Полтавський університет економіки і торгівлі
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп’ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____Олена ОЛЬХОВСЬКА
(підпис)
«___»_____202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

«РОЗРОБКА ІНФОРМАЦІЙНОГО САЙТУ ДЛЯ МАЙСТЕРНІ "SLIM LITE" З ВИГОТОВЛЕННЯ ЕЛЕКТРОНІКИ»

зі спеціальності 122 «Комп’ютерні науки»
освітня програма «Комп’ютерні науки»
ступеня бакалавр

Виконавець роботи Ткачов Віталій Володимирович
_____«__»_____202_р.
(підпис)
Науковий керівник к.ф.-м.н.,доц.,Олексійчук Юрій Федорович
_____«__»_____202_р.
(підпис)
Рецензент

ПОЛТАВА 2025

РЕФЕРАТ

Записка: 78 с., 56 рис., 13 джерел.

ВЕБСАЙТ, РЕСУРС, ЕЛЕКТРОННІ ВИРОБИ, ІНТЕРФЕЙС, ФУНКЦІОНАЛЬНІСТЬ.

Об’єкт розробки – вебсайт для майстерні, що включає сторінки з інформацією про продукцію, доставку, оплату, гарантійне обслуговування, розділ "Про нас", а також інтерактивний розділ "Посібник" з двома режимами для різних категорій користувачів.

Мета роботи – розробити сучасний, функціональний та зручний у використанні вебсайт, який забезпечить ефективне представлення продукції майстерні, надання детальної інформації про товари, підтримку користувачів через розділ "Посібник", а також забезпечить зручну навігацію, мультимовну підтримку та оптимізацію для мобільних пристроїв.

Методи дослідження - аналіз вимог замовника та ринкових тенденцій у розробці веб-сайтів, що дозволяє визначити основні цілі та функціональні вимоги до проекту. Проведено вивчення сучасних підходів до створення інтерфейсів та функціональності вебресурсів, що забезпечує використання актуальних практик у розробці. Проектування структури сайту та інтерфейсу здійснювалося з урахуванням зручності користування, що включає створення інтуїтивно зрозумілої навігації та адаптацію дизайну для різних пристроїв.

Використано сучасні технології веб-розробки, такі як HTML, CSS, JavaScript та системи управління контентом (CMS), що забезпечує високу якість та функціональність сайту. Для забезпечення стабільної роботи ресурсу проведено тестування та оптимізацію сайту для різних пристроїв та браузерів, що включає перевірку швидкості завантаження, коректності відображення та взаємодії з користувачем. Також впроваджено мультимовну підтримку та інтерактивні елементи навігації, що робить сайт доступним для міжнародної аудиторії та покращує користувацький досвід.

Здійснена програмна реалізація веб-сайту за всіма вимогами замовника.

В результаті тестування виявлено, що продукт задовольнив очікування.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, скорочень, символів
Технічне завдання (ТЗ)	Документ, який містить детальний опис вимог до проекту, включаючи цілі, функціональність, дизайн, технічні характеристики та інші аспекти розробки.
Системи управління контентом (CMS)	Програмне забезпечення, яке дозволяє створювати, редагувати та керувати контентом на вебсайті без необхідності глибоких технічних знань. Приклади: WordPress, Joomla, Drupal
DRY (Don’t Repeat Yourself)	Принцип програмування, який полягає в уникненні повторення коду. Він спрямований на спрощення підтримки та зменшення кількості помилок.
HTML (HyperText Markup Language)	Мова розмітки, яка використовується для створення структури вебсторінок. Вона визначає розташування тексту, зображень, посилань та інших елементів на сторінці.
CSS (Cascading Style Sheets)	Мова стилів, яка використовується для опису зовнішнього вигляду вебсторінок, написаних на HTML. Відповідає за кольори, шрифти, розміри та інші візуальні аспекти.
Вебсайт	Це сукупність веб-сторінок, оформлених в одному стилі та об'єднаних спільною концепцією.

Рантайм (run,time)	Середовище виконання, яке забезпечує роботу програмного коду, інтерпретує його або компілює в машинний код у режимі реального часу. Рантайм включає бібліотеки, інструменти та ресурси, необхідні для виконання програми
DevOps	Набір практик, які поєднують розробку програмного забезпечення (Development) та його експлуатацію (Operations). DevOps спрямований на автоматизацію процесів, покращення співпраці між командами та прискорення випуску продуктів.
JVM (Java Virtual Machine)	Віртуальна машина, яка дозволяє виконувати Java-код на різних платформах. Вона інтерпретує байт-код у машинний код, що забезпечує кросплатформеність Java-додатків.

ВСТУП

У сучасному світі вебсайти стали невід'ємною частиною бізнесу, оскільки вони є основним інструментом для представлення продуктів та послуг, взаємодії з клієнтами та підвищення рівня довіри до бренду. Для майстерні, яка спеціалізується на виробництві електронних виробів, створення інформаційного вебсайту є важливим кроком для залучення нових клієнтів, надання їм зручного доступу до інформації про продукцію, а також для підтримки користувачів через розділ "Посібник". Крім того, підтримка кількох мов та оптимізація для мобільних пристроїв роблять сайт доступним для широкої аудиторії, що є особливо актуальним у умовах глобалізації та зростання вимог до зручності користування вебресурсами.

Метою роботи є розробка сучасного, функціонального та зручного у використанні вебсайту для майстерні, що забезпечить ефективне представлення продукції, надання детальної інформації про товари, підтримку користувачів через розділ "Посібник", а також забезпечення зручної навігації та мультимовної підтримки.

Об'єктом дослідження є процес створення сайту для майстерні, що включає розробку інтерфейсу, функціональних модулів, підтримку кількох мов та оптимізацію для різних пристроїв.

Предметом є методи та підходи до створення інформаційного сайту, що забезпечує зручну навігацію, ефективне представлення продукції, підтримку користувачів та мультимовний інтерфейс.

Результатом проєкту є готовий вебсайт, який відповідає вимогам замовника, забезпечує зручний доступ до інформації про продукцію, має розділ "Посібник" з двома режимами для різних категорій користувачів, підтримує кілька мов та оптимізований для використання на мобільних пристроях. Сайт також буде включати всі необхідні сторінки, такі як "Доставка та оплата", "Гарантійне обслуговування", "Про нас", а також

інтерактивні елементи навігації та підвальну секцію з додатковою інформацією.

Оформлення роботи виконано відопвідно до методичних рекомендацій [5]. Результати виконання роботи апробовані на XLVIII Міжнародній науковій студентській конференції[6]. У процесі дослідження розробки вебсайту було проаналізовано статистичні дослідження [1-3] та використано літературу, що допомогла більш змістовно вивчити процес

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

Майстерня "Slim Lite" спеціалізується на розробці та виготовленні трекерів віртуальної реальності SlimeVR. Це пристрої, які дозволяють користувачам відслідковувати рухи тіла у віртуальному просторі, забезпечуючи повніше занурення у VR-досвід. Трекери створюються вручну, із застосуванням паяльної станції та 3D-принтера, що дозволяє досягати високої точності і гнучкості в індивідуальних замовленнях. На рисунку 1.1 зображено компоненти, які виготовляє майстерня

Компоненти для виробництва закуповуються як в Україні, так і в Китаї, що забезпечує оптимальне співвідношення ціни та якості. Майстерня працює вже понад 3 роки, постійно вдосконалюючи як технічну базу, так і самі пристрої, з урахуванням зворотного зв’язку від користувачів.

Продаж продукції здійснюється через платформу OLX, однак її функціональність обмежена — не можна надати повну інформацію про продукт, який є технічно складним та потребує попереднього ознайомлення.

Водночас у мережі бракує україномовних та загальнодоступних матеріалів про встановлення, використання і налаштування SlimeVR-трекерів.

У зв’язку з цим було прийнято рішення замовити створення власного веб-сайту, який слугуватиме централізованим інформаційним ресурсом. Його мета — надати покупцям всю необхідну інформацію в одному місці: від опису характеристик пристроїв до інструкцій з експлуатації ї поширених запитань. Це дозволить суттєво підвищити рішучість потенційних клієнтів щодо придбання продукції, адже завдяки доступності повної та зрозумілої інформації їм не доведеться витрачати час на особисте спілкування з продавцем задля отримання відповідей на базові запитання. Таким чином, сайт спростить процес ознайомлення і зменшить бар’єри для ухвалення рішення про покупку.

У рамках поставленого завдання було отримано технічне замовлення від власника майстерні на розробку вебсайту, що виконує функцію інформаційного ресурсу. Основною метою проєкту стало створення платформи для представлення асортименту виробів із можливістю перегляду назви товару, його ціни, мініатюри, а також доступом до розширеної інформації на окремій сторінці кожного виробу.

Передбачалося також розробити додаткові інформаційні сторінки, зокрема: про доставку та оплату, умови гарантійного обслуговування, а також сторінку з інформацією про саму майстерню ("Про нас").

Особливу увагу було приділено створенню розділу «Посібник», що повинен містити тематичні матеріали щодо використання, налаштування та модернізації електронних виробів, виготовлених у майстерні. Задля зручності користувачів цей розділ було вирішено реалізувати з підтримкою двох режимів перегляду: «спрощеного» — для новачків, та «просунутого» — для досвідченіших клієнтів.

Згідно з технічними вимогами, головна сторінка сайту має включати логотип, назву майстерні, інтерфейсну навігацію за розділами, а також зовнішні посилання.

Меню «Наша продукція» реалізується як інтерактивний елемент, що при взаємодії з користувачем розгортає компактну стрічку з товарами й автоматично згортається при втраті фокусу.

Сайт повинен підтримувати мультимовність — українську та англійську мови. Елемент перемикання мов розміщується в структурі головного меню.

Нижня частина сайту (футер) включає логотип, назву, інформацію про авторське право, відомості про розробника, а також посилання на сторінки «Доставка та оплата», «Гарантія», «Про нас» і соціальні мережі у вигляді іконок.

Проєкт також орієнтовано на кросплатформенність, тому адаптивність інтерфейсу до мобільних пристроїв є обов’язковою умовою, із можливістю гнучкого коригування елементів дизайну та функціональності за потреби.

На рисунку 1.14 Зображено можливість мобільної адаптивності сайту у мобільній версії за допомогою перегляду в браузері.

