

ЗМІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА
ПРИРОДОКОРИСТУВАННЯ

Навчально-науковий інститут кібернетики інформаційних технологій та
інженерії

Кафедра комп'ютерних наук та прикладної математики

«До захисту допущений»

Завідувач кафедри комп'ютерних наук
та прикладної математики

_____ Турбал Ю.В.

«_____» _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

**«Розробка веб-додатку для організації домашнього кінотеатру на основі
платформи Woocommerce.»**

Виконав: Власюк Андрій Вікторович

(прізвище, ім'я, по батькові)

(підпис)

група КНз-51

Керівник: професор Турбал Ю.В.

(науковий ступінь, вчене звання, посада, прізвище та ініціали)

(підпис)

Національний університет водного господарства та природокористування

Навчально-науковий інститут кібернетики інформаційних технологій та Інженерії

Кафедра комп'ютерних наук та прикладної математики

Рівень вищої освіти бакалавр

Галузь знань 12 «Інформаційні технології»

Спеціальність 122 «Комп'ютерні науки»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри
комп'ютерних наук та
прикладної математики
д.т.н., професор Турбал Ю.В.

«____» _____ 2024 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Власюку Андрію Вікторовичу

(прізвище, ім'я, по батькові)

1. Тема роботи Розробка веб-додатку для організації домашнього кінотеатру на основі платформи Woocommerce.

керівник роботи професор Турбал Юрій Васильович

затверджені наказом вищого навчального закладу від «22» квітня 2024 року

С №-525

2. Термін здачі студентом закінченої роботи 29.05.2024

3. Вихідні дані до роботи: документація замовника

4. Зміст розрахунково-пояснювальної записки Розділ 1. Огляд предметної області. Розділ 2. Інструменти розробки. Розділ 3. Етапи розробки. Розділ 4.

Особливості реалізації UI

5. Перелік графічного матеріалу мультимедійна презентація

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|-----------------|---|----------------|------------------|
| | | завдання видав | завдання прийняв |
| <i>Розділ 1</i> | <i>Турбал Ю.В., професор</i> | | |
| <i>Розділ 2</i> | <i>Турбал Ю.В., професор</i> | | |
| <i>Розділ 3</i> | <i>Турбал Ю.В., професор</i> | | |
| <i>Розділ 4</i> | <i>Турбал Ю.В., професор</i> | | |

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|----------|--|-------------------------------|----------|
| <i>1</i> | <i>Опрацювання літератури та інтернет-джерел на задану тему.</i> | | |
| <i>2</i> | <i>Узгодження із замовником.</i> | | |
| <i>3</i> | <i>Опрацювання завдання та виявлення ключових аспектів</i> | | |
| <i>4</i> | <i>Вибір технологій для реалізації серверної частини</i> | | |
| <i>5</i> | <i>Вибір технологій для реалізації клієнтської частини</i> | | |
| <i>6</i> | <i>Аналіз та вивчення документації для реалізації</i> | | |
| <i>7</i> | <i>Програмна реалізація.</i> | | |
| <i>8</i> | <i>Тестування та виправлення помилок.</i> | | |
| <i>9</i> | <i>Оформлення теоретичної частини</i> | | |

Студент _____ (Власюк А.В.)

Керівник кваліфікаційної роботи _____ (Турбал Ю.В.)

ЗМІСТ

| | |
|--|----|
| РЕФЕРАТ | 6 |
| АНОТАЦІЯ | 7 |
| ВСТУП | 8 |
| РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ | 10 |
| 1.1. Загальні відомості про веб-додатки для організації домашнього кінотеатру | 10 |
| 1.2. Ключові поняття | 11 |
| 1.3. Формулювання задачі | 12 |
| РОЗДІЛ 2 ІНСТРУМЕНТИ РОЗРОБКИ | 13 |
| 2.1. Вступ | 13 |
| 2.2. Серверна частина | 13 |
| 2.3. Клієнтська частина | 15 |
| РОЗДІЛ 3 ЕТАПИ РОЗРОБКИ | 17 |
| 3.1. Налаштування серверної частини | 17 |
| 3.2. Розробка клієнтської частини | 20 |
| 3.2.1. Встановлення CMS Wordpress на локальний сервер | 20 |
| 3.2.2. Створення власної теми для веб додатку | 22 |
| 3.2.3. Встановлення Woocommerce | 26 |
| 3.2.4. Створення сторінок, меню, віджетів та продуктів для веб-додатку | 26 |
| 3.2.5. Створення різних функцій та робота з файлами теми | 30 |
| 3.2.6. Зміна шаблонів Woocommerce | 35 |
| РОЗДІЛ 4 ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ UI | 39 |
| 4.1. Головна сторінка | 39 |
| 4.2. Сторінка каталогу | 40 |

| | |
|---|-----------|
| | 1 |
| 4.3. Сторінка категорії товарів | 41 |
| 4.4. Сторінка товару..... | 42 |
| 4.5. Сторінка корзини | 43 |
| 4.6. Сторінка оплати | 44 |
| 4.7. Сторінка акаунта користувача | 45 |
| 4.8. Додаткові сторінки веб-додатку..... | 46 |
| ВИСНОВОК | 48 |
| СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ..... | 49 |

РЕФЕРАТ

Кваліфікаційна робота: 45 листів, 50 малюнків

Мета: Розробка веб-додатку для організації домашнього кінотеатру на основі платформи Woocommerce

Завдання: Розробка веб-додатку, що спрощує процес вибору та покупки обладнання для домашнього кінотеатру, а також оптимізує обробку замовлень з боку команди підтримки.

Для досягнення поставленої мети, потрібно виконати наступні пункти:

- Провести аналіз існуючих веб-додатків у сфері домашніх кінотеатрів для виявлення їх недоліків.
- Визначити та аналізувати типові проблеми взаємодії між клієнтами та командою підтримки.
- Дослідити основні труднощі, з якими стикаються співробітники під час спілкування з клієнтами.
- Розробити детальний план створення додатку, включаючи вибір технологій та визначення ключового функціоналу.
- Програмування необхідного функціоналу.
- Створення зрозумілого та привабливого інтерфейсу для користувачів.
- Розробка інтерфейсу з акцентом на ефективність навчання персоналу.
- Запуск, тестування та підтримка веб-додатку.

Результати: Розробка веб-додатку, який забезпечує інтуїтивно зрозумілий, зручний для користувача інтерфейс, а також ефективні інструменти для роботи персоналу, спрямовані на оптимізацію процесу вибору та покупки обладнання для домашнього кінотеатру.

Ключові слова: веб-додаток, серверна та клієнтська частина, інтерфейс користувача, користувач, база даних, команда підтримки.

АНОТАЦІЯ

У цій кваліфікаційній роботі представлено розробку веб-додатку для організації домашнього кінотеатру. Основна мета полягає у створенні зручного, ефективного та візуально привабливого веб-додатку, який задовольняє потреби користувачів у виборі та покупці обладнання для домашнього кінотеатру, а також забезпечує простоту та зручність управління замовленнями для персоналу.

Проект реалізовано на платформі WordPress з використанням WooCommerce, що дозволяє ефективно інтегрувати функціонал електронної комерції та управління контентом. Додаткові плагіни та інструменти були використані для розширення можливостей веб-додатку, зокрема, для покращення інтерфейсу користувача та оптимізації процесу взаємодії між клієнтами та командою підтримки.

Веб-додаток надає користувачам інтуїтивно зрозумілий інтерфейс, який дозволяє легко вибирати та купувати обладнання для домашнього кінотеатру, відстежувати статус замовлень, а також здійснювати комунікацію з персоналом. Для співробітників розроблено зручні інструменти для ефективної роботи з замовленнями та клієнтами.

Результатом проекту став веб-додаток, який не тільки відповідає сучасним вимогам до естетики та функціональності, але й вирішує ряд ключових проблем у сфері організації домашнього кінотеатру, включаючи:

- Комунікація між працівником та клієнтом
- Покращення комунікації між персоналом та клієнтами.
- Підвищення якості обслуговування.
- Оптимізація часу, витраченого клієнтами на вибір та покупку обладнання.

ВСТУП

Сучасний світ, що стрімко розвивається в напрямку інформаційних технологій, суттєво впливає на всі аспекти нашого життя, змінюючи звичні підходи до розваг, освіти та споживання контенту. Значні зміни відбуваються і в сфері домашніх розваг, де з'являється все більше можливостей для створення комфортного та функціонального простору. Одним із важливих елементів сучасного домашнього простору стає домашній кінотеатр, який дозволяє зануритися в атмосферу кіно без необхідності виходити з дому.

Однак, процес вибору та покупки обладнання для домашнього кінотеатру може стати справжнім викликом. Клієнти стикаються з різноманітними проблемами: від недостатньої комунікації з продавцями до складнощів у виборі обладнання, що відповідає їхнім потребам та бюджету. Це вимагає не лише часу, але й глибокого розуміння технічних характеристик продуктів. Додатково, враховуючи різноманітність обставин, в яких можуть перебувати люди, зокрема обмеження через здоров'я або логістичні складнощі, доступність та зручність покупки в інтернеті стають ключовими.

Ось основні проблеми з якими стикаються клієнти:

- **Недостатня комунікація** – Клієнти часто стикаються з викликами у спілкуванні з продавцями, що може призвести до непорозумінь і помилок у виборі обладнання.
- **Складність вибору** – Різноманіття технічних характеристик та моделей обладнання може бути величезним, що ускладнює процес вибору для недосвідчених користувачів.
- **Логістичні складнощі** – Для деяких людей, особливо проживаючих у віддалених регіонах або маючих обмеження через здоров'я, відвідування фізичних магазинів може бути неможливим.

У цьому контексті, розробка веб-додатку для організації домашнього кінотеатру відіграє вирішальну роль. Такий веб-додаток не тільки спрощує процес вибору та покупки, але й надає користувачам інструменти для

ефективного управління своїми замовленнями, комунікації з консультантами та отримання персоналізованих рекомендацій. Це дозволяє кожному користувачеві створити оптимальне рішення для свого домашнього кінотеатру, враховуючи особисті потреби та умови.

Актуальність розробки такого веб-додатку обумовлена не лише потребами ринку, але й загальною тенденцією до цифровізації послуг та покращення якості обслуговування. Веб-додаток для домашнього кінотеатру відповідає на виклики сучасності, пропонуючи рішення, яке робить процес вибору та покупки не лише простішим і зручнішим, але й більш інформативним та приємним для кожного користувача.

Таким чином, ця дипломна робота спрямована на розробку веб-додатку, який не тільки вирішує практичні завдання, пов'язані з організацією домашнього кінотеатру, але й вносить вагомий вклад у покращення якості життя людей, роблячи доступ до високоякісних розваг більш вільним та зручним.

РОЗДІЛ 1

ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальні відомості про веб-додатки для організації домашнього кінотеатру

Веб-додатки для організації домашнього кінотеатру відіграють важливу роль у забезпеченні зручності та доступності обладнання для цього захоплюючого хобі. У порівнянні з традиційними фізичними магазинами, веб-платформи надають більшу гнучкість та можливість вибору для покупців, а також сприяють зручності та ефективності процесу покупок.

Інтернет-магазин, спрямований на організацію домашнього кінотеатру на основі платформи WooCommerce, створює унікальну можливість для покупців придбати необхідне обладнання для створення власного кінотеатру у зручний для них спосіб. Незалежно від місця та часу, покупець може швидко та зручно здійснити покупку за допомогою всього кількох кліків миші або дотиків екрану.

Основні переваги веб-додатків для організації домашнього кінотеатру включають:

- **Широкий вибір продуктів** - Веб-магазин надає доступ до широкого асортименту обладнання для домашнього кінотеатру, включаючи проектори, аудіосистеми, мультимедійні пристрої та аксесуари, що дозволяє покупцям з легкістю знайти необхідні товари.
- **Зручність та доступність** - Покупці можуть здійснювати покупки з будь-якого місця, де є доступ до Інтернету, що забезпечує зручність та доступність покупок навіть у зайнятий графік життя.
- **Персоналізований досвід** - Система може надати персоналізовані рекомендації щодо обладнання на основі попередніх покупок або вибору покупця, що полегшує процес вибору та забезпечує задоволення від покупок.

Таким чином, інтернет-магазин для організації домашнього кінотеатру створює унікальну можливість для покупців отримати необхідне обладнання зручно та ефективно, що сприяє подальшому розвитку цього захоплюючого хобі.

1.2. Ключові поняття

У контексті розробки веб-додатку для організації домашнього кінотеатру на платформі WooCommerce, де клієнти можуть придбати обладнання та аксесуари, існують деякі ключові поняття, які необхідно визначити:

Клієнт домашнього кінотеатру - Особа або організація, яка використовує веб-додаток для придбання обладнання та аксесуарів для організації домашнього кінотеатру. Це можуть бути ентузіасти кіно, власники будинків або приватні компанії, які прагнуть створити комфортне кінематографічне середовище вдома.

Веб-додаток для організації домашнього кінотеатру - Це інтернет-магазин, який надає можливість клієнтам швидко та зручно придбати обладнання для домашнього кінотеатру. Через цей додаток клієнти можуть переглядати товари, дізнаватися деталі щодо їх характеристик, робити замовлення та отримувати їх у зручний для них спосіб.

Працівник веб-додатку - Особа, яка відповідає за обслуговування клієнтів та вирішення їхніх питань у веб-додатку для організації домашнього кінотеатру. Це можуть бути менеджери з обслуговування клієнтів, оператори чату або спеціалісти з підтримки, які забезпечують якісне обслуговування та консультації.

Кабінет клієнта та кабінет працівника - Особистий обліковий запис, який надає клієнтам та працівникам доступ до різноманітних функцій та можливостей у веб-додатку. Ці кабінети дозволяють клієнтам керувати своїми замовленнями та особистою інформацією, а працівникам — взаємодіяти з клієнтами, обробляти замовлення та надавати підтримку.

Ці поняття є ключовими для розуміння функціоналу та особливостей веб-додатку для організації домашнього кінотеатру на платформі WooCommerce.

Розуміння їх допоможе як клієнтам, так і працівникам ефективно користуватися цим додатком та отримувати максимальну вигоду від його використання.

1.3. Формулювання задачі

Метою цієї дипломної роботи є розробка функціонального та зручного для користувача веб-додатку для організації домашнього кінотеатру на основі платформи WooCommerce. Основним завданням є створення інтернет-магазину, який надасть клієнтам можливість швидко та зручно придбати обладнання та аксесуари для облаштування домашнього кінотеатру.

Ключовими завданнями проекту є:

- 1. Розробка інтерфейсу** - Створення зручного та привабливого інтерфейсу, який буде легко засвоюватися користувачами будь-якого рівня технічної підготовки. Інтерфейс повинен бути інтуїтивно зрозумілим та забезпечувати зручний пошук, перегляд та замовлення товарів.
- 2. Реєстрація та особистий кабінет** - Розробка системи реєстрації та особистого кабінету для клієнтів, яка дозволить зберігати особисту інформацію, переглядати історію замовлень та здійснювати оплату.
- 3. Функціонал для клієнтів** - Реалізація можливості перегляду деталей товарів, додавання їх до кошика, оформлення замовлення та оплати через різноманітні платіжні системи.
- 4. Функціонал для працівників** - Створення інтерфейсу для працівників, який дозволить переглядати та обробляти замовлення, здійснювати зв'язок з клієнтами та надавати підтримку.
- 5. Адаптивний дизайн** - Розробка веб-додатку з адаптивним дизайном, що забезпечить коректне відображення та зручне використання на різних пристроях, включаючи комп'ютери, планшети та смартфони.

