders'])->name('out-for-delivery-orders');
Route::get('delivered-orders', [OrderController::class,
'deliveredOrders'])->name('delivered-orders');
Route::get('canceled-orders', [OrderController::class,
'canceledOrders'])->name('canceled-orders');
Route::resource('order', OrderController::class);

/** Order Transaction route */
Route::get('transaction', [TransactionController::class,
'index'])->name('transaction');

/** settings routes */
Route::get('settings', [SettingController::class, 'index'])->
>name('settings.index');
Route::put('generale-setting-update', [SettingController::class,
'generalSettingUpdate'])->name('generale-setting-update');
Route::put('email-setting-update', [SettingController::class,
'emailConfigSettingUpdate'])->name('email-setting-update');
Route::put('logo-setting-update', [SettingController::class,
'logoSettingUpdate'])->name('logo-setting-update');

/** home page setting route */
Route::get('home-page-setting', [HomePageSettingController::class,
'index'])->name('home-page-setting');
Route::put('popular-category-section',
[HomePageSettingController::class,
'updatePopularCategorySection'])->name('popular-category-
section');

Route::put('product-slider-section-one',
[HomePageSettingController::class,
'updateProductSliderSectionOn'])->name('product-slider-section-
one');
Route::put('product-slider-section-two',
[HomePageSettingController::class,
'updateProductSliderSectionTwo'])->name('product-slider-section-
two');
Route::put('product-slider-section-three',
[HomePageSettingController::class,
'updateProductSliderSectionThree'])->name('product-slider-section-
three');

```

```

/** Blog routes */
Route::put('blog-category/status-change',
[BlogCategoryController::class, 'changeStatus'])->name('blog-
category.status-change');
Route::resource('blog-category', BlogCategoryController::class);

Route::put('blog/status-change', [BlogController::class,
'changeStatus'])->name('blog.status-change');
Route::resource('blog', BlogController::class);
Route::get('blog-comments', [BlogCommentController::class,
'index'])->name('blog-comments.index');
Route::delete('blog-comments/{id}/destory',
[BlogCommentController::class, 'destory'])->name('blog-
comments.destory');

/** Subscribers route */
Route::get('subscribers', [SubscribersController::class,
'index'])->name('subscribers.index');
Route::delete('subscribers/{id}', [SubscribersController::class,
'destory'])->name('subscribers.destory');
Route::post('subscribers-send-mail',
[SubscribersController::class, 'sendMail'])->name('subscribers-
send-mail');

/** Advertisement Routes */
Route::get('advertisement', [AdvertisementController::class,
'index'])->name('advertisement.index');
Route::put('advertisement/homepage-banner-seccion-one',
[AdvertisementController::class, 'homepageBannerSecionOne'])->
name('homepage-banner-seccion-one');
Route::put('advertisement/homepage-banner-seccion-two',
[AdvertisementController::class, 'homepageBannerSecionTwo'])->
name('homepage-banner-seccion-two');
Route::put('advertisement/homepage-banner-seccion-three',
[AdvertisementController::class, 'homepageBannerSecionThree'])->
name('homepage-banner-seccion-three');
Route::put('advertisement/homepage-banner-seccion-four',
[AdvertisementController::class, 'homepageBannerSecionFour'])->
name('homepage-banner-seccion-four');
Route::put('advertisement/productpage-banner',
[AdvertisementController::class, 'productPageBanner'])->
name('productpage-banner');
Route::put('advertisement/cartpage-banner',
[AdvertisementController::class, 'cartPageBanner'])->
name('cartpage-banner');

/** Vendor request routes */
Route::get('vendor-requests', [VendorRequestController::class,
'index'])->name('vendor-requests.index');
Route::get('vendor-requests/{id}/show',
[VendorRequestController::class, 'show'])->name('vendor-
requests.show');

```

```

Route::put('vendor-requests/{id}/change-status',
[VendorRequestController::class, 'changeStatus'])->name('vendor-
requests.change-status');

/** coustomer list routes */
Route::get('customer', [CustomerListController::class, 'index'])->
>name('customer.index');
Route::put('customer/status-change',
[CustomerListController::class, 'changeStatus'])->
>name('customer.status-change');
/** coustomer list routes */
Route::get('admin-list', [AdminListController::class, 'index'])->
>name('admin-list.index');
Route::put('admin-list/status-change',
[AdminListController::class, 'changeStatus'])->name('admin-
list.status-change');
Route::delete('admin-list/{id}', [AdminListController::class,
'destory'])->name('admin-list.destory');

/** manage user routes */
Route::get('manage-user', [ManageUserController::class, 'index'])->
>name('manage-user.index');
Route::post('manage-user', [ManageUserController::class,
'create'])->name('manage-user.create');

Route::get('vendor-list', [VendorListController::class, 'index'])->
>name('vendor-list.index');
Route::put('vendor-list/status-change',
[VendorListController::class, 'changeStatus'])->name('vendor-
list.status-change');

Route::get('vendor-condition', [VendorConditionController::class,
'index'])->name('vendor-condition.index');
Route::put('vendor-condition/update',
[VendorConditionController::class, 'update'])->name('vendor-
condition.update');