Потрібно врахувати, що на сучасному етапі розвитку цифрових технологій наявність онлайн-платформи стала критично важливою для ефективної діяльності суб’єктів малого підприємництва. Незважаючи на стрімкий розвиток інтернет-інфраструктури, значна частина малих бізнесів в Україні досі стикається з численними труднощами, пов’язаними із забезпеченням стабільної та якісної цифрової присутності.

Однією з ключових проблем є обмеженість фінансових ресурсів, яка ускладнює можливість замовлення повноцінної розробки сайту, хостингу, технічної підтримки та просування ресурсу. Багато підприємців змушені вдаватися до самостійного створення вебресурсів або використовувати безкоштовні платформи, що суттєво обмежує функціональні можливості та ускладнює інтеграцію з платіжними системами, CRM-сервісами тощо.

Ще однією суттєвою проблемою є брак цифрової грамотності серед власників малих бізнесів. Нерідко представники цієї категорії мають поверхневе уявлення про SEO-оптимізацію, UX/UI-дизайн, адаптивність інтерфейсу або принципи безпечної обробки персональних даних. Це, у свою чергу, знижує конкурентоспроможність цифрових продуктів та ускладнює залучення нових клієнтів через онлайн-канали.

Крім того, варто відзначити, що споживчі очікування щодо дизайну, швидкодії та функціональності сайтів постійно зростають. Відсутність адаптації до мобільних пристроїв, повільне завантаження сторінок або неінтуїтивна навігація можуть негативно вплинути на довіру до бренду та скоротити конверсію.

РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Інструменти фронтенд та бекенд розробки

Невід'ємною складовою реалізації будь-якого веборієнтованого проєкту є етап проєктування, що передбачає як концептуальне планування структури сайту, так і обґрунтований вибір середовища розробки. Вибір інструментів і технологій безпосередньо впливає на ефективність процесу розробки, швидкість тестування, зручність командної роботи та подальшу підтримку продукту.

При виборі середовища для реалізації проєкту було враховано низку ключових критеріїв: підтримка сучасних вебтехнологій, зручність роботи з кодом, можливість гнучкої інтеграції з іншими інструментами, а також забезпечення умов для колективної розробки. На рівні клієнтської частини (frontend) було обрано використання HTML, CSS та JavaScript як базових технологій, що дозволяють реалізувати адаптивну, інтерактивну та кросбраузерну верстку. Для створення динамічних компонентів і оптимізації продуктивності інтерактивних елементів застосовувалися сучасні JavaScript-фреймворки.

Рисунок 2.1 – Логотипи мов HTML, CSS та JavaScript

На рисунку 2.1 зображено логотипи мов HTML, CSS та JavaScript.

Серверна частина (backend) реалізована з використанням Node.js — платформи, що забезпечує побудову масштабованих та продуктивних вебрішень завдяки асинхронній обробці запитів і модульному підходу.

В якості основного середовища розробки обрано Visual Studio Code, що забезпечує розширену підтримку мов програмування, систему підказок і автодоповнення, інтеграцію з системами контролю версій та широкий спектр розширень, які підвищують ефективність розробника.

Контроль версій та організація командної роботи реалізуються за допомогою системи Git із використанням платформи GitHub. Це дозволяє вести повну історію змін у коді, спрощує виявлення та усунення помилок, а також забезпечує збереження та синхронізацію репозиторіїв між учасниками проєкту.

Рисунок 2.2 – Логотип платформи «GitHub»

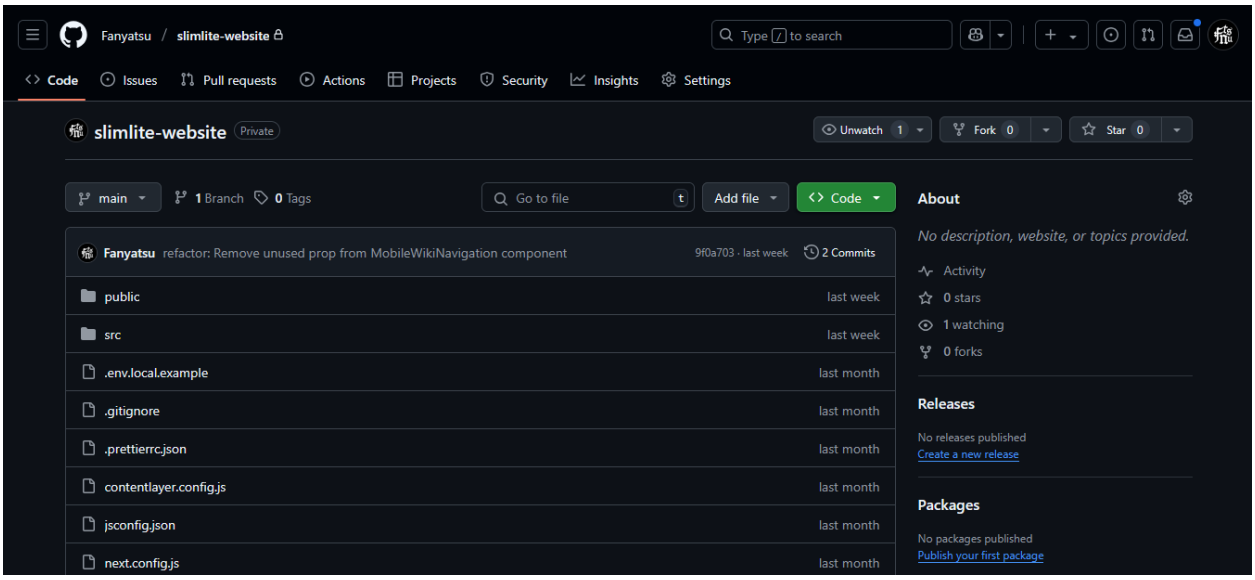


Рисунок 2.3 – Сторінка GitHub з вихідним кодом вебсайту майстерні SlimLite

Окрему увагу у процесі розробки було приділено тестуванню, що є критичним для перевірки стабільності та функціональності вебзастосунку. Для автоматизованої перевірки окремих компонентів, функцій і логіки сайту було застосовано такі інструменти, як Jest, Selenium та Cypress. Їх використання дозволило своєчасно виявляти та усувати помилки на ранніх етапах розробки, що позитивно вплинуло на якість фінального продукту.

Оглянемо використані інструменти:

1. Visual Studio Code

Visual Studio Code – це текстовий редактор коду, який став популярною заміною масивним середовищам розробки (IDE). Visual Studio Code дозволяє розробляти проекти практично будь-якою популярною мовою програмування, має модульну систему у вигляді плагінів, установка яких розширює можливості редактора. Використання цього текстового редактора дозволило налаштувати середовище під себе, заощадити місце на жорсткому диску, прискорити розробку за рахунок високої швидкості роботи, а також розширити можливості розробки завдяки інструментарію, що додається плагінами.

Рисунок 2.4 – Логотип текстового редактора «Visual Studio Code»

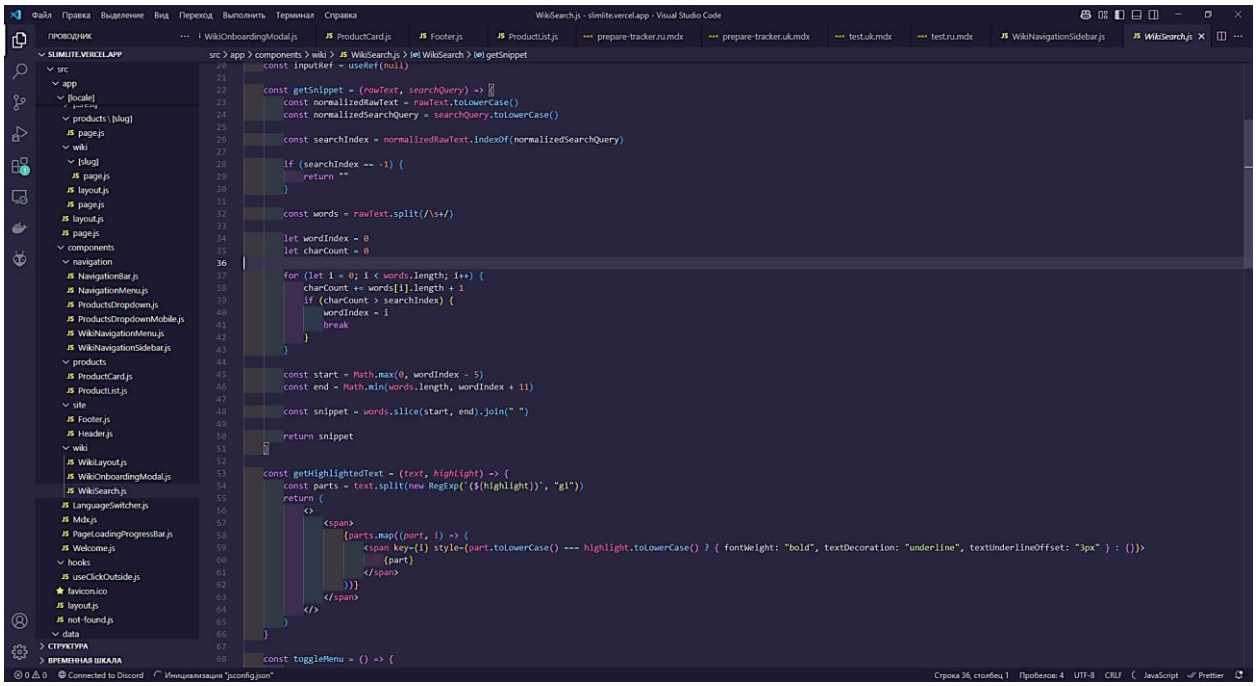


Рисунок 2.5 – Інтерфейс текстового редактора «Visual Studio Code» (1)