Ці завдання спрямовані на створення функціонального та ефективного веб-додатку для організації домашнього кінотеатру, який забезпечить зручність та задоволення потреб користувачів.

РОЗДІЛ 2

ІНСТРУМЕНТИ РОЗРОБКИ

2.1. Вступ

Для успішної реалізації веб-додатку для організації домашнього кінотеатру необхідно чітко розуміти, що включає в себе його структура та архітектура. Цей розділ присвячений розгляду інструментів розробки, які були використані для створення серверної та клієнтської частин веб-додатку.

Серверна частина веб-додатку є важливим елементом, що забезпечує обробку запитів від клієнтів та зберігання необхідних даних. У нашому випадку, серверна частина реалізована з використанням локального сервера Open Server Panel, але також для розміщення веб-додатку можна використовувати відразу повноцінний хостинг. Такий підхід дозволяє ефективно взаємодіяти з базою даних, керувати контентом та забезпечувати функціональні можливості магазину.

Клієнтська частина веб-додатку відповідає за відображення та взаємодію з користувачем. Вона реалізована за допомогою різних технологій, зокрема HTML, CSS та JavaScript. Для зручності можна використовувати CMS, як у нашому випадку це Wordpress, що дозволяє створювати інтерактивний та зручний користувацький інтерфейс.

Цей розділ надасть детальний огляд інструментів та технологій, використаних у розробці обох частин веб-додатку, та процесу їхньої інтеграції для забезпечення оптимального функціонування системи.

2.2. Серверна частина

Open Server Panel (OSP) - є локальним серверним середовищем для розробки та тестування веб-додатків на базі Windows-операційних систем. Включає в себе веб-сервер Apache, базу даних MySQL, а також можливість встановлення додаткових компонентів, таких як PHP, Python, Perl та інші.

Функції:

1. **Локальне середовище розробки** - OSP дозволяє створювати та тестувати веб-додатки на локальному комп'ютері без необхідності встановлення окремих серверних компонентів.
2. **Керування веб-сервером** - Забезпечує можливість запуску, зупинки та налаштування веб-сервера Apache для відповіді на HTTP-запити.
3. **Управління базою даних** - Включає в себе інструменти для роботи з базою даних MySQL, такі як створення, редагування та видалення баз даних та таблиць.

Хостинг - є веб-сервісом, який надає фізичне або віртуальне місце для розміщення веб-сайтів та додатків на серверах, що підключені до Інтернету.

Функції:

1. **Публікація веб-сайтів** - Хостинг дозволяє розміщувати веб-сайти в Інтернеті, забезпечуючи їх доступність для користувачів з усього світу.
2. **Надійність та безпека** - Забезпечує надійність роботи веб-сайту та захист від втрати даних за рахунок регулярних резервних копій та застосування сучасних методів захисту інформації.
3. **Технічна підтримка** - Багато хостинг-провайдерів пропонують технічну підтримку для вирішення будь-яких проблем або запитань, пов'язаних з роботою веб-сайту.

MySQL база даних - це система управління базами даних (СУБД), яка використовує мову структурованих запитів SQL для взаємодії з даними.

Функції:

1. **Зберігання даних** - MySQL дозволяє зберігати великі обсяги даних у структурованому форматі, що спрощує їх подальшу обробку та аналіз.
2. **Операції з даними** - Забезпечує можливість виконувати різноманітні операції з даними, такі як додавання, оновлення, видалення та вибірка.
3. **Безпека даних** - MySQL має різні механізми безпеки, такі як автентифікація користувачів, контроль доступу та шифрування

даних, що забезпечує захист конфіденційності та цілісності інформації.

2.3. Клієнтська частина

Клієнтська частина веб-додатку - це фронтенд, або інтерфейс користувача, який бачить кінцевий користувач. Це та частина додатку, яка відповідає за відображення контенту, взаємодію з користувачем та виконання дій на стороні клієнта.

Функції:

1. **Відображення інтерфейсу** - Клієнтська частина відповідає за відображення вмісту веб-додатку у вигляді веб-сторінок, форм, кнопок, меню тощо.
2. **Взаємодія з користувачем** - Забезпечує можливість користувачам взаємодіяти з веб-додатком шляхом введення даних, натискання кнопок, вибору елементів тощо.
3. **Виконання дій на стороні клієнта** - Клієнтська частина відповідає за виконання дій, які не потребують взаємодії з сервером, таких як валідація форм, анімація, переключення вкладок тощо.

WordPress - це безкоштовна система управління контентом (CMS), яка дозволяє створювати та управляти веб-сайтами та блогами.

Функції:

1. **Створення вмісту** - WordPress надає зручний інтерфейс для створення, редагування та публікації контенту, такого як статті, фотографії, відео тощо.
2. **Управління веб-сайтом** - Забезпечує можливість керувати веб-сайтом, встановлювати теми, додатки, розширення, налаштовувати налаштування тощо.
3. **Розширення можливостей** - WordPress підтримує велику кількість розширень та плагінів, які дозволяють розширити функціональність сайту, таку як SEO, аналітика, безпека тощо.

WooCommerce - це безкоштовний плагін для WordPress, який дозволяє перетворити веб-сайт на повноцінний інтернет-магазин.

Функції:

1. **Створення магазину** - WooCommerce надає засоби для створення та управління каталогом товарів, категоріями, варіантами товарів тощо.
2. **Оформлення замовлень** - Забезпечує можливість користувачам додавати товари до кошика, оформляти замовлення та здійснювати оплату.
3. **Управління складом** - WooCommerce дозволяє керувати залишками товарів на складі, встановлювати ціни, знижки, акції тощо.

HTML, CSS та JavaScript (JS) - це основні мови, що використовуються для розробки клієнтської частини веб-додатків.

Функції:

1. **HTML (Hypertext Markup Language)** - Використовується для створення структури веб-сторінок та визначення їхнього контенту.
2. **CSS (Cascading Style Sheets)** - Використовується для визначення зовнішнього вигляду веб-сторінок, стилізації елементів та розміщення їх на сторінці.
3. **JavaScript** - Використовується для реалізації інтерактивності на веб-сторінках, додавання анімації, обробки подій та взаємодії з користувачем.

РОЗДІЛ 3

ЕТАПИ РОЗРОБКИ

3.1. Налаштування серверної частини

Для налаштування серверної частини веб-додатку використаємо програму Open Server Panel (OSP). Її потрібно спершу встановити на ПК з операційною системою Windows.

Після встановлення і запуску програми, на робочій панелі комп'ютера з'явиться іконка, при натисканні на яку ми можемо побачити контекстне меню програми. (Рис. 3.1.1.)

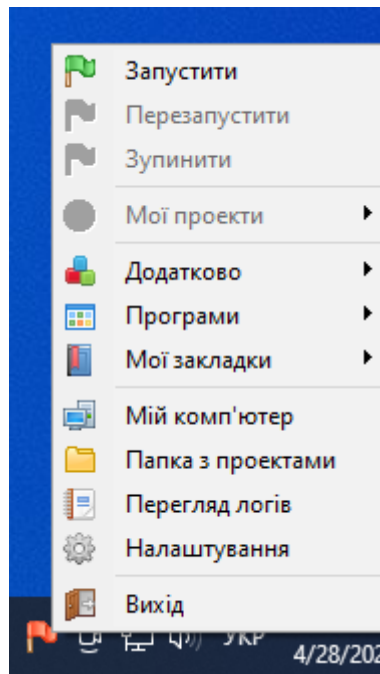


Рис. 3.1.1. Вигляд контекстного меню програми OSP.

У відкритому меню потрібно перейти в «Налаштування», після чого розгорнеться окреме вікно з різними підпунктами, тут нам потрібно перейти до пункту «Модулі». (Рис. 3. 1.2.)

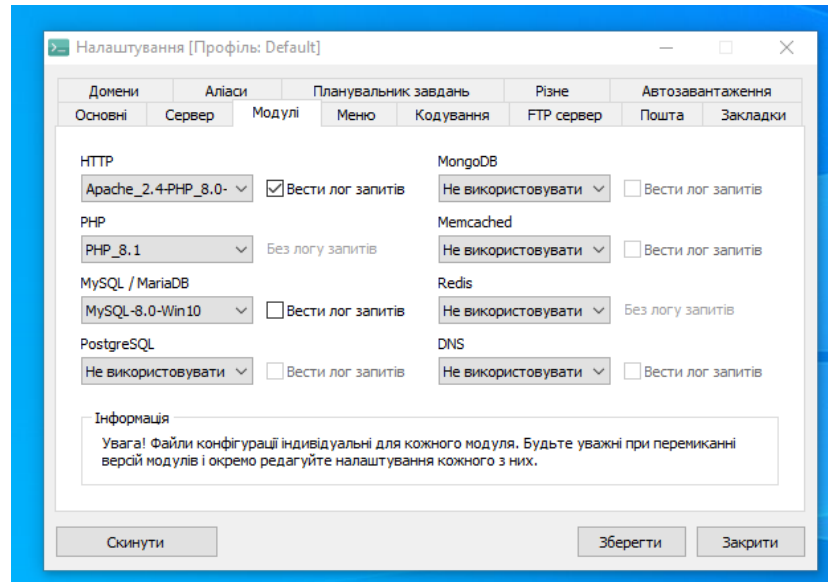


Рис. 3.1.2. Окреме вікно налаштувань програми OSP.

Для повноцінної роботи платформи Wordpress версії 6.5.2 нам потрібно вказати наступні налаштування

- HTTP – Apache_2.4-PHP_8.0-8.1
- PHP – PHP_8.1
- MySQL / MariaDB – MySQL-8/0-Win10

Саме такі налаштування рекомендовані розробниками платформи Wordpress в їх документації в розділі «Requirements».

Після виконання цих дій можемо запускати налаштований локальний сервер, для цього в контекстному меню програми OSP натискаємо пункт «Запустити». Наш сервер запущений та готовий до роботи.

Ще одним важливим налаштуванням перед розробкою клієнтської частини є створення бази даних. В програмі OSP, в модулях, ми вже вказали використання бази даних MySQL тому в контекстному меню програми, після її запуску, в пункті «Додатково», нам потрібно вибрати підпункт «PhpMyAdmin». (Рис. 3.1.3.)

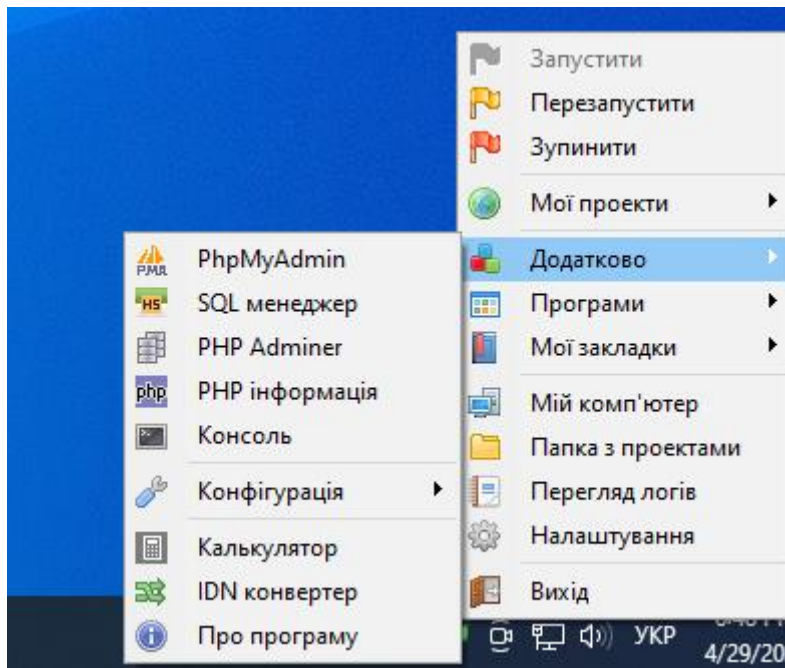


Рис. 3.1.3. Контексте меню програми OSP з вибором додаткових опцій.

PhpMyAdmin – це один із програмних інтерфейсів, написаний на PHP, який призначений для швидкого адміністрування бази даних. Даний інтерфейс дозволить нам швидко створити базу даних для нашого веб-додатку. Для цього у відкритому вікні натискаємо «New» та вказуємо назву нашої бази даних, з вибором набору символів, рекомендованим від Wordpress – «utf8mb4_general_ci». (Рис. 3.1.4.)

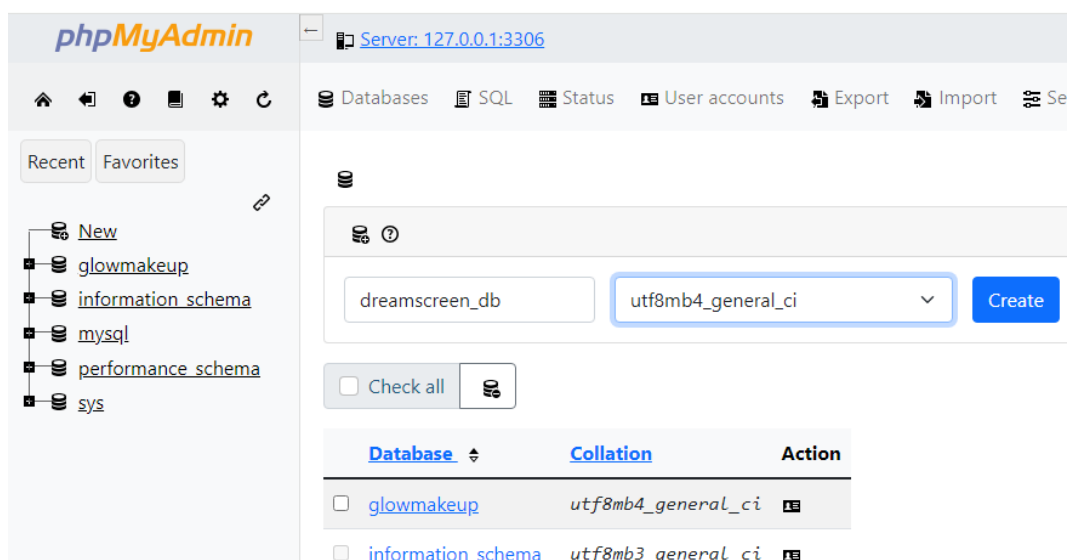


Рис. 3.1.4. Вигляд програмного інтерфейсу PhpMyAdmin.

Після виконання налаштувань та створення бази даних наш сервер готовий для розробки клієнтської частини.

3.2. Розробка клієнтської частини

3.2.1. Встановлення CMS Wordpress на локальний сервер

Для Розробки клієнтської частини нашого веб-додатку, будемо використовувати платформу Wordpress, яку для початку потрібно завантажити з офіційного сайту <https://wordpress.org/>. Після отримання архіву файлів для створення веб-додатку, його потрібно розгорнути в файлах нашого локального сервера за шляхом «"\\ospanel\domains\dreamscreen"», де dreamscreen це назва папки, яка і буде доменною назвою нашого веб-додатку.

Після розархівування файлів Wordpress в нашій папці локального сервера, ми можемо відкрити в браузері наступне посилання «http://dreamscreen/», ця дія дозволить локальному серверу запусити нашу папку, і тим самим запусити установщик Wordpress. (Рис. 3.2.1.1.)