/** about routes */
Route::get('about', [AboutController::class, 'index'])->
>name('about.index');
Route::put('about/update', [AboutController::class, 'update'])->
>name('about.update');
/** terms and conditons routes */
Route::get('terms-and-conditions',
[TermsAndConditionController::class, 'index'])->name('terms-and-
conditions.index');
Route::put('terms-and-conditions/update',
[TermsAndConditionController::class, 'update'])->name('terms-and-
conditions.update');

/** footer routes */

```

```

Route::resource('footer-info', FooterInfoController::class);
Route::put('footer-socials/change-status',
[FooterSocialController::class, 'changeStatus'])->name('footer-
socials.change-status');
Route::resource('footer-socials', FooterSocialController::class);

Route::put('footer-grid-two/change-status',
[FooterGridTwoController::class, 'changeStatus'])->name('footer-
grid-two.change-status');
Route::put('footer-grid-two/change-title',
[FooterGridTwoController::class, 'changeTitle'])->name('footer-
grid-two.change-title');
Route::resource('footer-grid-two',
FooterGridTwoController::class);

Route::put('footer-grid-three/change-status',
[FooterGridThreeController::class, 'changeStatus'])->name('footer-
grid-three.change-status');
Route::put('footer-grid-three/change-title',
[FooterGridThreeController::class, 'changeTitle'])->name('footer-
grid-three.change-title');
Route::resource('footer-grid-three',
FooterGridThreeController::class);

/** Payment settings routes */
Route::get('payment-settings', [PaymentSettingController::class,
'index'])->name('payment-settings.index');
Route::resource('paypal-setting', PaypalSettingController::class);
Route::put('stripe-setting/{id}', [StripeSettingController::class,
'update'])->name('stripe-setting.update');
Route::put('razorpay-setting/{id}',
[RazorpaySettingController::class, 'update'])->name('razorpay-
setting.update');
Route::put('cod-setting/{id}', [CodSettingController::class,
'update'])->name('cod-setting.update');

```

auth.php

```

<?php

use App\Http\Controllers\Auth\AuthenticatedSessionController;
use App\Http\Controllers\Auth\ConfirmablePasswordController;
use
App\Http\Controllers\Auth>EmailVerificationNotificationController;
use App\Http\Controllers\Auth>EmailVerificationPromptController;
use App\Http\Controllers\Auth\NewPasswordController;
use App\Http\Controllers\Auth>PasswordController;
use App\Http\Controllers\Auth>PasswordResetLinkController;
use App\Http\Controllers\Auth\RegisteredUserController;
use App\Http\Controllers\Auth\VerifyEmailController;

```

```

use Illuminate\Support\Facades\Route;

Route::middleware('guest')->group(function () {
    // Route::get('register', [RegisteredUserController::class,
    'create'])
    //             ->name('register');

    Route::post('register', [RegisteredUserController::class,
    'store'])->name('register');

    Route::get('login', [AuthenticatedSessionController::class,
    'create'])
        ->name('login');

    Route::post('login', [AuthenticatedSessionController::class,
    'store']);

    Route::get('forgot-password',
    [PasswordResetLinkController::class, 'create'])
        ->name('password.request');

    Route::post('forgot-password',
    [PasswordResetLinkController::class, 'store'])
        ->name('password.email');

    Route::get('reset-password/{token}',
    [NewPasswordController::class, 'create'])
        ->name('password.reset');

    Route::post('reset-password', [NewPasswordController::class,
    'store'])
        ->name('password.store');
});

Route::middleware('auth')->group(function () {
    Route::get('verify-email',
    EmailVerificationPromptController::class)
        ->name('verification.notice');

    Route::get('verify-email/{id}/{hash}',
    VerifyEmailController::class)
        ->middleware(['signed', 'throttle:6,1'])
        ->name('verification.verify');

    Route::post('email/verification-notification',
    [EmailVerificationNotificationController::class, 'store'])
        ->middleware('throttle:6,1')
        ->name('verification.send');

    Route::get('confirm-password',
    [ConfirmablePasswordController::class, 'show'])
        ->name('password.confirm');
});

```

```

Route::post('confirm-password',
[ConfirmablePasswordController::class, 'store']);

Route::put('password', [PasswordController::class, 'update'])->name('password.update');

Route::post('logout', [AuthenticatedSessionController::class, 'destroy'])
->name('logout');
});

```

2023_03_21_093706_create_products_table.php

```

<?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('name');
            $table->string('slug');
            $table->text('thumb_image');
            $table->integer('vendor_id');
            $table->integer('category_id');
            $table->integer('sub_category_id')->nullable();
            $table->integer('child_category_id')->nullable();
            $table->integer('brand_id');
            $table->integer('qty');
            $table->text('short_description');
            $table->text('long_description');
            $table->text('video_link')->nullable();
            $table->string('sku')->nullable();
            $table->double('price');
            $table->double('offer_price')->nullable();
            $table->date('offer_start_date')->nullable();
            $table->date('offer_end_date')->nullable();
            $table->string('product_type')->nullable();
            $table->boolean('status');
            $table->integer('is_approved')->default(0);
            $table->string('seo_title')->nullable();
            $table->text('seo_description')->nullable();

            $table->timestamps();
        });
    }
};

```

```

    }

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

```