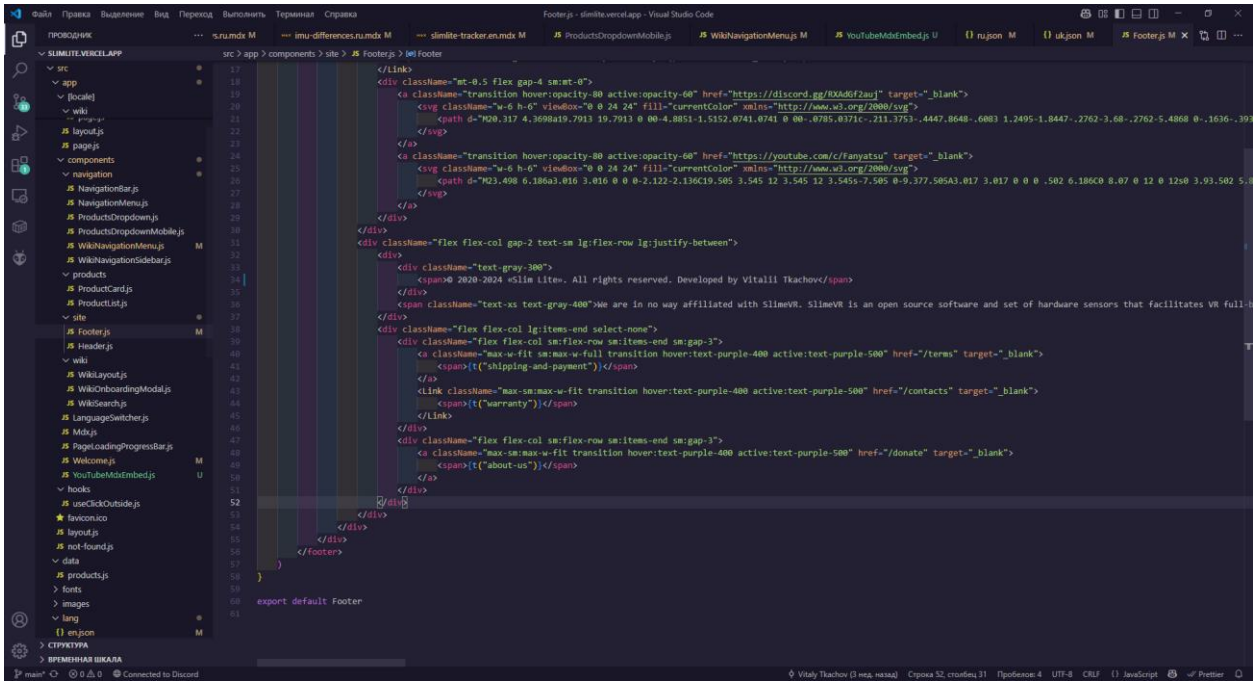


Рисунок 2.6 – Інтерфейс текстового редактора «Visual Studio Code» (2)

2. NodeJS

NodeJS – це JavaScript рантайм, який використовується майже в кожному JavaScript проекті. Дозволяє запускати та підтримувати в роботі програмний код та значно розширює можливості бекенду, що зробило його чудовим

вибором для хостингу та тестування цього проєкту. До його складу входить пакетний менеджер NPM, за допомогою якого в проєкт були швидко встановлені усі необхідні JavaScript бібліотеки.

Рисунок 2.7 – Логотип JavaScript рантайму «Node.js»

На рисунку 2.7 зображено логотип JavaScript рантайму «Node.js».

3. React

React - це популярна JavaScript бібліотека, що дозволяє створювати складні користувацькі інтерфейси вебсайтів. Включає багато технологій, частину з яких потрібно встановлювати окремими пакетами, так як їх функціонал занадто великий, щоб вмістити його в базовій комплектації.

Рисунок 2.8 – Логотип JavaScript бібліотеки «React»

На рисунку 2.8 зображено Логотип JavaScript бібліотеки «React». React був обраний саме тому, що вміє запам'ятовувати стан компонентів між відображеннями, і швидко відображати на екрані всі зміни зовнішнього вигляду компонентів. Приклад використання:

import { useState, useRef } from "react"

```
const ProductsDropdownMobile = () => {
  const [isDropdownOpen, setIsDropdownOpen] = useState(false)

  const toggleDropdown = () => {
    setIsDropdownOpen(!isDropdownOpen)
  }
...
```

4. Tailwind CSS

Tailwind CSS - це CSS фреймворк. На відміну від традиційного CSS та його препроцесорів, стилі вже заздалегідь згенеровані у вигляді готових CSS класів, що дозволяють прописувати їх прямо в HTML-тезі class (або JSX-тезі className), видаляючи потребу постійного переміщення між файлом розмітки і файлом опису стилів. Якщо заздалегідь підготовлених класів не вистачає, за розробником залишається можливість створювати власні. Саме завдяки можливості швидко працювати зі стилями та створювати свої, Tailwind CSS був включений у проєкт:

styles.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

HTML код

```
<span className="text-xs text-zinc-300">
  ...
</span>
```

Рисунок 2.9 – Логотип CSS фреймворка «Tailwind CSS»

На рисунку 2.9 зображено Логотип CSS фреймворка «Tailwind CSS».

5. NextJS

NextJS - це один із найвідоміших JavaScript фреймворків. Постійно вдосконалюється та вводить інноваційні методи розробки вебсайтів, так і роботи вебсайтів. Можливості фреймворку дозволили розробити повністю самостійний веб-додаток, який може працювати або як повністю клієнтський, або як повністю серверний, або поєднуючи обидва підходи. В проєкті використано обидва. Сторінки представлені як статичні (заздалегідь відмальованими), так і динамічні. Фреймворк також підтримує можливість створення вбудованого API. Постачається разом з React та Tailwind CSS, які якраз і планувалося включати в проєкт на етапі проектування.

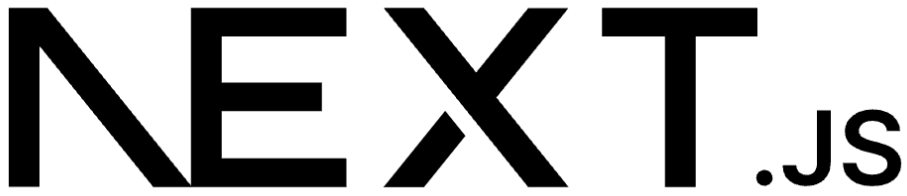


Рисунок 2.10 – Логотип JavaScript фреймворку «Next.js»

На рисунку 2.10 зображено вигляд логотипу JavaScript фреймворку «Next.js».

Приклад використання NextJS:

```
"scripts": {
  "dev": "next dev",
  "build": "next build",
  "start": "next start",
  "lint": "next lint"
}
```

6. ContentLayer

ContentLayer - це пакет NPM та набір інструментів для роботи з контентом, що відображається на вебсайтах. Його особливість від інших наборів інструментів для роботи з контентом в тому, що він використовує в основі мову розмітки "markdown", відомий своїми абревіатурами MD і MDX (останній дозволяє використовувати можливості JavaScript у розмітці. Обидва формати підтримуються contentlayer). ContentLayer переводить вміст markdown у звичний HTML, що дозволяє його стилізувати та відображати на вебсайті. Цю можливість часто використовують для створення блогів, статей, вікі-розділів. Має підтримку плагінів для розширення можливостей наповнення контентом, таких як вбудовування відео, зміна синтаксису, підтримку HTML-тегів, яких немає в чистому makdown та інше. В цьому проєкті ContentLayer відповідає за відображення вмісту сторінок з детальною інформацією про товари та сторінки розділу посібника.

Рисунок 2.11 – Логотип «ContentLayer»

На рисунку 2.11 зображено логотип «ContentLayer».

Приклад використання ContentLayer:

```
const WikiPost = defineDocumentType(() => ({
  name: "WikiPost",
  filePathPattern: `wiki/**/*.mdx`,
  contentType: "mdx",
  fields: {
    id: { type: "number", required: true },
```

```
    mode: { type: "number", required: false, default: 0 },
    title: { type: "string", required: true },
    description: { type: "string", required: true },
    date: { type: "date", required: true },
    category: { type: "string", required: false, default: null },
    advanced_version: { type: "string", required: false, default: null },
  },
  computedFields: {
    slug: {
      type: "string",
      resolve: computeSlugField,
    },
    locale: {
      type: "string",
      resolve: computeLocaleField,
    },
  },
}))
```

7. NextIntl

NextIntl - це пакет NPM для NextJS, що додав можливість реалізувати підтримку відображення елементів інтерфейсу програми різними мовами. Має вбудовані форматери для зміни формату відображення різних величин, що відрізняються в різних країнах, такі як дата і час, кількості, одиниць виміру, валюти та інше. За допомогою NextIntl було реалізоване автоматичне визначення мови відвідувача сайту, перемикання на іншу мову, локалізація як контенту, згенерованого сервером, так і браузером. На відміну від вбудованого інструментарію з впровадження кількох мов у NextJS, має значно більший функціонал і виправляє недоліки вбудованого варіанту.

Рисунок 2.12 – Логотип «NextIntl»

На рисунку 2.12 зображено логотип «NextIntl».

Приклад використання у проєкті:

```
import { getRequestConfig } from "next-intl/server"
import { notFound } from "next/navigation"
import { locales } from "@/navigation"

export default getRequestConfig(async ({ locale }) => {
  if (!locales.includes(locale)) {
    notFound()
  }

  return {
    timeZone: "Etc/UTC",
    messages: (await import(`@/lang/${locale}.json`)).default,
  }
})
```

8. Prettier

Prettier - це просунутий форматтер для коду. Вміє форматувати JavaScript, HTML, CSS, markdown, YAML, GraphQL, Java, PHP, Ruby, Rust, XML та інші мови. Інтегрується з безліччю текстових редакторів, у тому числі Visual Studio Code, що і було зроблено. Форматер коду знадобився для автоматичного виправлення синтаксису коду, патернів написання та

дотримання інших загальноприйнятих правил написання коду. Поведінку Prettier можна налаштувати на свій смак. Проєкт має у своєму складі файл налаштувань, до якого були задані параметри, що дозволили використовувати Prettier для поставленої мети:

```
{
   "plugins": ["prettier-plugin-tailwindcss"],
   "trailingComma": "es5",
   "tabWidth": 4,
   "printWidth": 600,
   "semi": false
}
```

Рисунок 2.13 – Логотип форматтеру «Prettier»

На рисунку 2.13 зображено зовнішній вигляд використовуваного форматтеру для коду.

2.2. Порівняння технологій розробки вебсайтів

У сучасному світі існує безліч сайтів, які слугують людям з абсолютно різною метою. Наразі, саме вебсайт є невід'ємною частною, наприклад, для бізнесу, як і локального, так і для великих компаній.

З моменту появи вебресурсів спостерігалася необхідність їх диференціації за функціональним призначенням. Однак усталеної та загальновизнаної класифікації не було розроблено, що зумовило певні термінологічні труднощі. Зокрема, деякі вебсайти слугували джерелами новинної інформації, інші – пошуковими платформами, а також існували індивідуальні сторінки користувачів.