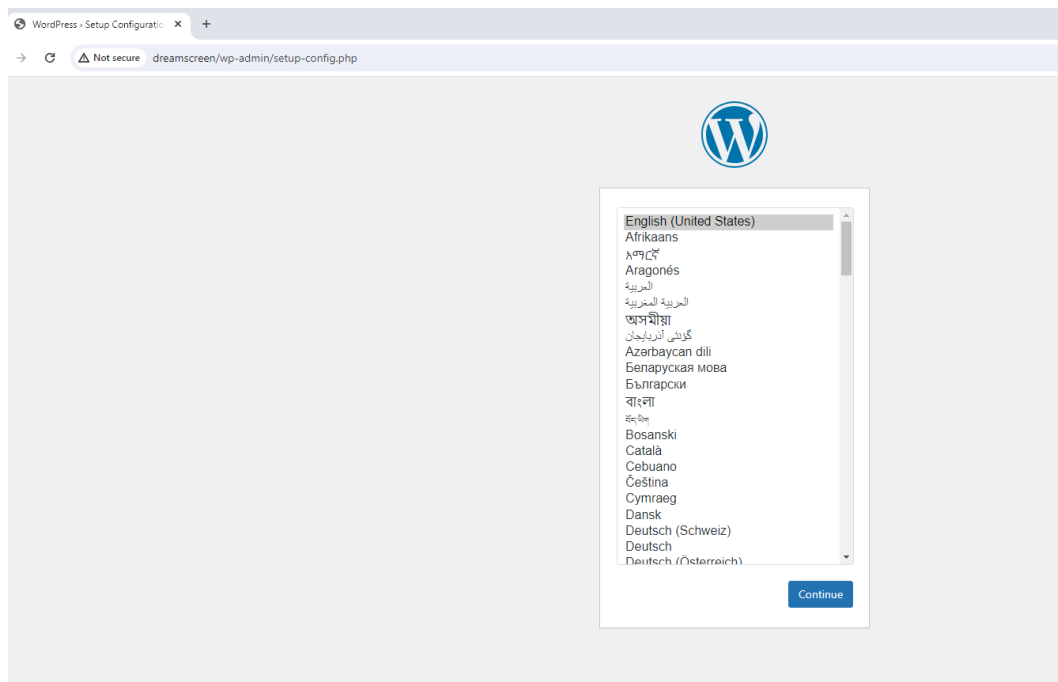
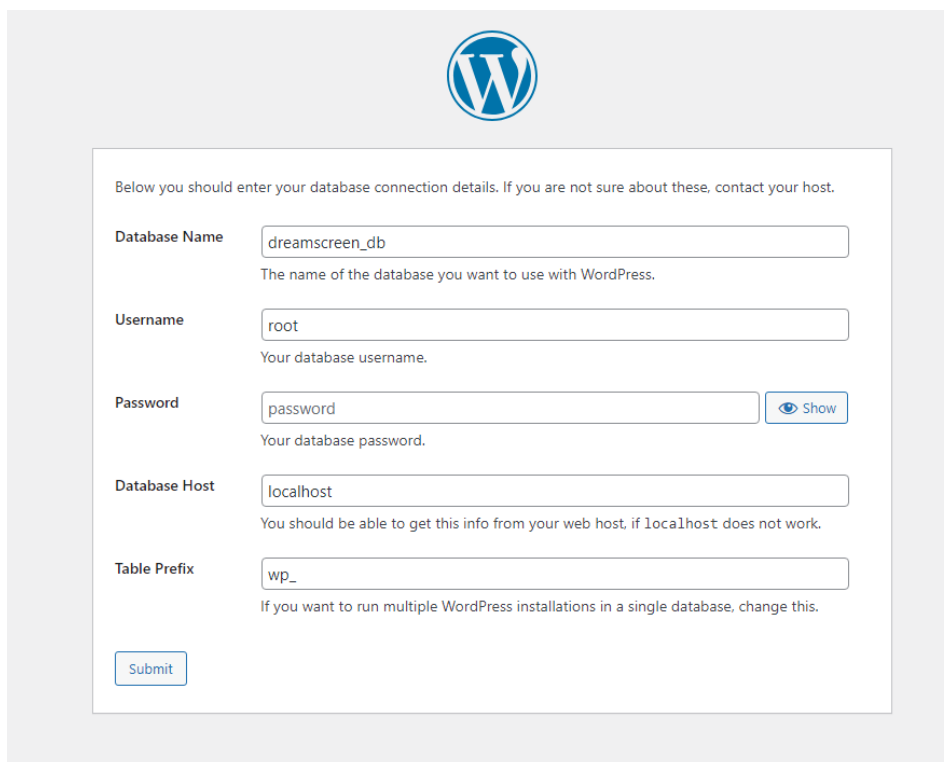


Рис. 3.2.1.1. Запуск папки веб-додатку у браузері.

Як бачимо на Рис. 3.2.1.1. перед нами з'явився вибір мови для інсталяції платформи Wordpress. Для зручності оберемо мову за замовчуванням «English (United State)», так як в процесі розробки будемо звертатися до документації, яка достовірніше викладена саме на англійській мові.

Наступним етапом буде підключення до бази даних, яку ми створювали раніше. (Рис. 3.2.1.2.) За замовчування в програмі OSP, ім'я користувача бази даних є «root», а пароль не потребується.



Below you should enter your database connection details. If you are not sure about these, contact your host.

Database Name
The name of the database you want to use with WordPress.

Username
Your database username.

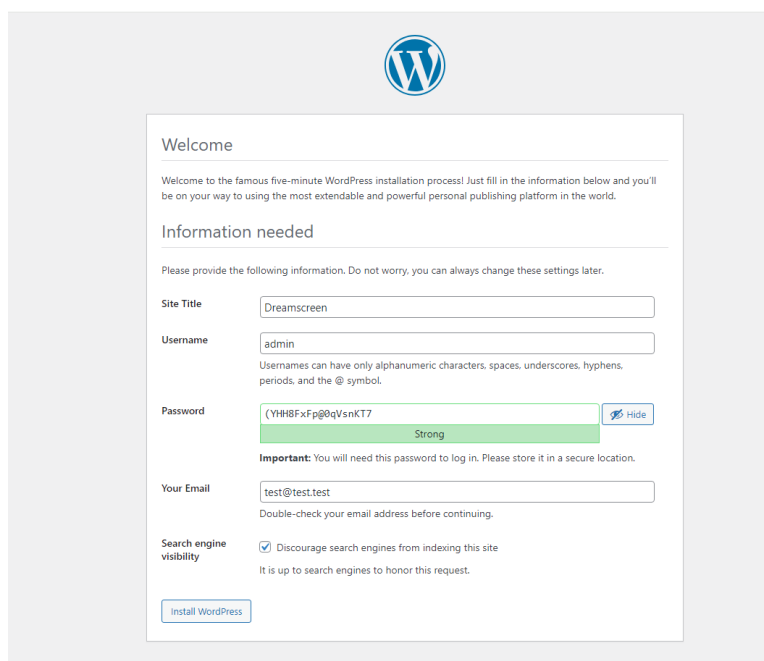
Password
Your database password.

Database Host
You should be able to get this info from your web host, if localhost does not work.

Table Prefix
If you want to run multiple WordPress installations in a single database, change this.

Рис. 3.2.1.2. Підключення до бази даних.

Після успішного підключення до бази даних, у нас з'являється нове вікно, де можна вказати назву веб-додатку та вказати доступи для адміністратора. (Рис. 3.2.1.3.)



Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Do not worry, you can always change these settings later.

Site Title

Username
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password
Strong

Important: You will need this password to log in. Please store it in a secure location.

Your Email
Double-check your email address before continuing.

Search engine visibility Discourage search engines from indexing this site
It is up to search engines to honor this request.

Рис. 3.2.1.3. Створення доступів адміністратора веб-додатку

Після виконання цих дій, нас перенаправить на панель адміністратора (Рис. 3.2.1.4.) що означає що наша платформа Wordpress інстальована на локальний сервер та готова для розробки веб-додатку.

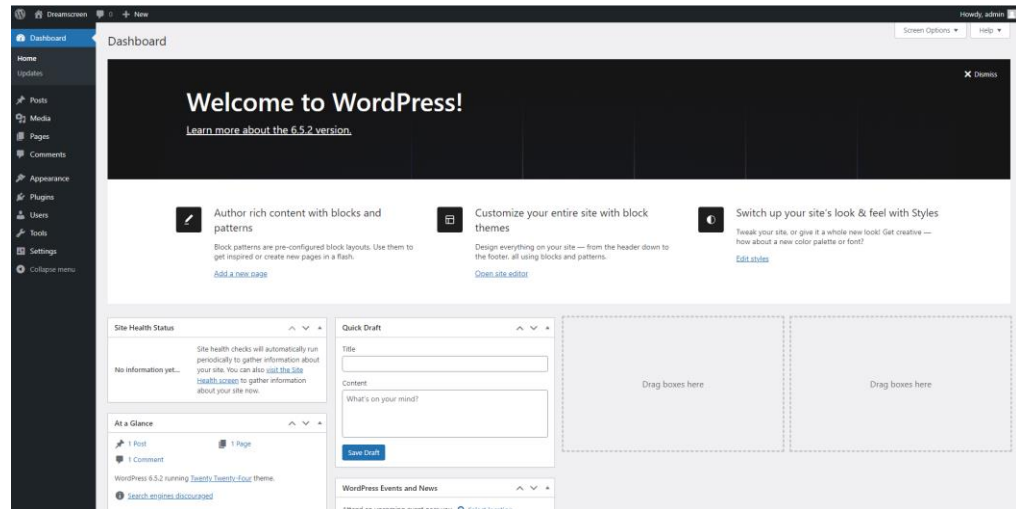


Рис. 3.2.1.4. Панель адміністратора веб-додатку на Wordpress

3.2.2. Створення власної теми для веб додатку

CMS Wordpress після встановлення пропонує на вибір велику кількість вже готових тем, з різним функціоналом, але найкращим варіантом, для нас, буде створення власної теми з використанням і написанням тільки того функціоналу, який нам потрібний. Для цього потрібно перейти та ознайомитися з документацією Wordpress в розділ «Theme Development».

Для створення та ініціалізації власної теми, потрібно відкрити папку тем за шляхом «\ospanel\domains\dreamscreen\wp-content\themes» (для роботи з файлами, краще відразу відкрити нашу папку з доменним ім'ям у будь якому зручному редакторі коду, у нашому випадку це буде PhpStorm). Тут ми створюємо нову папку «dreamscreen», в якій створюємо файл «index.html» та «style.css», в останньому файлі прописуємо початкову інформацію про нашу тему, як вказано в документації. (Рис. 3.2.2.1.)

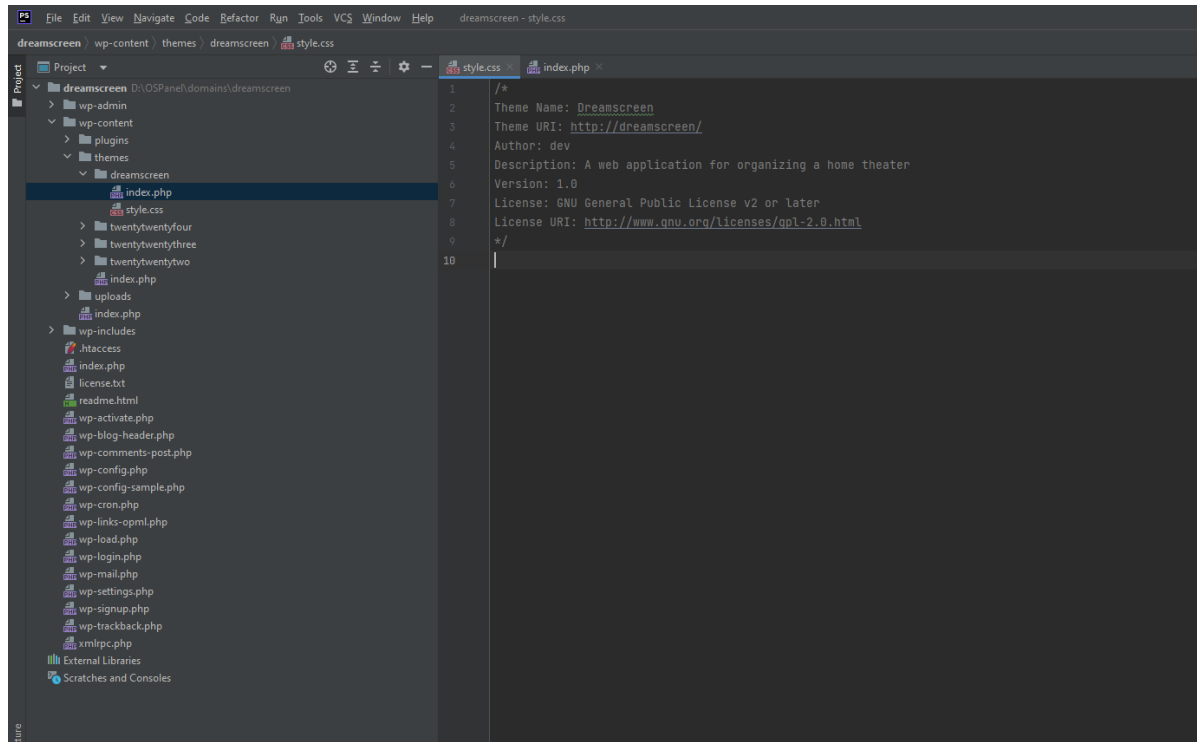


Рис. 3.2.2.1.

Після цього, платформа Wordpress ініціалізує нашу тему та виведе її для активації на панелі адміністратора в пункті «Appearance». (Рис. 3.2.2.2.)

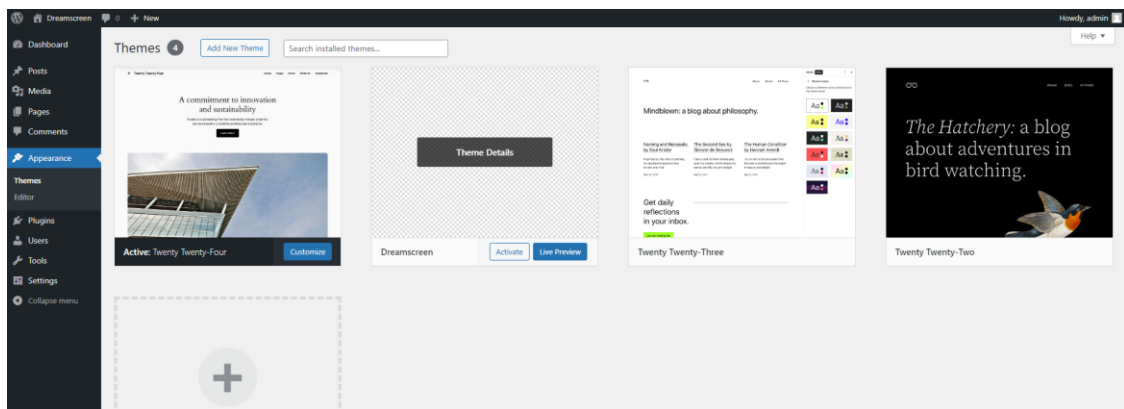


Рис. 3.2.2.2.