Різні вебстудії пропонували власні підходи до класифікації вебсайтів, що призводило до варіативності термінології. Внаслідок цього один і той самий за функціональними характеристиками ресурс міг мати різні найменування, такі як «вебсайт», «презентаційний вебсайт», «фірмовий сайт» або «корпоративний сайт». Давайте розглянемо деякі з підвидів вебсайтів:

1. Сайт візитка

Сайт, призначений для окремої особи або фірми, зазвичай містить візуальне представлення суб'єкта, коротку інформацію про нього, контактні дані для зв'язку та відомості про сферу діяльності. Часто сайти-візитки з часом еволюціонують у повноцінні вебсайти з розширеною структурою та численними розділами.

На рисунку 2.14 зображено сайт-візитку.

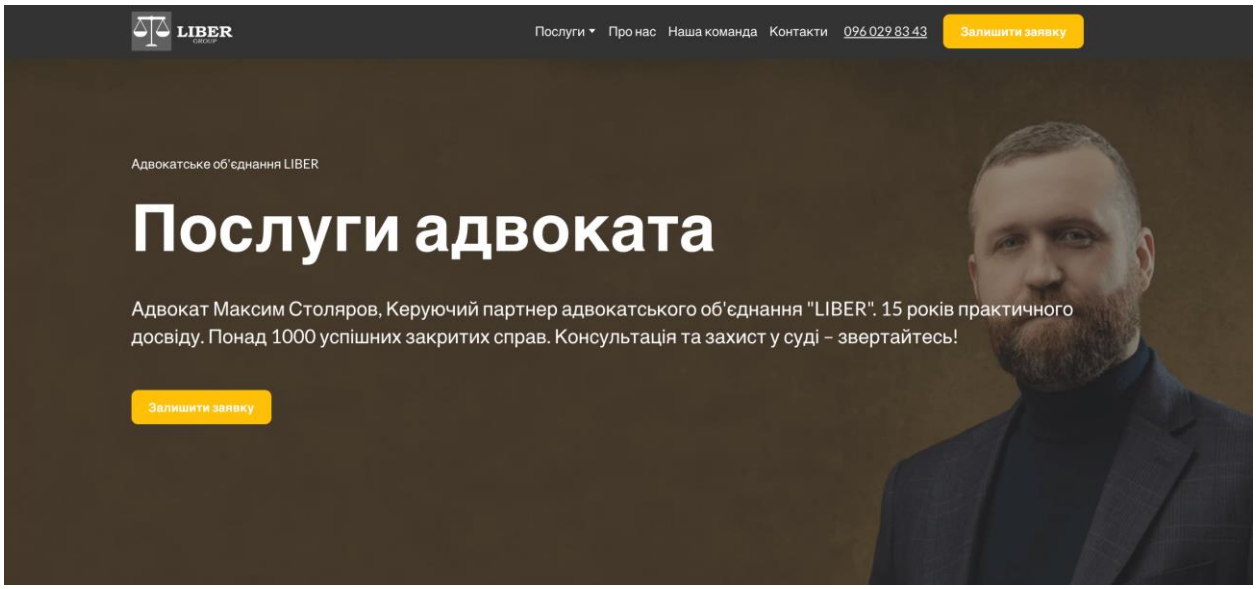


Рисунок 2.14 – Сайт-візитка з адвокатських послуг

Одними з основних переваг таких сайтів є простота їх створення, можливість розміщення лише ключової інформації, популярність як альтернативи або доповнення до традиційних візиток, а також низькі вимоги до ресурсів для розміщення на сервері або в хмарних сховищах.

Проте існують і певні недоліки, зокрема можливість недостатності розміщеної інформації для деяких користувачів, а також ризик втрати унікальності через схожість дизайну та структури з іншими сайтами-візитками.

2. Корпоративний сайт

Корпоративний сайт є багатофункціональним вебресурсом, який представляє компанію в інтернеті та забезпечує комплексну інформацію про її діяльність, послуги, продукцію, історію та контакти. Такі сайти часто включають розділи для клієнтів, партнерів та інвесторів, а також можуть містити новинний блок, галереї, документи та інші елементи, що підкреслюють професійний підхід компанії.

Основною перевагою корпоративних сайтів є можливість повноцінно представити бренд, забезпечити довіру клієнтів та налагодити ефективну комунікацію з аудиторією. Однак їх створення вимагає значного часу, ресурсів

та постійного оновлення контенту, що може бути складним для невеликих компаній.

На рисунку 2.15 зображено корпоратиний сайт компанії «Microsoft».

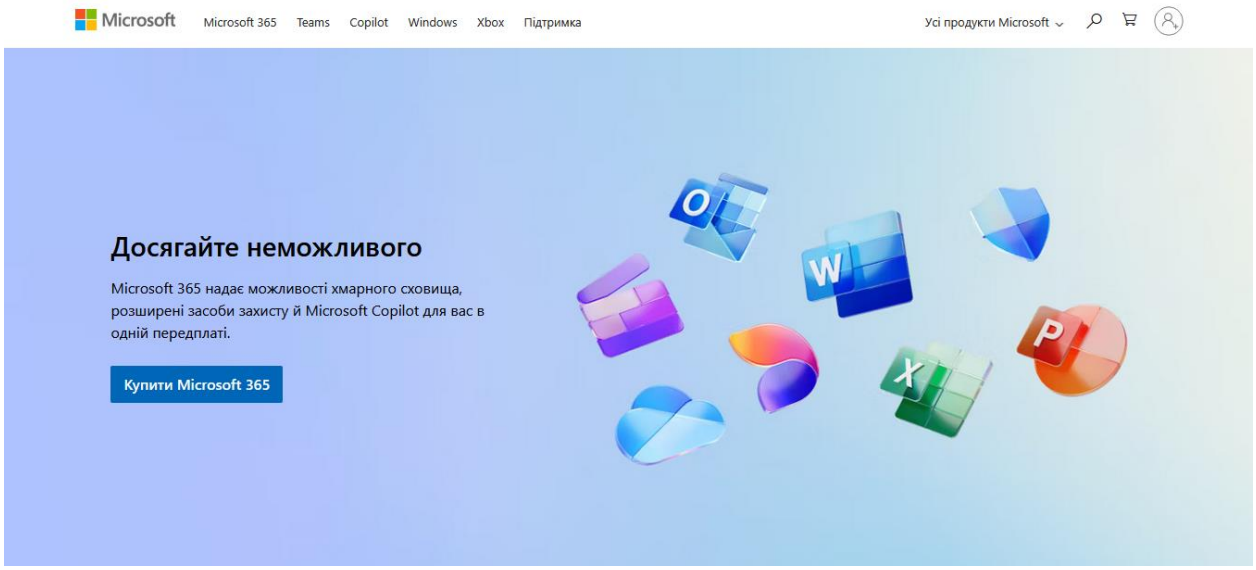


Рисунок 2.15 – Корпоративний сайт компанії «Microsoft»

3. Інтернет-магазин

Інтернет-магазин є спеціалізованим вебресурсом, основним завданням якого є продаж товарів або послуг онлайн. Він включає каталог продукції, кошик для замовлень, систему оплати та доставки, а також інструменти для управління замовленнями.

Інтернет-магазини дозволяють компаніям охопити широку аудиторію, забезпечити зручний процес покупок та автоматизувати багато бізнес-процесів. Проте їх розробка та підтримка вимагають значних фінансових вкладень, а також постійного оновлення асортименту та технічного обслуговування.

На рисунку 2.16 зображено вигляд маркетплейсу «Розетка».

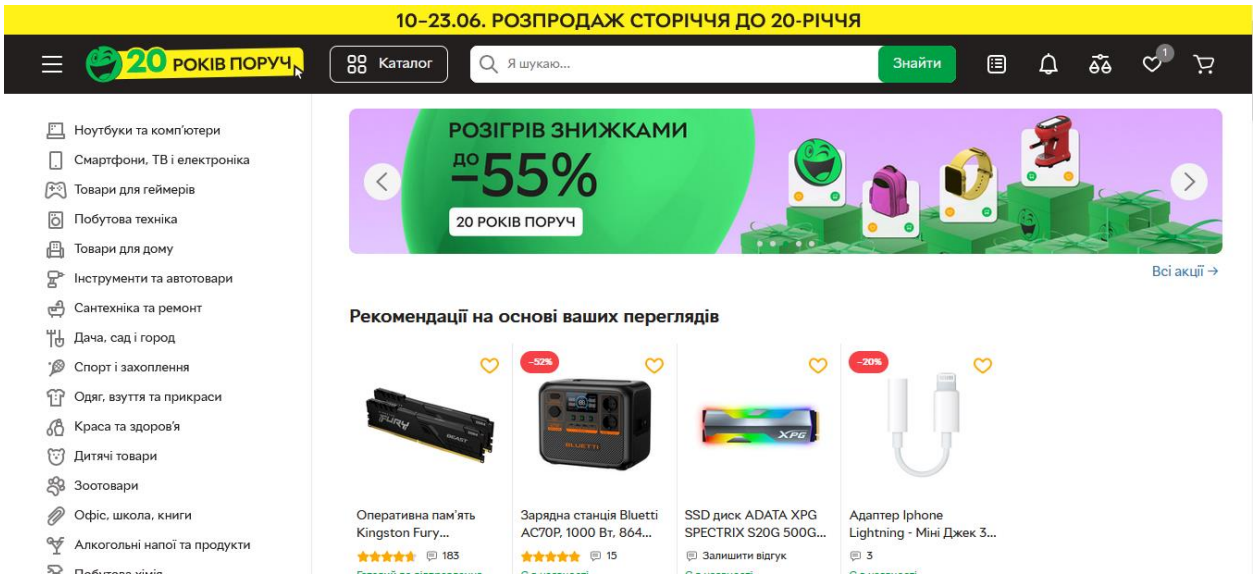


Рисунок 2.16 – Інтернет магазин «Розетка»

4. Односторінковий сайт (лендінг)

Односторінковий сайт, або лендінг, є компактним вебресурсом, основним завданням якого є презентація одного продукту, послуги або події. Він містить стислий, але ефективний контент, спрямований на залучення уваги користувача та мотивацію до певної дії, наприклад, замовлення, реєстрації або покупки.

Лендінг відрізняється простотою структури, швидкістю завантаження та зручністю для мобільних пристроїв. Однак обмежений обсяг інформації може бути недостатнім для складних продуктів або послуг, що потребують детального опису. На рисунку 2.17 проілюстровано приклад лендінг-вебсайту.

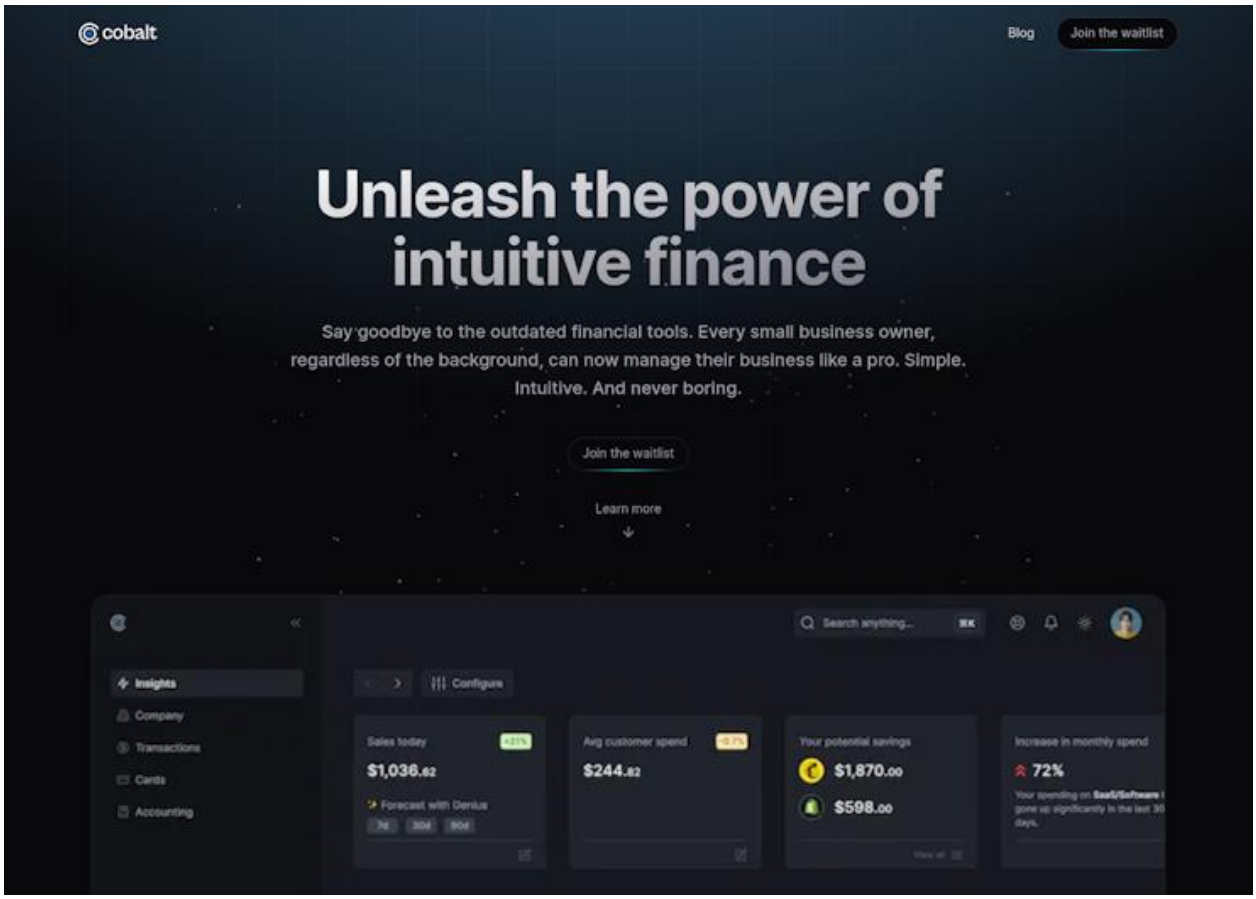


Рисунок 2.17 – Односторінковий сайт (лендінг)

5. Маркетплейс

Маркетплейс є масштабним онлайн-майданчиком, який об’єднує багато продавців та покупців на одній платформі. Він надає можливість приватним особам та компаніям розміщувати свої товари або послуги, а покупцям – вибирати з широкого асортименту.

Маркетплейси забезпечують зручний інтерфейс для пошуку, порівняння та покупок, а також інтегровані системи оплати, доставки та відгуків. Незважаючи на їх популярність, створення та підтримка маркетплейсу вимагають значних інвестицій, складних технічних рішень та постійної модерації контенту.

На рисунку 2.18 проілюстровано маркетплейс «Amazon».

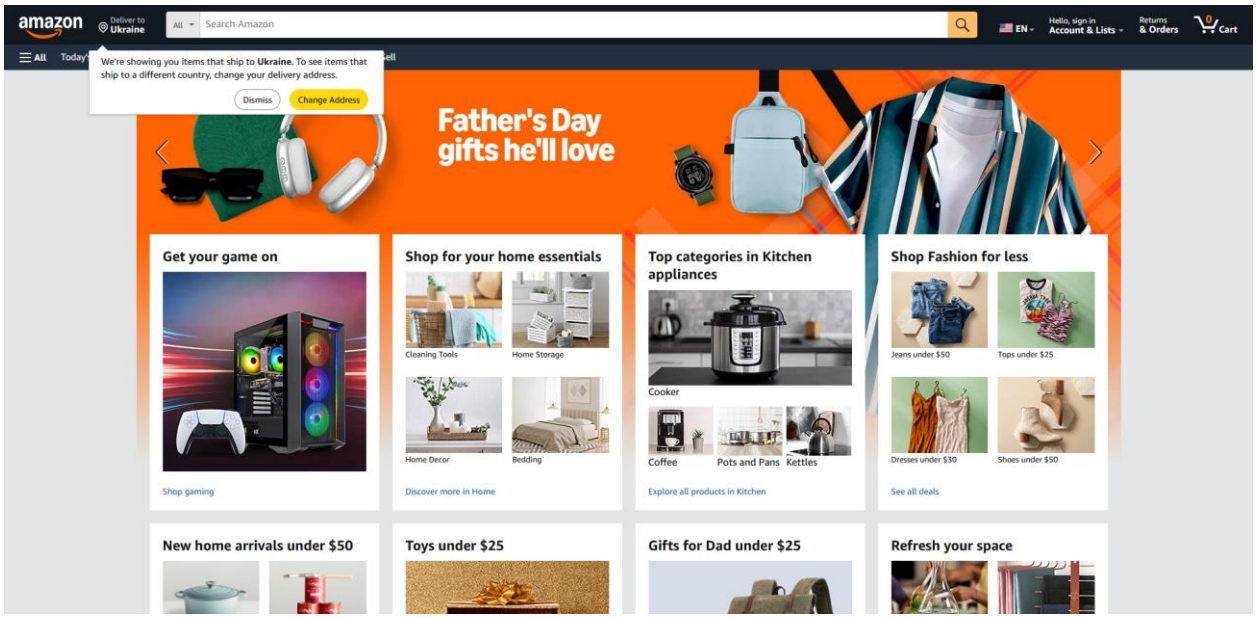


Рисунок 2.18 – Головна сторінка маркетплейсу «Amazon»

6. Інші види сайтів

До інших видів сайтів відносяться спеціалізовані ресурси, такі як блоги, форуми, соціальні мережі, освітні платформи, портфоліо тощо. Кожен з них має свої унікальні функції та призначення. Наприклад, блоги дозволяють публікувати статті та новини, форуми забезпечують спілкування користувачів, а освітні платформи надають доступ до навчальних матеріалів.

Такі сайти можуть бути як простими, так і складними за структурою, залежно від їх цілей та аудиторії. Їх перевагою є гнучкість у використанні, але вони часто вимагають постійного оновлення контенту та технічної підтримки.

Еволюція вебсайтів відображає розвиток інтернет-технологій, зміни у вимогах користувачів та трансформацію підходів до вебдизайну та функціональності. З моменту появи перших вебсайтів у 1990-х роках вони пройшли значний шлях від простих текстових сторінок до складних, інтерактивних платформ.

Протягом усієї історії Інтернету, вебсайти використовували та продовжують використовувати у своїй основі протокол HTTP (HyperText Transfer Protocol) та мову розмітки HTML (HyperText Markup Language).

Згодом з'явилася мова опису стилів CSS (Cascading Style Sheets), яка дозволила тонко налаштовувати зовнішній вигляд вебсайтів. Далі з'явилася мова JavaScript, яка дозволила вебсайтам виконувати обчислення та відображати більше видів контенту, використовуючи потужність браузера відвідувача сайту. У наш час JavaScript є такою ж основою багатьох більшої частини вебсайтів, як і вищеописані технології. Також у наш час стало популярно використовувати JavaScript і на серверній частині завдяки рантаймам, таким як NodeJS (хоча PHP продовжує лідирувати за кількістю сайтів, доступних в мережі), а також робити програми та програми, які використовують технології вебсайтів для інтерфейсу користувача завдяки технології webview, або повністю реалізовувати функціонал всього програмного забезпечення за допомогою веб технологій. Сучасні сайти проте вже не використовують JavaScript у голому вигляді, так як з'явилося безліч різних фреймворків, що значно розширюють можливості створення вебсайту, і додають інноваційні веб технології, доступні для використання прямо з фреймворку. Стало можливим поєднувати серверний та клієнтський код у межах одного проекту, інтегрувати проект з іншими проектами чи мікросервісами.

Зокрема, сучасні вебсайти характеризуються високим рівнем інтерактивності, використанням штучного інтелекту, віртуальної та доповненої реальності, а також інтеграцією з хмарними сервісами. Велику увагу приділяється швидкості завантаження, безпеці та доступності для людей з обмеженими можливостями. Також зростає популярність вебсайтів, які використовують мінімалістичний дизайн та фокусуються на користувацькому досвіді (UX).

Майбутнє вебсайтів пов’язане з подальшим розвитком технологій, таких як штучний інтелект, блокчейн, IoT (інтернет речей) та квантові обчислення. Очікується, що вебсайти стануть ще більш персоналізованими, автоматизованими та інтегрованими з різними пристроями та сервісами.

Еволюція вебсайтів демонструє, як технологічний прогрес і зміни у поведінці користувачів формують нові стандарти та підходи до створення вебресурсів.