Наша початкова тема створена та готова до подальших дій. Також для продовження, спершу слід ознайомитись з «Template Hierarchy» в документації для тем Wordpress. (Рис. 3.2.2.3.) Це допоможе зрозуміти структуру файлів яку потрібно використовувати при створенні теми.



Рис. 3.2.2.3. Ієрархія файлів для тем Wordpress

Як бачимо з Рис. 3.2.2.3. в кінці ієрархії стоїть файл `index.php` який ми створювали для ініціалізації теми. Використовуючи знання PHP та HTML та вказівки з документації розробляємо ось таку структуру як на Рис. 3.2.2.4.

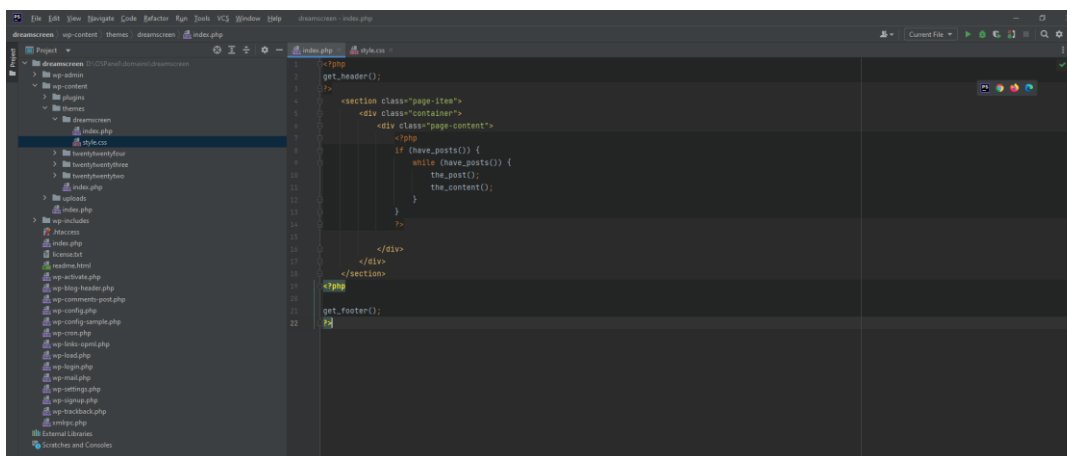


Рис. 3.2.2.4.

Додамо деякі інші файли з якими будемо працювати в нашій темі:

- `screenshot.png` – фото, що буде показано як preview нашої теми.
- `404.php` – файл шаблону сторінки 404.
- `header.php` – файл шапки веб-додатку.
- `footer.php` – файл підвалу веб-додатку.

- front-page.php – файл шаблону головної сторінки.
- page.php – файл шаблону за умовчуванням для інших сторінок.
- searchform.php – файл шаблону виводу форми для пошуку по сайту.
- single-post.php – файл шаблону сторінок постів.
- functions.php – основний файл для додавання різних функцій веб-додатку.

Також додамо в нашу тему папку «assets» в якій додамо css та js файли, для роботи із стилями та скриптами. На даному етапі маємо таку структуру власної теми як на Рис. 3.2.2.5.

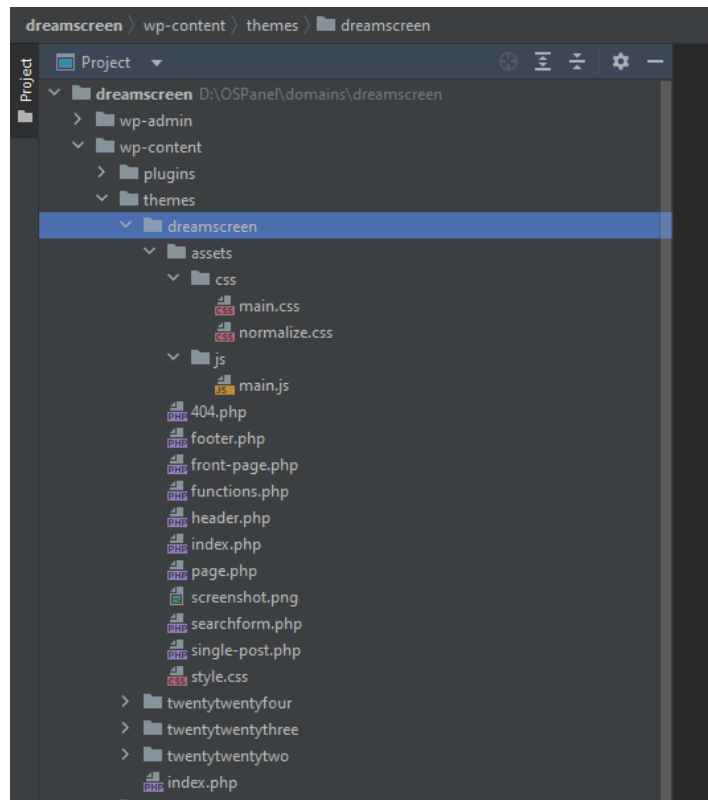


Рис. 3.2.2.5.

Наступним етапом в файлі «functions.php» потрібно додати підключення стилів та скриптів, увімкнути підтримку меню, віджетів, логотипу та інших функцій, які потрібні для реалізації нашого вуб-додатку.

3.2.3. Встановлення Woocommerce

Наступним важливим етапом розробки є встановлення плагіну Woocommerce, який дозволить зробити з нашого веб-додатку повноцінний інтернет-магазин.

Для цього потрібно перейти до пункту «Plugins» на панелі адміністратора та натиснути «Add New Plugin» де в поле пошуку потрібно вписати слово «Woocommerce». Після виконання пошуку, система запропонує нам різні варіанти, пов'язані з цим ключовим словом, нам потрібно вибрати той варіант, де автором є «By Automattic» після чого натиснути «Install Now». (Рис. 3.2.3.1.)

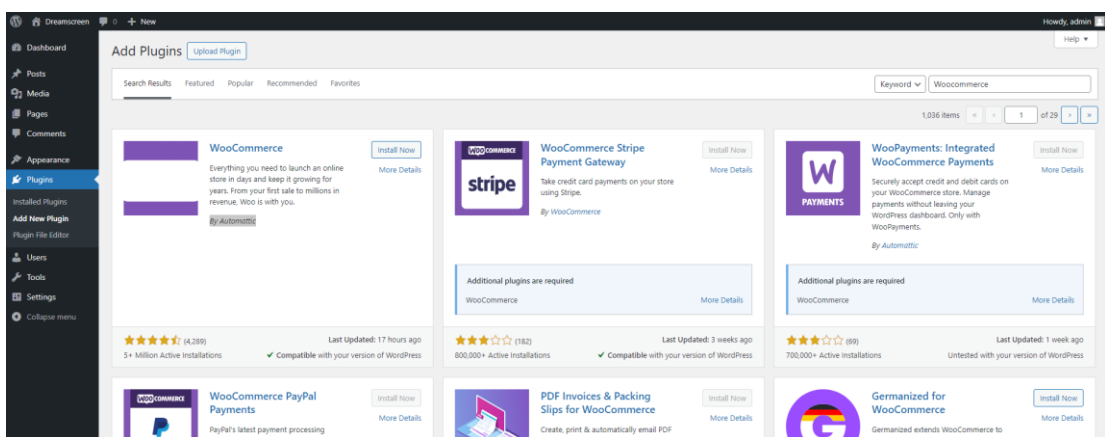


Рис. 3.2.3.1.

Після встановлення та активації нам буде запропоновано пройти швидке налаштування, де потрібно ввести адресу, валюту, методи доставки та інші дані щодо нашого інтернет магазину.

3.2.4. Створення сторінок, меню, віджетів та продуктів для веб-додатку

Далі потрібно продумати дизайн нашого веб додатку, додати та заповнити сторінки, на їх основі створити меню, додати та вивести віджети з контактною інформацією.

Для початку додамо сторінки, які потрібні для нашого сайту. Деякі сторінки Woocommerce вже створив автоматично, тому нам лишається додати ще декілька сторінок. які ми запланували для виведення користувачам веб-додатку. Сторінки можемо створити перейшовши на панелі адміністратора в пункт «Pages» (Рис. 3.2.4.1.) та натиснувши «Add New Page».

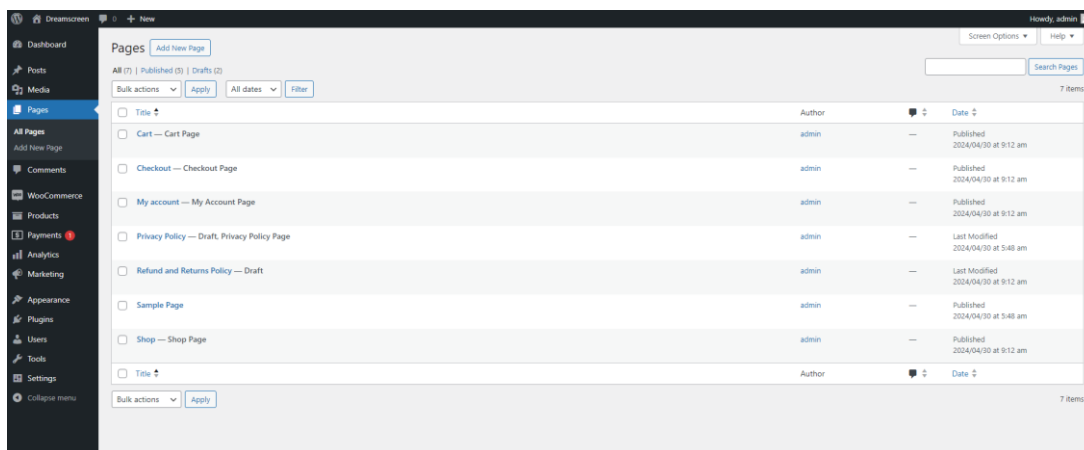


Рис. 3.2.4.1.

При створенні сторінок можемо відразу їх заповняти, для цього у Wordpress встановлений редактор Gutenberg, який дозволяє зібрати сторінки веб-додатку з різних блоків (Рис. 3.2.4.2.), а пізніше на PHP можна буде вказати де потрібно виводити їхній вміст.

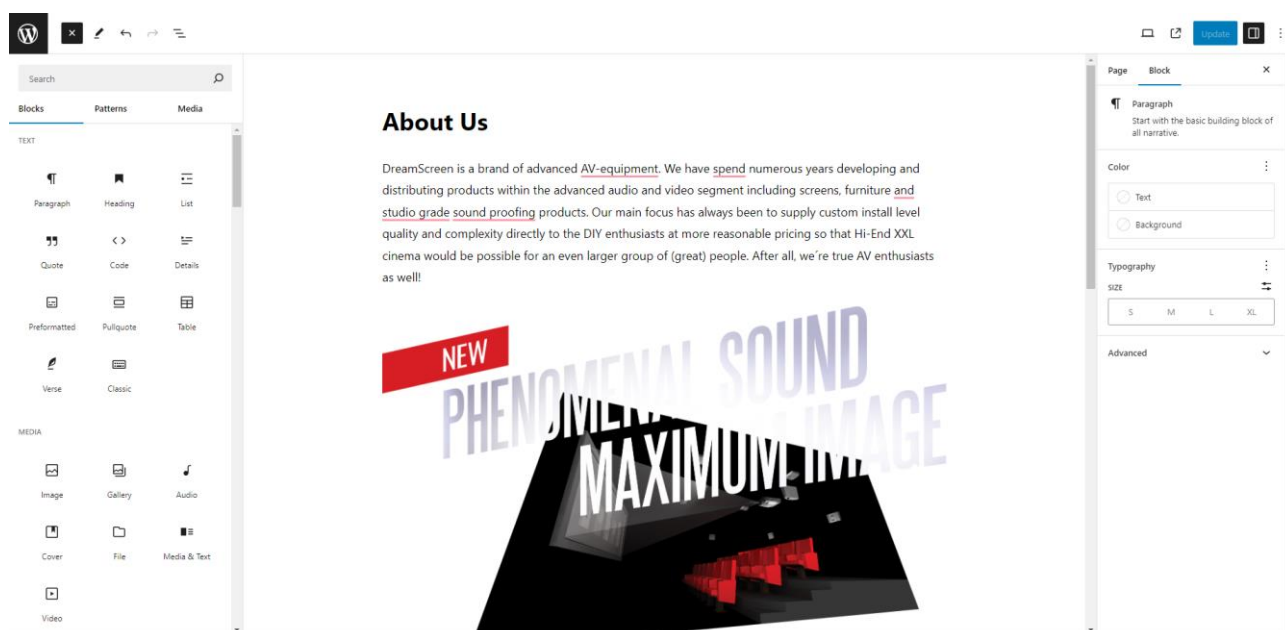


Рис. 3.2.4.2.

Якщо ми хочемо додати ще декілька слайдерів для картинок на наших сторінках, тоді встановимо декілька плагінів, таких як «Smart Slider 3» або «MetaSlider», які дозволяють швидко створити, редагувати та вивести потрібний слайдер. Для виводу такі плагіни зазвичай використовують shortcode – це функціонал Wordpress, який дозволяє запрограмувати потрібну функцію в короткий код, який можна запусити вивівши на сторінці. Для прикладу за

допомогою плагіну «Smart Slider 3» створимо слайдер картинок для головної сторінки і виведемо його. (Рис. 3.2.4.3.)

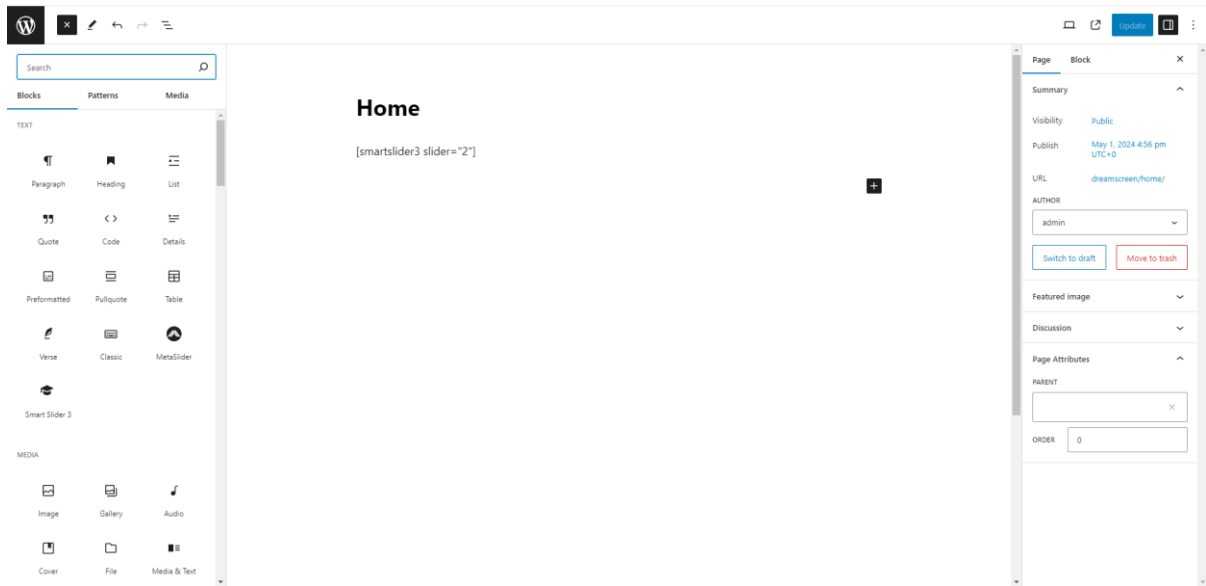


Рис. 3.2.4.3.