2.3. Обґрунтування вибору інструментів для розробки

Для реалізації проєкту на стороні сервера використано Node.js - високопродуктивне середовище виконання JavaScript, яке дозволяє створювати масштабовані бекенд-рішення. Для швидкого встановлення бібліотек та фреймворків використовувався його менеджер пакетів NPM.

Фронтенд реалізовано за допомогою React - бібліотеки для створення користувацьких інтерфейсів. React дозволяє повторно використовувати компоненти, зберігати їх стан та ефективно рендерити зміни. Для стилізації використано CSS фреймворк Tailwind, який дозволяє використовувати попередньо згенеровані класи без необхідності створювати окремі CSS файли. Це значно скорочує час розробки та підвищує гнучкість у налаштуванні зовнішнього вигляду.

Для побудови архітектури веб-додатку було обрано Next.js - повноцінний фреймворк, який поєднує в собі можливості серверного рендерингу, статичної генерації та взаємодії з клієнтом. Це дало можливість створювати як динамічні, так і статичні сторінки та інтегрувати вбудований API. До складу проекту також входить ContentLayer, який обробляє контент у форматах Markdown та MDX, що є корисним для реалізації інформаційних розділів, зокрема інструкції та описів продуктів.

Для реалізації багатомовності в інтерфейсі використовується пакет NextIntl, який забезпечує локалізацію як серверного, так і клієнтського контенту з урахуванням мовних і регіональних особливостей користувача.

Пакет підтримує автоматичне визначення мови, перемикання між мовами та форматування дати, часу, чисел тощо відповідно до локальної мови.

Таким чином, налаштовуване середовище розробки включає в себе цілий набір сучасних технологій та інструментів, які взаємодіють між собою, щоб забезпечити стабільну, масштабовану і багатофункціональну веб-платформу, адаптовану до різних пристроїв і мов.

2.4. Огляд та аналіз матеріалів контенту

При розробці, основним джерелом інформації стали дані, надані замовником, зокрема описи продукції, її характеристик, цін та зображень.

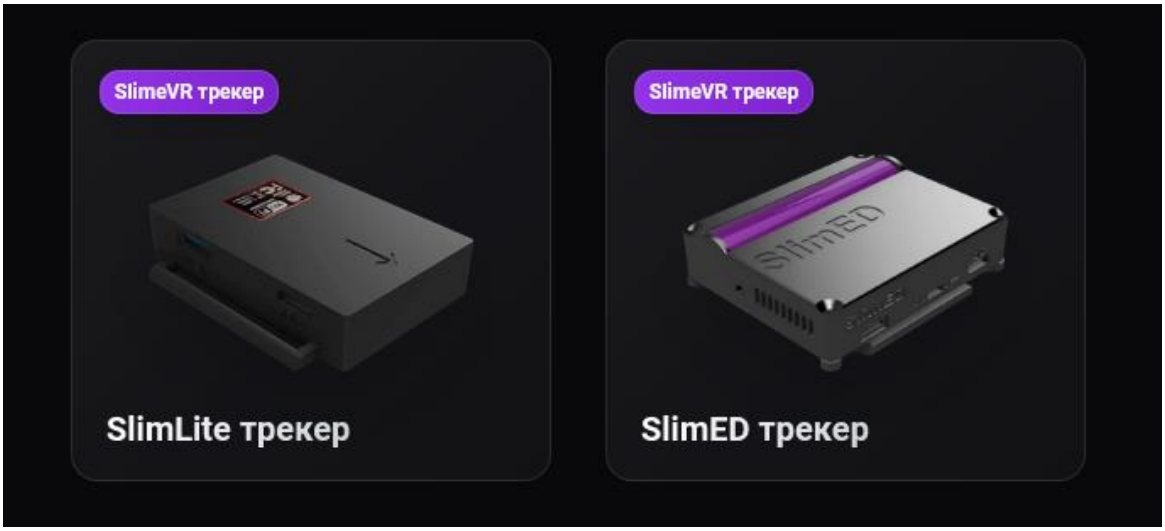


Рисунок 2.19 – Основні товари, що надає майстерня

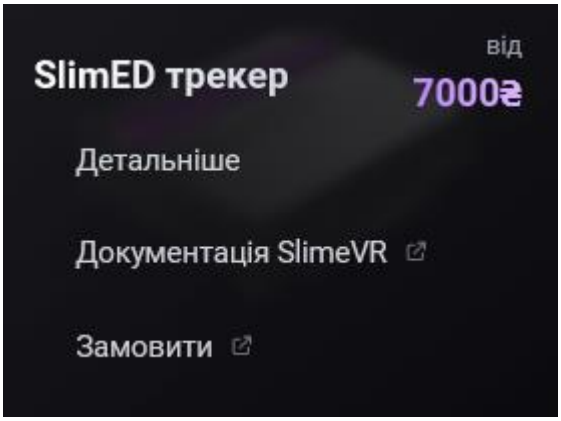


Рисунок 2.20 – Інформація та посилання в картці товару

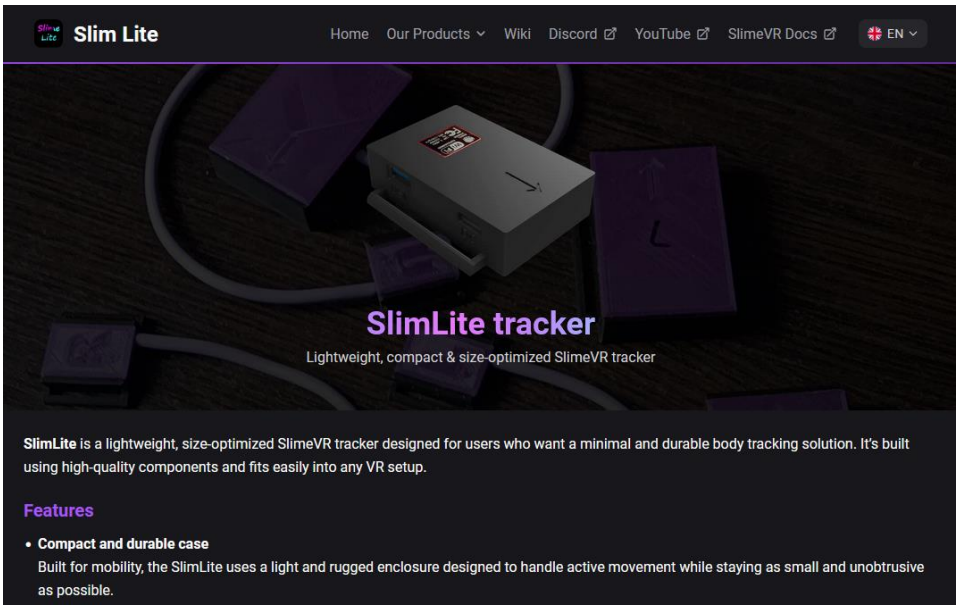


Рисунок 2.21 – Детальна інформація про продукт на окремій сторінці

На рисунках 2.19 – 2.21 зображено елементи інтерфейсу, що пов’язано з товарами, які розміщені на сайті.

Також враховано необхідність створення розділів, таких як "Доставка та оплата", "Гарантія", "Про нас" та "Посібник", які містять детальну інформацію про послуги майстерні та інструкції для користувачів.

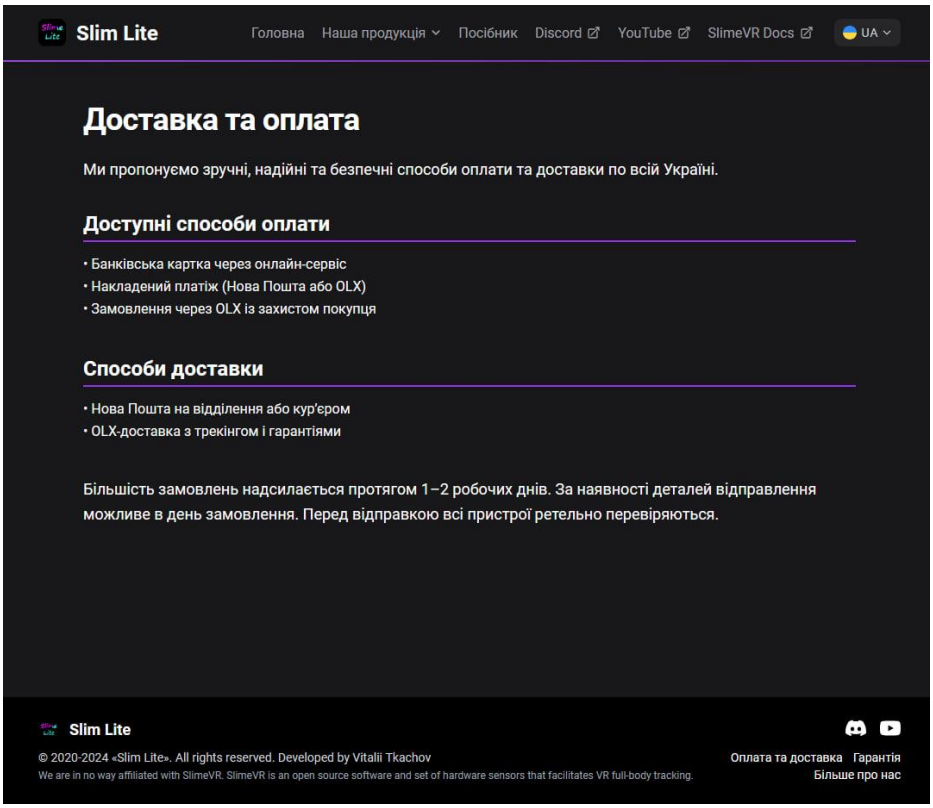


Рисунок 2.22 – Сторінка «Доставка та оплата»

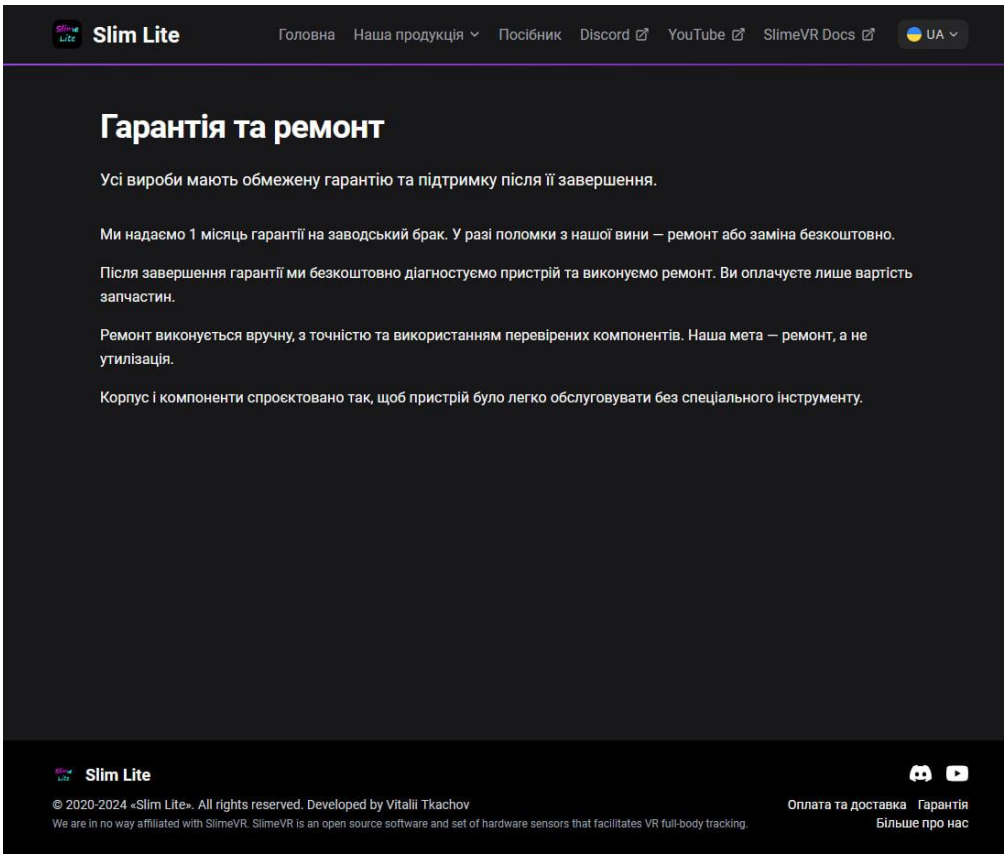


Рисунок 2.23 – Сторінка «Гарантія та ремонт»

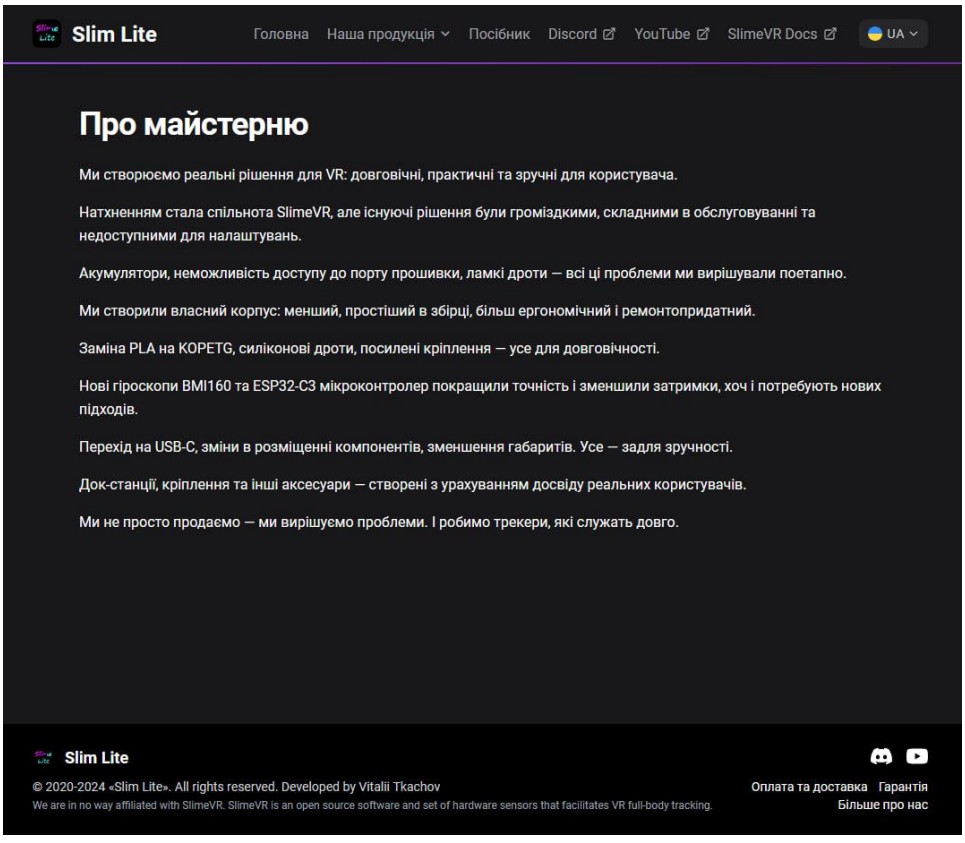


Рисунок 2.24 – Сторінка «Про майстерню» (або «Про нас»)

На рисунку 2.25 зображено футер вебсайту

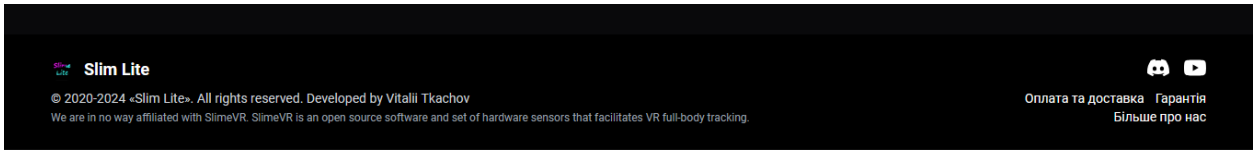


Рисунок 2.25 – Нижня частина (футер) вебсайту

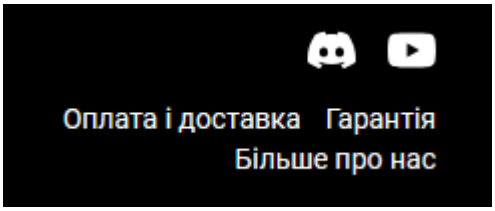


Рисунок 2.26 – Розділи вебсайту, доступні через футер

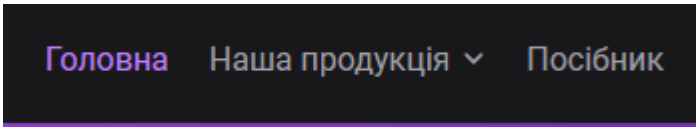


Рисунок 2.27 – Розділи вебсайту, доступні через хедер (основні розділи)

На рисунку 2.28 представлено висувне меню розділу «Наша продукція».

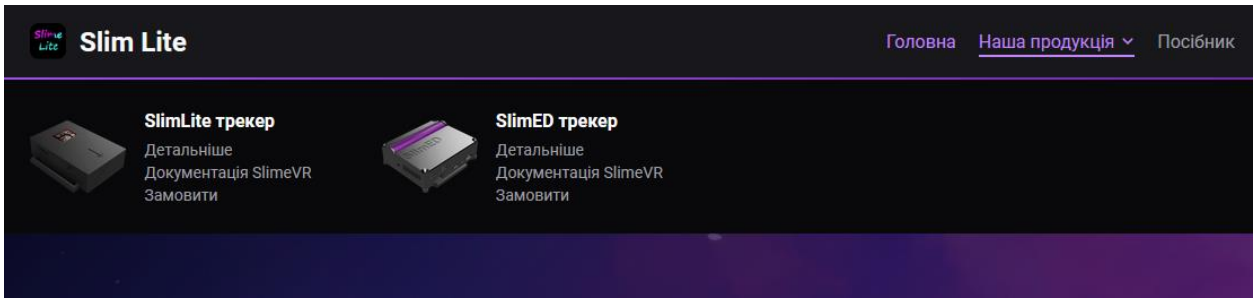


Рисунок 2.29 – Меню розділу «Наша продукція»

Також, на рисунку 2.30 продемонстровано головну сторінку, яку прийнято вважати «обличчям» вебсайту.

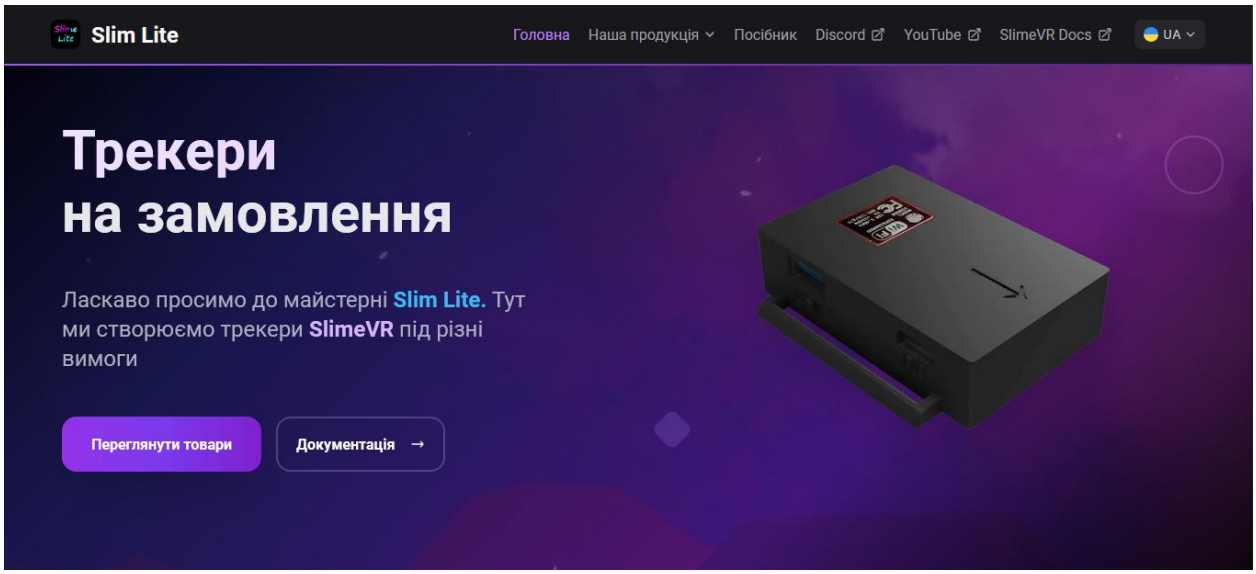


Рисунок 2.30 – Головна сторінка вебсайту

Особливу увагу приділено розділу "Посібник", який включає дві категорії статей: "спрощений" та "просунутий" режими, що забезпечують адаптацію контенту для різних рівнів підготовки користувачів.