Коли всі потрібні сторінки створенні, перейдемо в «Settings» на панелі адміністратора, щоб дозаповнити дані веб-додатку:

- Короткий опис веб-додатку
- Favicon картинка
- Структура посилань
- Назначення головної сторінки

Далі перейдемо до створення основного меню для нашого веб-додатку, для цього в нашій темі, ми додаємо функцію, яку запускаємо на hook «after_setup_theme». (Рис. 3.2.4.4.) Це дозволяє нам ініціалізувати наше меню на панелі адміністратора.

```
function createMenu()
{
    register_nav_menu( location: 'header', description: 'Header Menu');
    register_nav_menu( location: 'footer', description: 'Footer Menu');
}
add_action('after_setup_theme', 'createMenu');
```

Рис. 3.2.4.4.

Після зберігання і перезавантаження панелі адмінстратора, в пункті «Appearance» з'являється підпункт «Menus», де ми можемо створити наше основне меню для веб-додатку. (Рис. 3.2.4.5.)

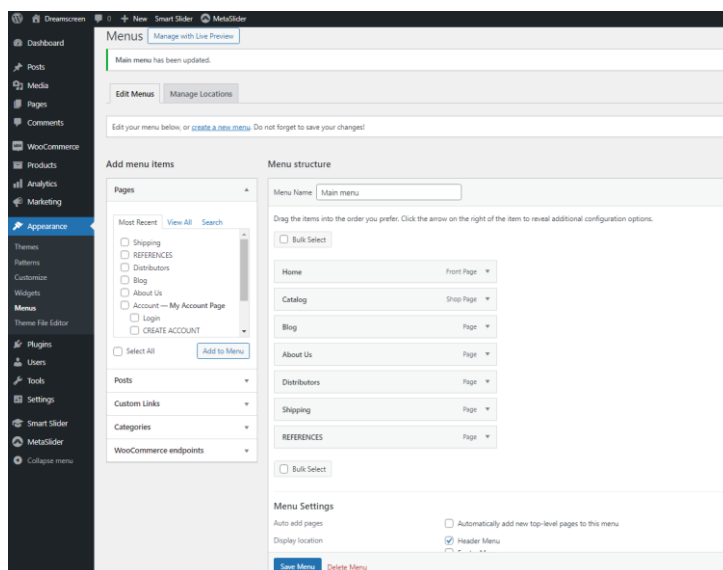


Рис. 3.2.4.5.

Щоб вивести контактну інформацію в нашому веб-додатку будемо використовувати віджети, для їх створення додамо функцію в нашому файлі «functions.php», яка допоможе ініціалізувати функціонал додавання віджетів. (Рис. 3.2.4.6.)

```

59
60 add_action('widgets_init', 'createWidgetZone');
61
62 function createWidgetZone()
63 {
64     register_sidebar([
65         'name' => 'Main sidebar',
66         'id' => "main-sidebar",
67         'before_widget' => '<div id="main-sidebar" class="widget">',
68         'after_widget' => "</div>\n",
69         'before_title' => '<h3 class="widgettitle">',
70         'after_title' => "</h3>\n",
71     ]);
72     register_sidebar([
73         'name' => 'QUICK LINKS',
74         'id' => "quick-links-widget",
75         'before_widget' => '<div id="quick-links-widget-block">',
76         'after_widget' => "</div>\n",
77     ]);
78     register_sidebar([
79         'name' => 'GET IN TOUCH',
80         'id' => "get-in-touch-widget",
81         'before_widget' => '<div id="get-in-touch-widget-block">',
82         'after_widget' => "</div>\n",
83     ]);
84     register_sidebar([
85         'name' => 'COPYRIGHT',
86         'id' => "copyright-widget",
87         'before_widget' => '<div id="copyright-widget" class="widget">',
88         'after_widget' => "</div>\n",
89     ]);
90 }
91
  
```

Рис. 3.2.4.6.

Після додавання функції для віджетів, на панелі адміністратора в пункті «Appearance» з'являється підпункт «Widgets». Перейшовши по ньому, ми можемо бачити окремі блоки які ми створити за допомогою функцій, тепер ми можемо наповнити даними ці блоки. (Рис. 3.2.4.7.)

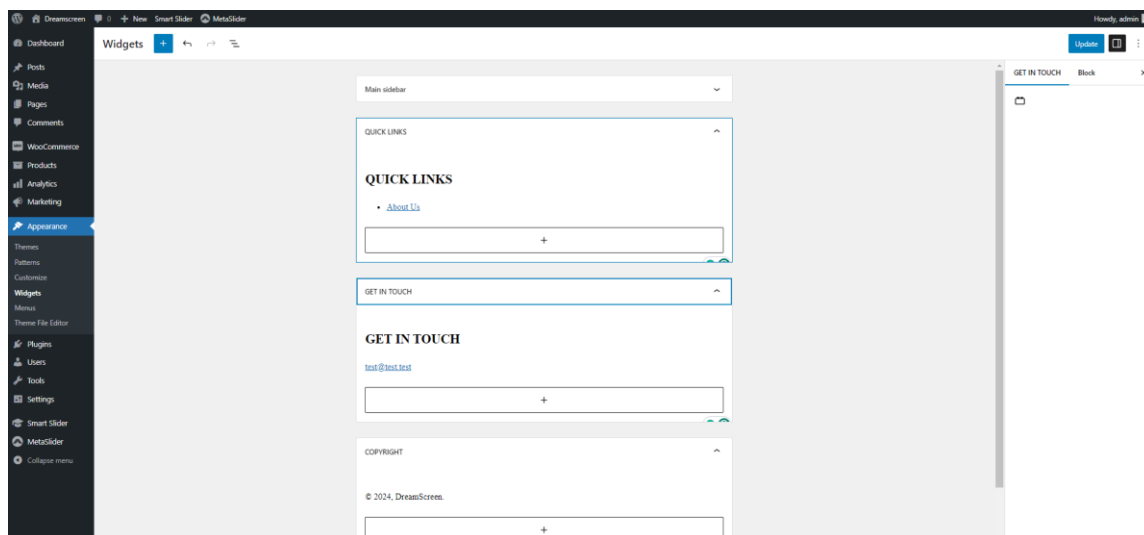


Рис. 3.2.4.7.

Наступним етапом буде створення продуктів до нашого веб-додатку, розподіл їх на категорії та додавання до їх тегів та атрибутів. Це все ми можемо зробити, перейшовши до пункту «Products» на панелі адміністратор. (Рис. 3.2.4.8.)

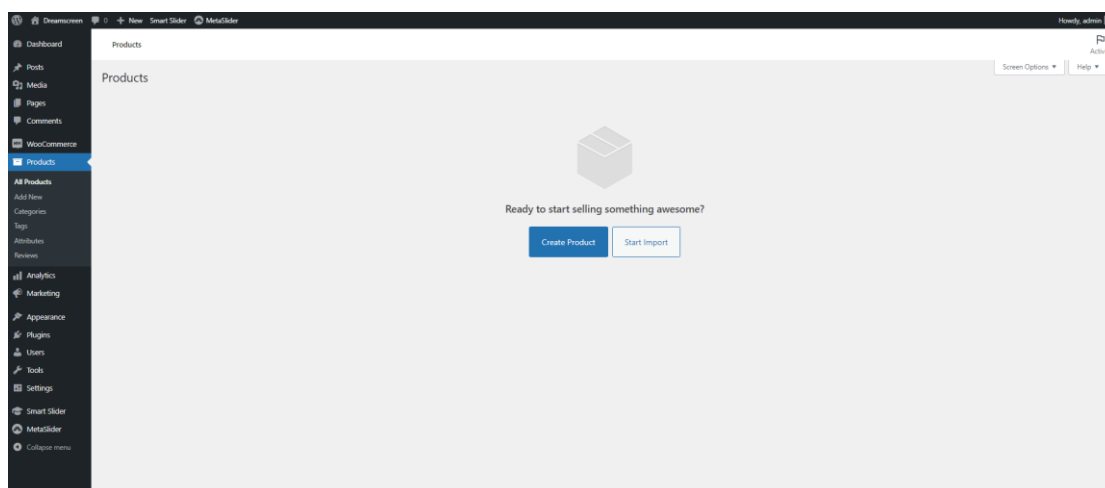


Рис. 3.2.4.8.

3.2.5. Створення різних функцій та робота з файлами теми

Після створення всіх сторінок, продуктів, та налаштувань через панель адміністратора можна перейти до функцій виводу вмісту.

Для початку створимо файли css та js для роботи із стилями та скриптами. Для цього додамо в нашу тему папку «assets» в яку будемо додавати наші файли стилів та скриптів. Далі в файлі «functions.php», додамо функцію на хук «wp_enqueue_scripts», яка ініціалізує та підключить наші стилі до сторінок веб-додатку. (Рис. 3.2.5.1.)

```
function addStyle()
{
    wp_enqueue_style( handle: 'font-oswald', src: 'https://fonts.googleapis.com/css?family=Oswald&display=swap');
    wp_enqueue_style( handle: 'font-open-sans', src: 'https://fonts.googleapis.com/css?family=Open+Sans&family=Oswald&display=swap');
    wp_enqueue_style( handle: 'dream_css_normalize', src: get_template_directory_uri() . '/assets/css/normalize.css');
    wp_enqueue_style( handle: 'dream_css_main', src: get_template_directory_uri() . '/assets/css/main.css');
    wp_enqueue_style( handle: 'style', get_stylesheet_uri());
}

add_action( wp_enqueue_scripts, 'addStyle');

function addScript()
{
    wp_deregister_script( handle: 'jquery-core');
    wp_deregister_script( handle: 'jquery');

    wp_register_script( handle: 'jquery-core', src: 'https://code.jquery.com/jquery-3.5.1.min.js', deps: false, ver: null, args: true);
    wp_register_script( handle: 'jquery', src: false, array( 'jquery-core' ), ver: null, args: true);

    wp_enqueue_script( handle: 'dream_js_main', src: get_template_directory_uri() . '/assets/js/main.js', [], ver: '', args: true);

    wp_enqueue_script( handle: 'jquery');
    wp_localize_script( handle: 'jquery', object_name: 'myAjax', array(
        'url' => admin_url( 'admin-ajax.php' ),
    )
    );
}

add_action( wp_enqueue_scripts, 'addScript');
```

Рис. 3.2.5.1.

Перейдемо до файлу «header.php», він підвантажується до кожної сторінки, яку ми відкриваємо в нашому веб-додатку, за допомогою функції get_header(). Тому пропишемо тут html код для ініціалізації кожної сторінки, додамо основні теги, пропишемо структуру для меню, в яку додамо функцію виводу нашого створеного меню, а також додамо функціонал швидких посилань, пошук по сайту та посилання на корзину і сторінку акаунта клієнта. (Рис. 3.2.5.2.)

```

1 <!DOCTYPE html>
2 <html <?php language_attributes(); ?>>
3 <head>
4 <meta charset="<?php bloginfo( show: 'charset'); ?>" />
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <link rel="pingback" href="<?php bloginfo( show: 'pingback_url'); ?>" />
7 <?php wp_head(); ?>
8 </head>
9 <body <?php body_class(); ?>>
10 <main>
11 <header>
12 <div class="container">
13 <div class="header_content">
14 <div class="header_cap"...>
69 <div class="header_menu">
70 <div class="header-mobile-nav">
71 <div class="header-mobile-nav-icon">
72 <button class="c-hamburger c-hamburger--htx">
73 <span>toggle menu</span>
74 </button>
75 </div>
76 <div class="header-mobile-nav-text"...>
79 </div>
80 <div class="header_nav header_nav_descst"...>
92 <div class="header_account"...>
100 </div>
101 </div>
102 </div>
103 </header>
104

```

Рис. 3.2.5.2.

У файлі «footer.php» додамо закриваючі теги, які виводять основний вміст наших сторінок. Також введемо створені віджети, за допомогою функції «dynamic_sidebar()». (Рис. 3.2.5.3.) Footer на кожній сторінці веб-додатку виводимо за допомогою функції get_footer().

```

<footer>
  <div class="container">
    <div class="footer_widget">
      <div id="quick-links-widget" class="widget">
        <?php dynamic_sidebar( index: 'quick-links-widget') ?>
      </div>
      <div id="get-in-touch-widget" class="widget">
        <?php dynamic_sidebar( index: 'get-in-touch-widget') ?>
      </div>
    </div>
    <div class="footer_copyright">
      <?php dynamic_sidebar( index: 'copyright-widget') ?>
    </div>
  </div>
</footer>

<?php wp_footer(); ?>
</main>
</body>
</html>

```

Рис. 3.2.5.3.

Після підключення стилів, заповнення header та footer, можемо перейти до виводу головної сторінки в файлі «front-page.php». Так як в редагуванні сторінки ми додали шорткод слайдера, то виведемо його через стандартний цикл while у Wordpress, використовуючи функцію «the_content()». (Рис. 3.2.5.4.)

```
<div class="slider-home">
  <?php if (have_posts()) {
    while (have_posts()) {
      the_post();
      the_content();
    }
  } ?>
</div>
```

Рис. 3.2.5.4.

Далі за допомогою функції «get_categories()» отримаємо наші продукти по категоріях. Функція повертає нам масив даних, який ми опрацюємо через цикл foreach і далі вже виводимо ті дані, які потрібно вивести для карток продуктів. (Рис. 3.2.5.5.)

```
<div class="product-category-home">
  <?php $categories = get_categories([
    'taxonomy' => 'product_cat',
    'type' => 'product',
  ]);
  if ($categories) {
    foreach ($categories as $cat) {
      $thumbnail_id = get_term_meta($cat->term_id, 'thumbnail_id', true);
      $image = wp_get_attachment_url($thumbnail_id);
      $url = get_category_link($cat->term_id); ?>
      <div class="category-preview-item"
        style="...">
        <a href="<?php echo $url; ?>" class="category-link"></a>
        <div class="category-content">
          <div class="category-title">
            <?php echo $cat->name; ?>
          </div>
          <div class="category-view">
            <a href="<?php echo $url; ?>"> view all</a>
          </div>
        </div>
      </div>
    } ?>
  } ?>
</div>
```

Рис. 3.2.5.5.

Також використаємо глобальний об'єкт WP_Query за допомогою якого, вказавши певні параметри, отримаємо лише продукти з тегом featured, тобто

отримаємо та виведемо блок з рекомендованими продуктами. Дані отримані з о'єкту опрацьовуємо в стандартному циклі постів - while. (Рис. 3.2.5.6.)

```

<div class="popular-product">
  <div class="popular-product-all">
    <a href="/featured" class="popular-product-all-link"></a>
    <p class="popular-product-all-button">View All</p>
  </div>
  <?php global $product;
  $args = array(
    'post_type' => array('product'),
    'tax_query' => array(
      'relation' => 'AND',
      array(
        'taxonomy' => 'product_visibility',
        'field' => 'name',
        'terms' => 'featured',
      ),
      array(
        'taxonomy' => 'product_visibility',
        'field' => 'name',
        'terms' => 'exclude-from-catalog',
        'operator' => 'NOT IN',
      ),
    ),
    'order' => 'desc',
    'posts_per_page' => 4,
  );
  $popular_products = new WP_Query($args);
  if ($popular_products->have_posts() :
  while ($popular_products->have_posts() : $popular_products->the_post(); ?>
    <div class="popular-product-item">
      <a href="<?php the_permalink(); ?>" class="view-button">View</a>
      <?php wc_get_template( 'template_name: 'loop/sale-flash.php'); ?>
      <a class="popular-product-item-link" href="<?php the_permalink(); ?>"></a>
      <div class="popular-product-item-img">...</div>
      <div class="popular-product-item-desc">...</div>
    </div>
  <?php endwhile;
  endif;
  wp_reset_postdata(); ?>
</div>

```

Рис. 3.2.5.6.

В процесі додаємо блокам html повноцінні назви класів, та формуємо структуру, яку стилізуємо через css у вже створеному та підключеному файлі стилів. На цьому головна сторінка готова.

Для стандартного шаблону сторінок (шаблон, який використовується за замовчуванням), додамо окрему структуру в файл «page.php». (Рис. 3.2.5.7.)

```

<?php get_header(); ?>
<section class="page-item">
  <div class="container">
    <div class="page-content">
      <?php if (have_posts()) {
        while (have_posts()) {
          the_post();
          the_content();
        } ?>
      </div>
    </div>
  </div>
</section>
<?php get_footer(); ?>

```

Рис. 3.2.5.7.

Для того щоб створити окремий шаблон для певної сторінки, у Wordpress ми можемо створити файл «page-name_template.php», де name_template потрібно замінити назвою шаблону. Створимо декілька окремих шаблонів та створимо для їх окрему папку «template-parts» для зручності. (Рис. 3.2.5.8.)

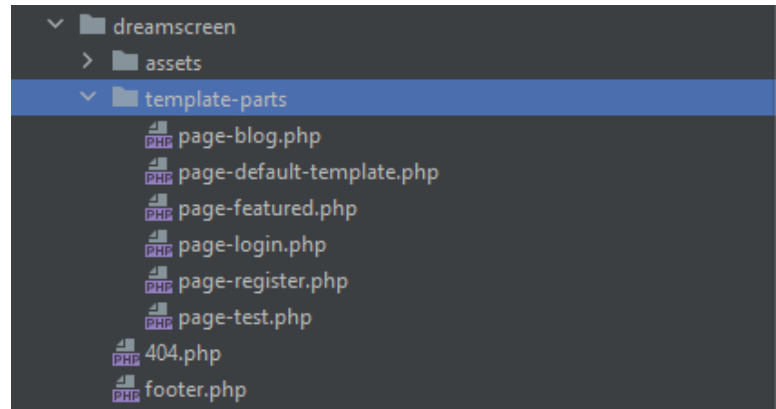


Рис. 3.2.5.8.

Також в нашому веб-додатку створено сторінку «Blog», на якій будемо виводити різні пости. Тож створимо також окремий шаблон для постів. Для цього потрібно лише створити файл з назвою «single-post.php».

3.2.6. Зміна шаблонів Woocommerce

Платформа Woocommerce вже має готові шаблони з окремими стилями, для таких сторінок як:

- Сторінка каталогу
- Сторінка корзини
- Сторінка оплати
- Сторінка реєстрації / логування клієнта
- Сторінка акаунту клієнта

Щоб налаштувати структуру цих шаблонів під наш веб-додаток, ми можемо скопіювати їх з папки плагіну в нашу тему, як це вказано в документації Woocommerce, після цього вже можемо вносити свої зміни в структуру, та налаштовувати свої стилі. (Рис. 3.2.6.1.)

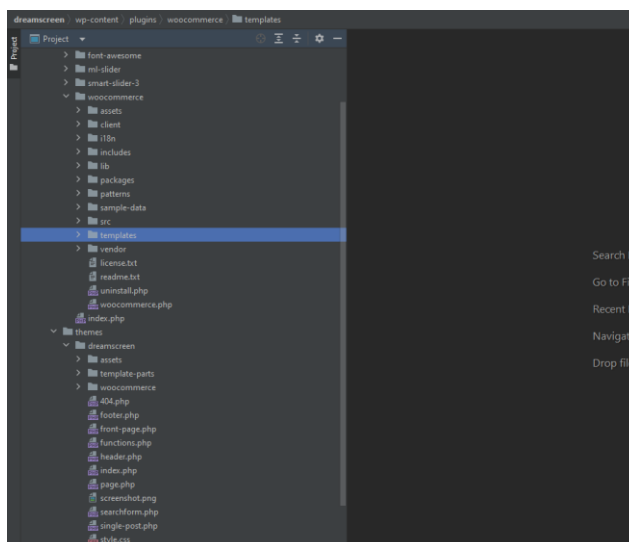


Рис. 3.2.6.1.

Для того щоб налаштувати вивід продуктів, на сторінці категорій та сторінці каталогу, у woocommerce є окремий файл - «archive-product.php», який ми вже перенесли в папку «woocommerce» в нашій темі. В ньому зробимо перевірку чи це сторінка каталогу, чи це сторінка категорії, щоб вивести потрібні продукти в потрібному місці. Також в цьому файлі додамо перевірку на сторінку пошуку, і коли користувачі будуть проводити пошук по сайту за ключовими словами, їм підвантажиться структура вже з нашого файлу архіву. Для таких перевірок в Woocommerce вже існують функції «is_search()» та «is_tax()». (Рис. 3.2.6.2.)

```

<?php
/** The Template for displaying product archives, including the main shop page which is a post type archive ... */
defined( 'ABSPATH' ) || exit;

get_header( 'shop' );
if ( is_search() ) {
    <section class="page-item">
        <div class="container">
            <div class="page-content">
                <div class="page-catalog">
                    <div class="page-search">
                </div>
            </div>
        </div>
    </section>
} else {
    <section class="page-item">
        <div class="container">
            <div class="page-content">
                <div class="page-catalog">
                    <?php
                    if ( is_tax() ) {
                        <div class="page-category">
                    <?php } else {
                        <div class="product-category">
                    <?php }
                </div>
            </div>
        </div>
    </section>
}
get_footer( 'shop' );

```

Рис. 3.2.6.2.

Для зручності у Woocommerce всі блоки розділені на окремі файли, що дозволяє підтягувати певні блоки відразу в різні місця. Тому для виводу структури картки продукту, на сторінці категорій та каталогу, існує окремий файл «content-product.php», в якому ми прописуємо структуру нашої картки товару, підтягуючи різними хуками потрібні дані. (Рис. 3.2.6.3.)

```

<?php
/** The template for displaying product content within loops ...*/

defined( 'ABSPATH' ) || exit;

global $product;

// Ensure visibility.
if ( empty( $product ) || ! $product->is_visible() ) {
    return;
}
?>
<?php wc_product_class( class '', $product ); ?>
<a href="#">?php echo get_permalink( $product->get_id() );?>" class="view-button">View</a>
</php

/**
 * Hook: woocommerce_before_shop_loop_item.
 *
 * @hooked woocommerce_template_loop_product_link_open - 10
 */
do_action( 'hook_name: woocommerce_before_shop_loop_item' );

/**
 * Hook: woocommerce_before_shop_loop_item_title.
 *
 * @hooked woocommerce_show_product_loop_sale_flash - 10
 * @hooked woocommerce_template_loop_product_thumbnail - 10
 */
do_action( 'hook_name: woocommerce_before_shop_loop_item_title' );

/**
 * Hook: woocommerce_shop_loop_item_title.
 *
 * @hooked woocommerce_template_loop_product_title - 10
 */
do_action( 'hook_name: woocommerce_shop_loop_item_title' );

```

Рис. 3.2.6.3.

Також у Woocommerce є окремий файл, який відповідає за вивід сторінки продукту – це «single-product.php». В якому ми можемо змінити структуру обгортки продукта. (Рис. 3.2.6.4.)

```

<?php
/** The Template for displaying all single products ...*/

if ( ! defined( 'ABSPATH' ) ) {
    exit; // Exit if accessed directly
}

get_header( 'shop' ); ?>
<section class="page-item page-item-product">
    <div class="container">
        <div class="page-content">
            <div class="page-product">
                <?php while ( have_posts() ) : ?>
                    <?php the_post(); ?>

                    <?php wc_get_template_part( 'slug:content', 'name: single-product' ); ?>

                <?php endwhile; // end of the loop. ?>

                <div class="page-share">
                    <?php dynamic_sidebar( 'share-products' ); ?>
                </div>
            </div>
        </div>
    </div>
</section>

<?php
get_footer( 'shop' );

/* Omit closing PHP tag at the end of PHP files to avoid "headers already sent" issues. */

```

Рис. 3.2.6.4.

Наповнення сторінки продукту розміщено в окремому файлі «content-single-product.php», в якому підключені хуки, які підтягують різні блоки. Всі блоки для виводу сторінки продукту розміщені в окремій папці «single-product» (Рис. 3.2.6.5.) це дозволяє за потреби відключити деякі блоки, які ми не бажаємо виводити.

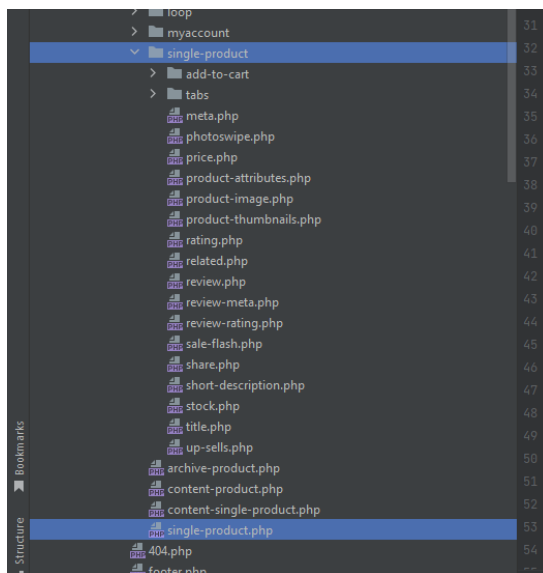


Рис. 3.2.6.5.

Тепер коли в нас налаштовані всі сторінки на панелі адміністратора та в кодї, ми можемо додати стилі для зміни вигляду, та деякі скрипти для динамічних блоків анімаці, тим самим реалізуємо зручний та естетичний дизайн нашого веб-додатку.

РОЗДІЛ 4

ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ UI

4.1. Головна сторінка

Щоб протестувати головну сторінку, відкриємо її в браузері за доменним ім'ям «dreamscreen/». Тут ми побачимо повноцінний header, в якому розміщено логотип, іконку для пошуку по сайту, корзину, меню та посилання для входу в особистий акаунт. Також відразу буде видно першу секцію, в якій розміщується інформаційний слайдер останніх новинок. (Рис. 4.2.1.)

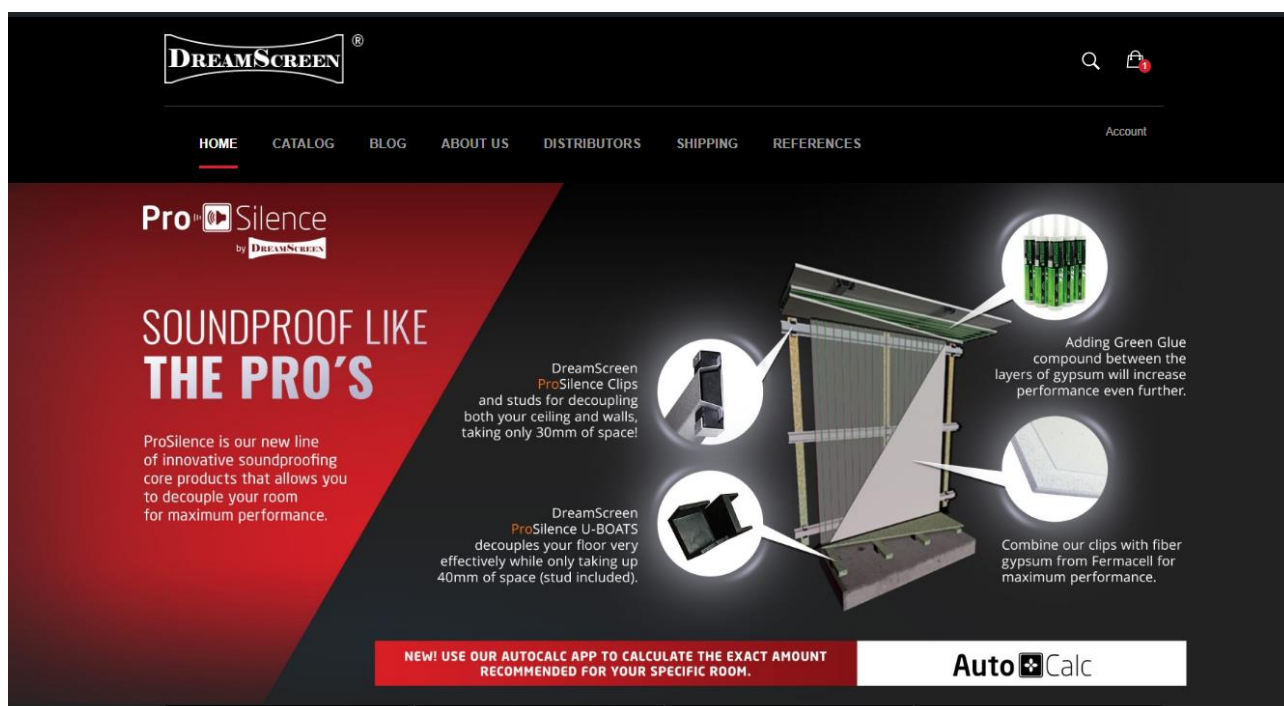


Рис. 4.2.1.

Наступні дві секції в тестуванні - це посилання на категорії товарів та секція рекомендованих товарів. (Рис. 4.2.2.)

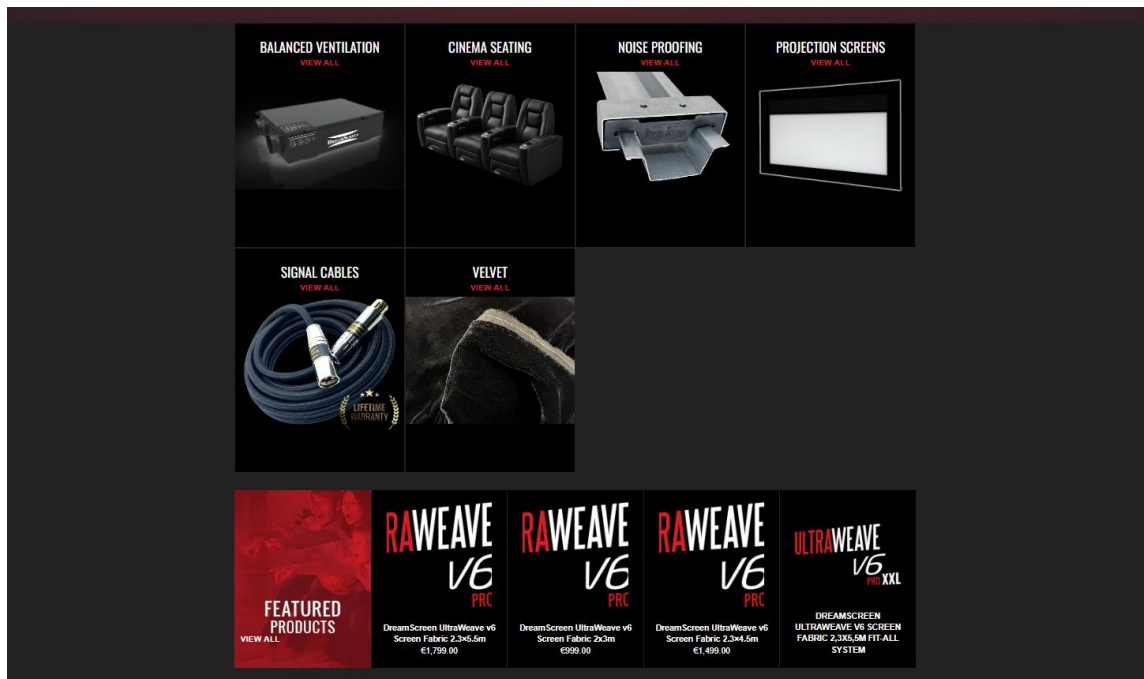


Рис. 4.2.2.

Завершаюча секція головної сторінки – це footer. В ньому розміщений віджет швидких посилань, контактна пошта, та віджет для підписки на оновлення. (Рис. 4.2.3.)

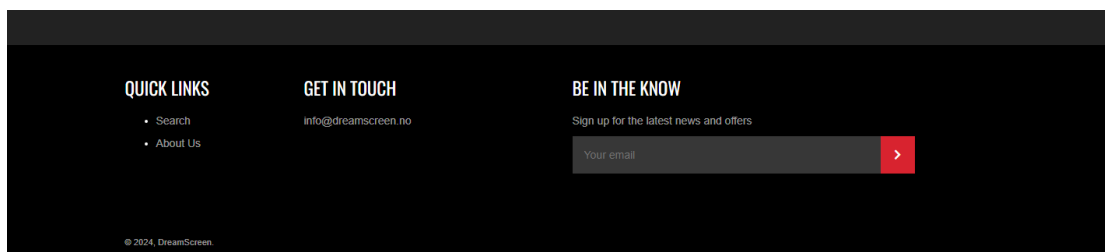


Рис. 4.2.3

4.2. Сторінка каталогу

Наступною в нашому тесті є сторінка каталогу товарів. Для зручності тут є розподіл на категорії. (Рис. 4.3.1.)

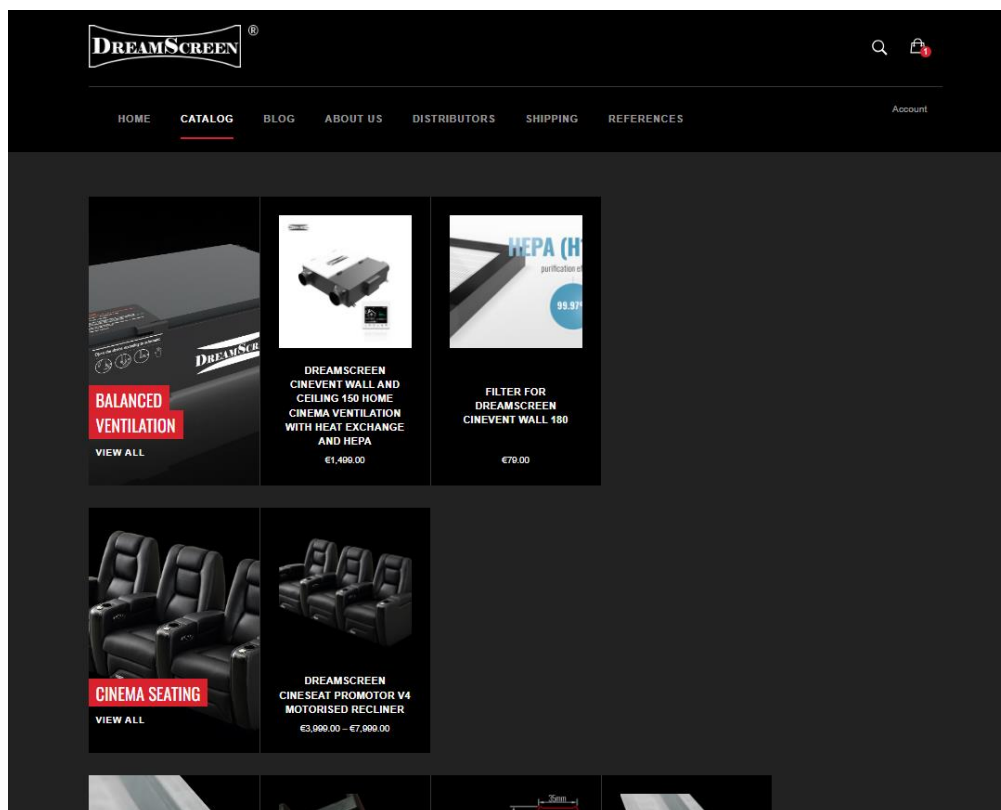


Рис. 4.3.1.

4.3. Сторінка категорії товарів

Відразу із головної сторінки або сторінки каталогу, перейдемо на сторінку категорій, для вибору і порівняння певних груп товарів. (Рис. 4.4.1.)

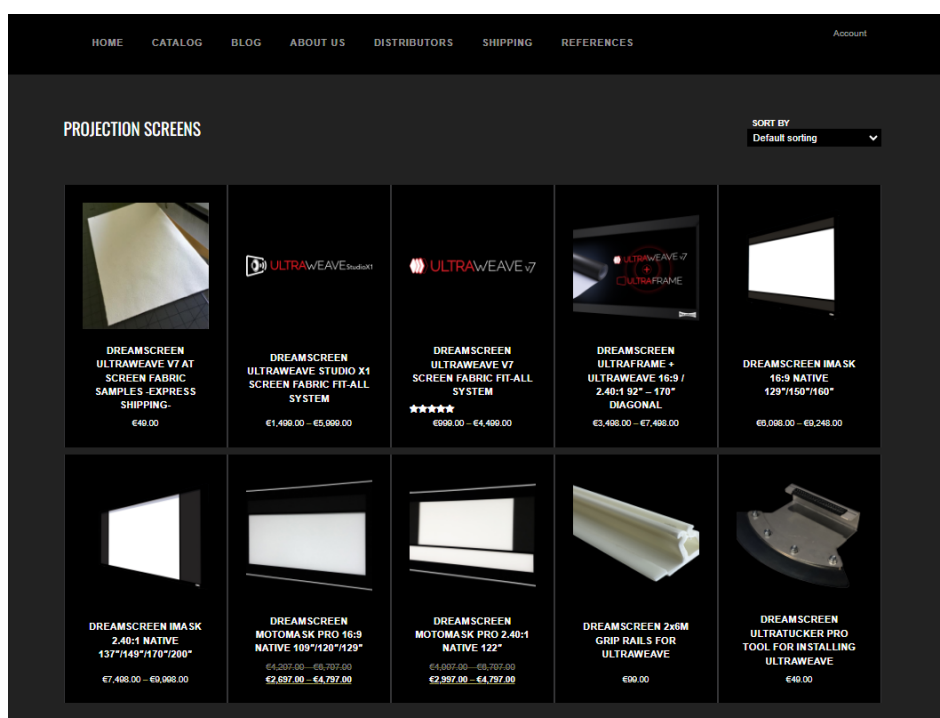


Рис. 4.4.1.

4.4. Сторінка товару

Перейшовши на сторінку товару, буде показано основний слайдер з картинками (Рис. 4.5.1.), назва, ціна та кнопка для покупки (Рис. 4.5.2.) Також нижче розміщується опис товару. (Рис. 4.5.3.).

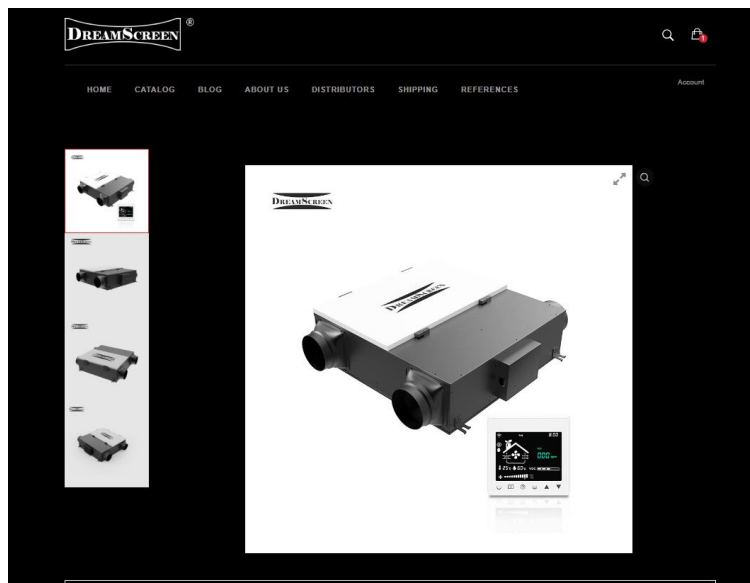


Рис. 4.5.1.

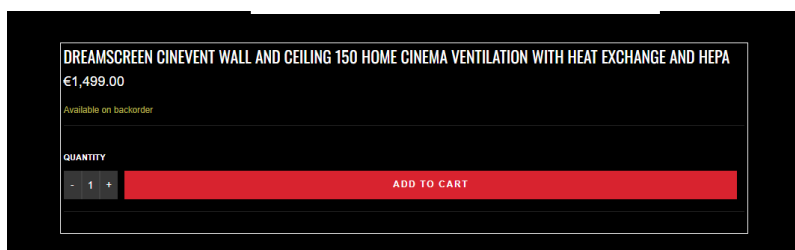


Рис. 4.5.2.

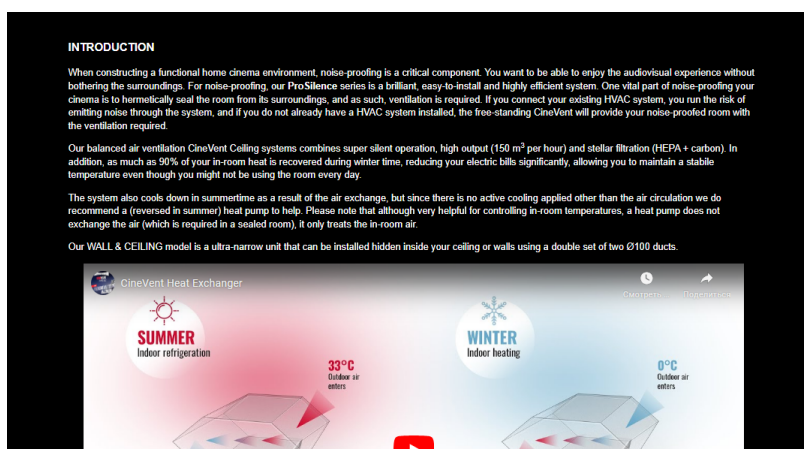


Рис. 4.5.3.

За потреби користувачі можуть отримати консультацію, для цього можна написати лист напряму через контактну форму, або ж в правому кутку веб-додатку є іконка, яка відкриває чат консультації онлайн. (Рис. 4.5.4.)

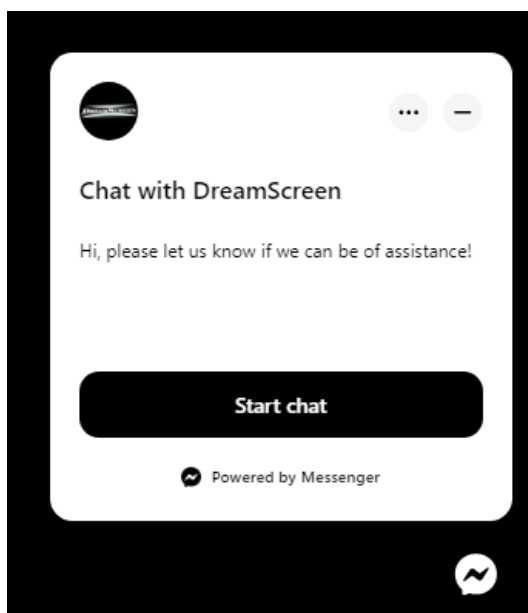


Рис. 4.5.4.

4.5. Сторінка корзини

Після того, як натиснемо кнопку «Add To Cart» на сторінці товару, нас перенаправить на сторінку корзини, де є можливість побачити всі деталі замовлення та скористатися купоном (якщо такий надається). (Рис. 4.6.1.)

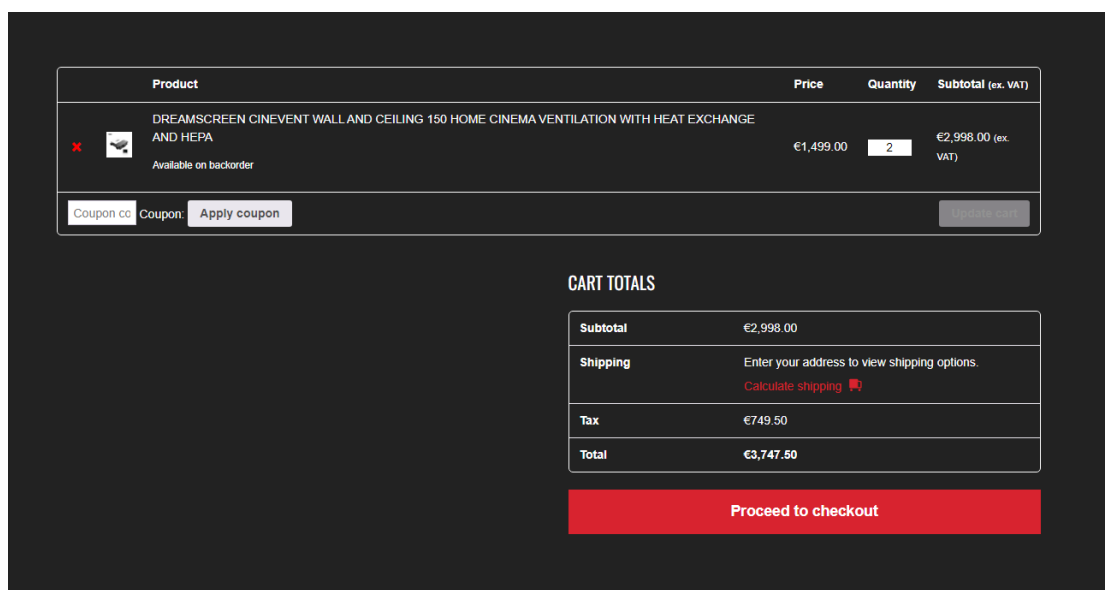


Рис. 4.6.1.

4.6. Сторінка оплати

Коли натискаємо на «Proceed to checkout» на сторінці корзини, нас перенаправляє на сторінку оплати, де потрібно вказати свої дані, адресу доставки та вибрати метод оплати. (Рис. 4.7.1.)

Phone:

Email address:


Subscribe to our newsletter

Your order

| Product | Subtotal (ex. VAT) |
|---|--|
| DREAMSCREEN CINEVENT WALL AND CEILING 150 HOME CINEMA VENTILATION WITH HEAT EXCHANGE AND HEPA × 2 | €2,998.00 (ex. VAT) |
| Subtotal | €2,998.00 |
| Shipping | Enter your address to view shipping options. |
| Tax | €749.50 |
| Total | €3,747.50 |

Direct bank transfer

Make your payment directly into our bank account. Please use your Order ID as the payment reference. Your order will not be shipped until the funds have cleared in our account.

PayPal  [What is PayPal?](#)

[Place order](#)

Рис. 4.7.1.

Після успішного, тестового, оформлення замовлення, нам буде показано його деталі (Рис. 4.7.2.), які також будуть продубльовані на електронну пошту замовника.

HOME CATALOG BLOG ABOUT US DISTRIBUTORS SHIPPING REFERENCES Account

Thank you. Your order has been received.

ORDER NUMBER: 476 DATE: May 3, 2024 TOTAL: €999.00 PAYMENT METHOD: Cash on delivery

Pay with cash upon delivery.

ORDER DETAILS

| Product | Total |
|---|-------------------------|
| DREAMSCREEN CINEVENT WALL 180 HOME CINEMA VENTILATION WITH HEAT EXCHANGE AND HEPA × 1 | €999.00 |
| Subtotal: | €999.00 |
| Shipping: | Free shipping |
| Payment method: | Cash on delivery |
| Total: | €999.00 |

Рис. 4.7.2.

4.7. Сторінка акаунта користувача

Акаунт користувача створюється автоматично, коли користувач створює замовлення. Йому на пошту приходить лист з даними для входу і посилання для створення особистого паролю. Після успішного логування, користувач може перейти в свій акаунт, де зберігаються всі його замовлення, та дані щодо адреси доставки (які відповідно можна тут змінити). (Рис. 4.8.1.)

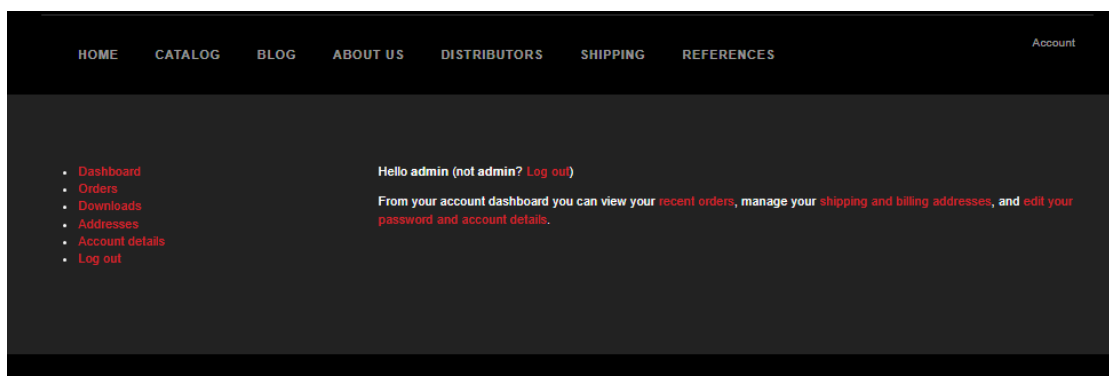


Рис. 4.8.1.

Перевіримо вхід в свій акаунт. Це можна зробити, перейшовши за посиланням в header – «Account» та вказавши пошту і пароль. (Рис. 4.8.2.)

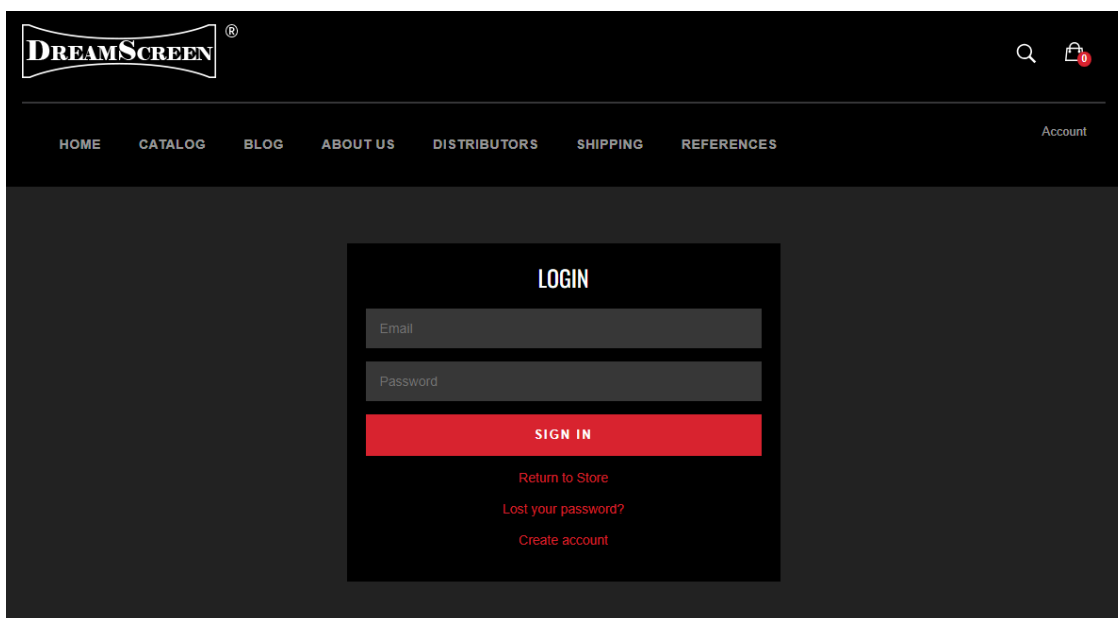


Рис. 4.8.2.

Якщо користувач хоче створити акаунт, ще до створення замовлення, він може натиснути «Create account» на сторінці логування, тим самим з'явиться форма реєстрації. (Рис. 4.8.3.)

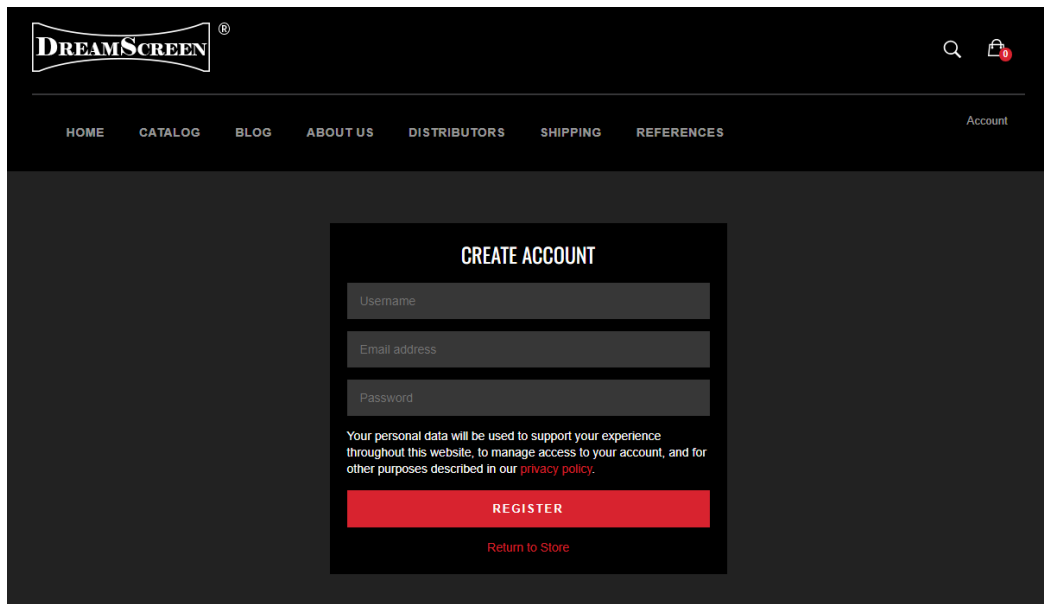


Рис. 4.8.3.

4.8. Додаткові сторінки веб-додатку

Користувач за бажанням також може ознайомитися з іншими, цікавими сторінками сайту, такими як блог (Рис. 4.9.1.), доставка, дистриб'ютори, про нас і такі інші. (Рис. 4.9.2.)

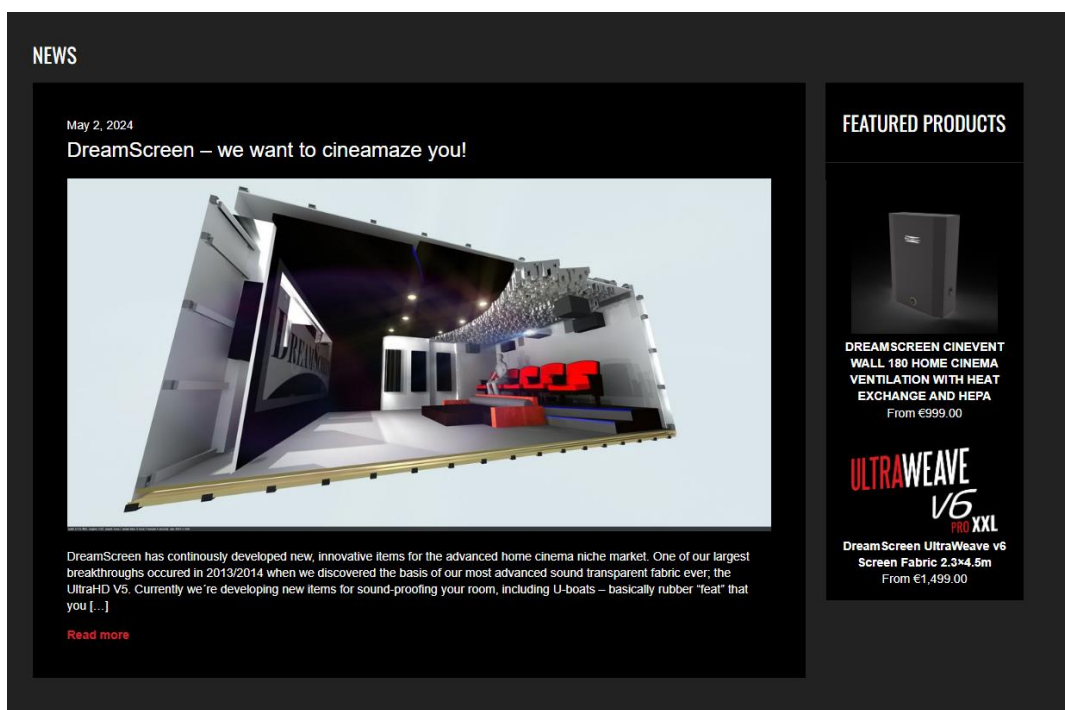


Рис. 4.9.1.

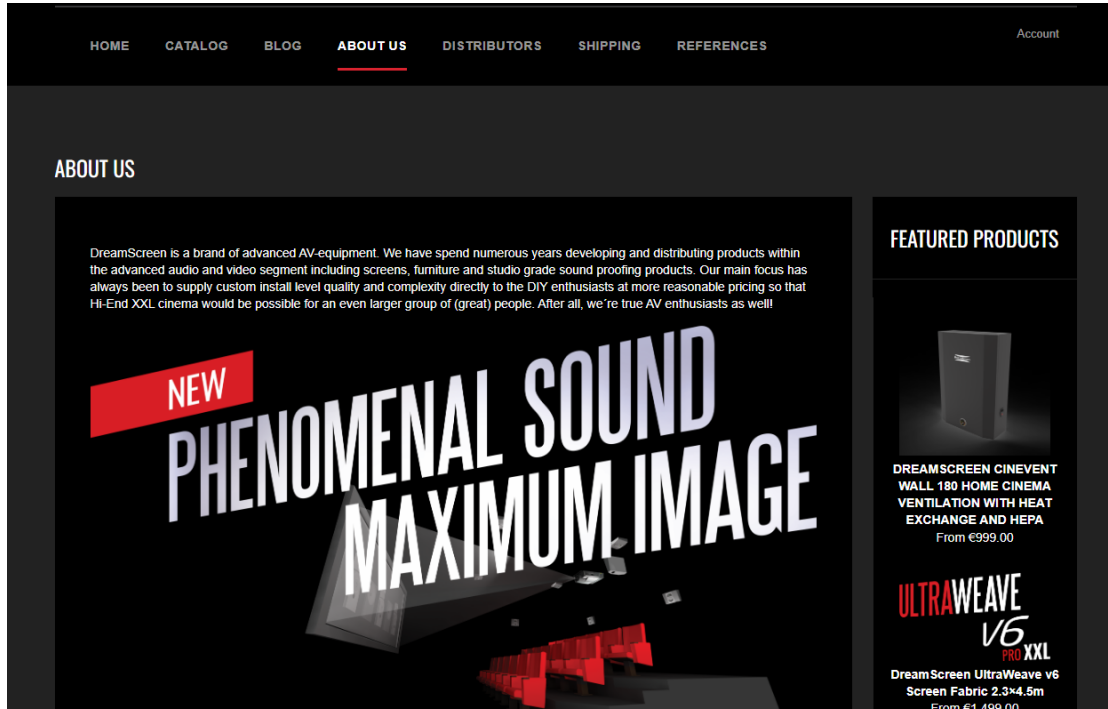


Рис. 4.9.2.

ВИСНОВОК

Результатом виконаної кваліфікаційної роботи є розроблений веб-додаток для організації домашнього кінотеатру на основі платформи Woocommerce. Цей веб-додаток не лише відповідає функціональним вимогам, але й має естетично привабливий та інтуїтивно зрозумілий інтерфейс для користувача.

В процесі розробки було поставлено конкретні завдання, включаючи створення серверної та клієнтської частини. Для забезпечення серверної частини було використано програму Open Server Panel, в якій також була налаштована взаємодія з базою даних. Для клієнтської частини було використано WordPress та плагін Woocommerce, які дозволили створити основу додатку.

У процесі розробки було надано особливу увагу як функціональним, так і естетичним аспектам. Постійне тестування дозволило виявити та виправити усі функціональні та візуальні дефекти, забезпечуючи високий рівень якості кінцевого продукту.

Таким чином, розроблений веб-додаток є результатом систематичної роботи, спрямованої на створення високоякісного та функціонального рішення для організації домашнього кінотеатру. Його можливості відповідають сучасним вимогам та впевнено задовольняють потреби користувачів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. PHP: Hypertext Preprocessor [електронний ресурс] -
<https://www.php.net/>
2. HTML Підручник. Початок. Уроки для початківців. W3Schools українською [електронний ресурс] -
<https://w3schoolsua.github.io/html/index.html>
3. CSS Підручник. Каскадна таблиці стилів. Стилзація сайту. Уроки для початківців. W3Schools українською [електронний ресурс] -
<https://w3schoolsua.github.io/css/index.html>
4. Сучасний підручник з JavaScript [електронний ресурс] -
<https://uk.javascript.info/>
5. MySQL [електронний ресурс] -
<https://www.mysql.com/>
6. phpMyAdmin – Documentation [електронний ресурс] -
<https://www.phpmyadmin.net/docs/>
7. CMS – WordPress.org [електронний ресурс] -
<https://wordpress.org/>
8. Requirements – WordPress.org [електронний ресурс] -
<https://wordpress.org/about/requirements/>
9. Theme Development « WordPress Codex [електронний ресурс] -
https://codex.wordpress.org/Theme_Development
10. Template Hierarchy – Theme Handbook | Developer.WordPress.org [електронний ресурс] -
<https://developer.wordpress.org/themes/basics/template-hierarchy/>
11. WordPress Plugins | WordPress.org [електронний ресурс] -
<https://wordpress.org/plugins/>
12. Shortcode API « WordPress Codex [електронний ресурс] -
https://codex.wordpress.org/Shortcode_API
13. WooCommerce Documentation – WooCommerce [електронний ресурс] -
<https://woocommerce.com/documentation/woocommerce/>

14. Template structure & Overriding templates via a theme Documentation –

WooCommerce [электронный ресурс] -

<https://woocommerce.com/document/template-structure/>