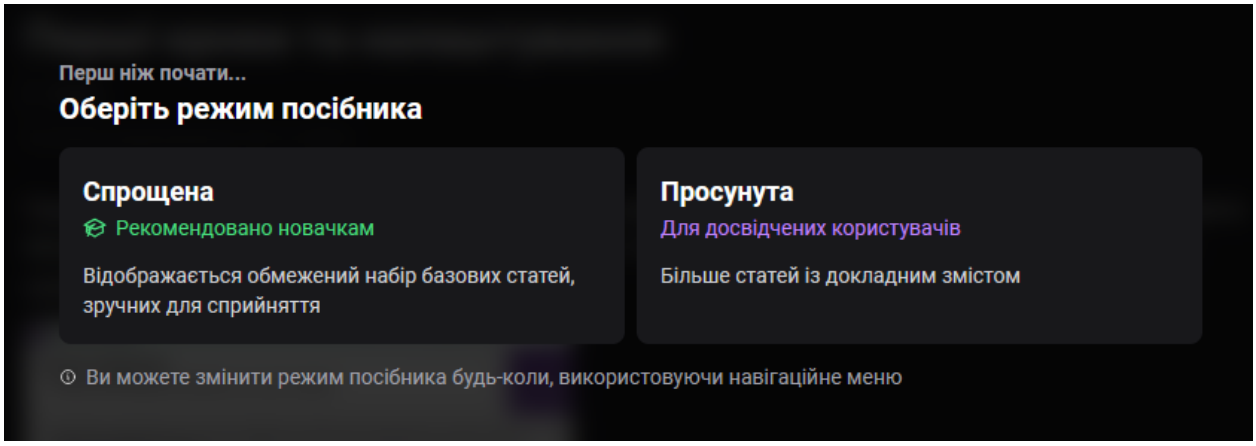


Рисунок 2.31 – Екран вибору режиму посібника при першому візиті

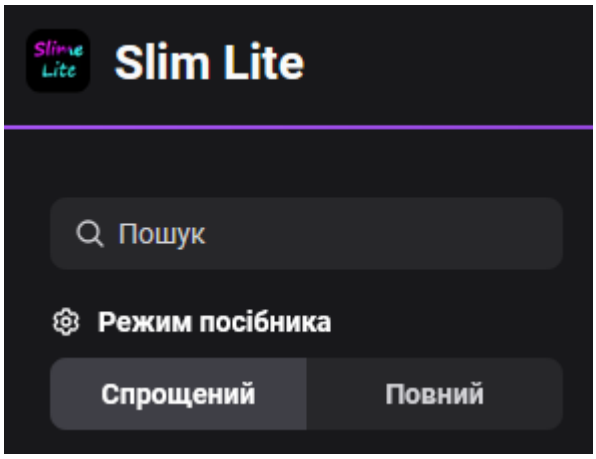


Рисунок 2.32 – Перемикач режимів розділа «Посібник»

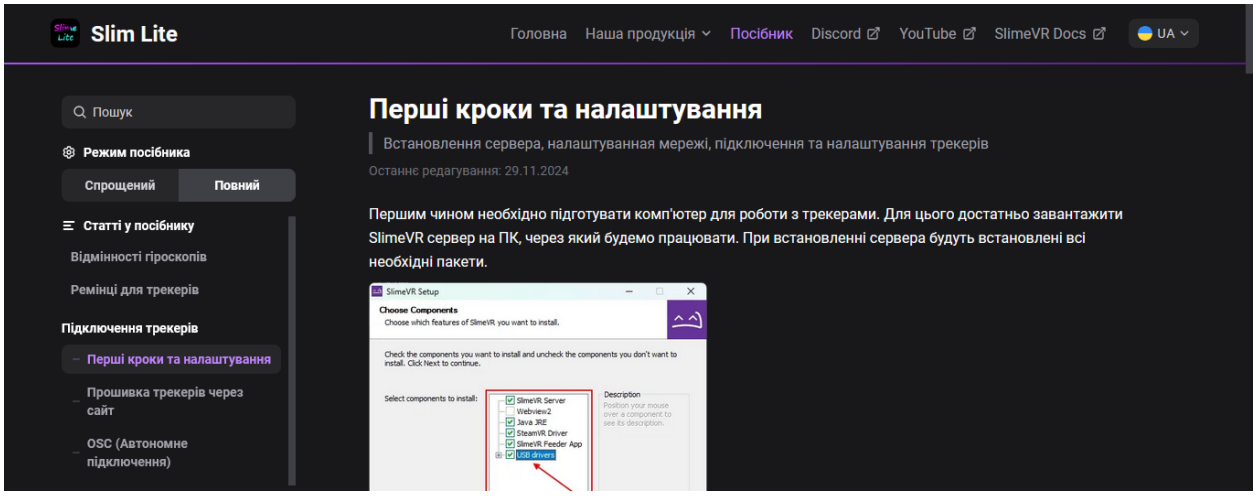


Рисунок 2.33 – Вигляд сторінок у спрощеному режимі посібника

На рисунках 2.31 - 2.33 зображено режими розділу «Посібник» та їх вибір.

Крім того, було проаналізовано необхідність підтримки кількох мов (української та англійської) для забезпечення доступності сайту для міжнародної аудиторії.

На рисунку 2.34 зображено вибір локалізації сайту

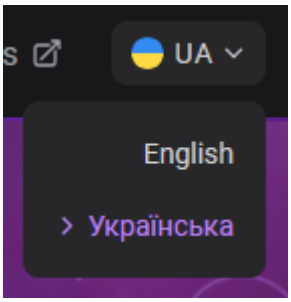


Рисунок 2.34 – Елемент «Перемикач мови» для вибору мови вебсайту

Також буде проведено аналіз візуального контенту, зокрема логотипу майстерні, зображень продукції та іконок для соціальних мереж. Враховуючи вимоги до мобільної оптимізації, було вирішено адаптувати контент для коректного відображення на різних пристроях, що включає зменшення розмірів зображень та оптимізацію текстового наповнення.

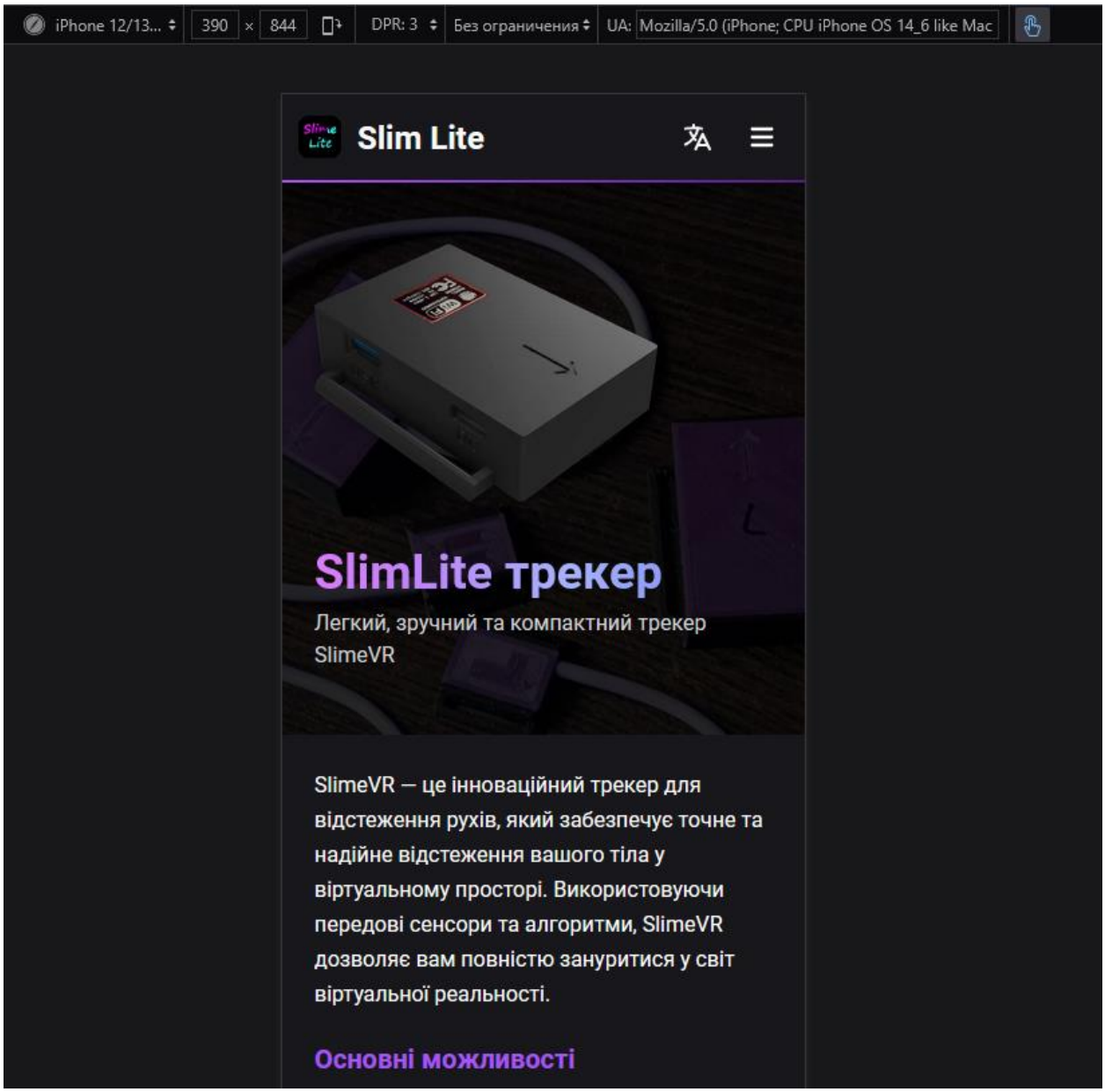


Рисунок 2.35 – Адаптивність (мобільна версія) вебсайту на малих екранах

У результаті аналізу було сформовано чітку структуру контенту, яка забезпечує зручну навігацію, доступність інформації та відповідність вимогам замовника.

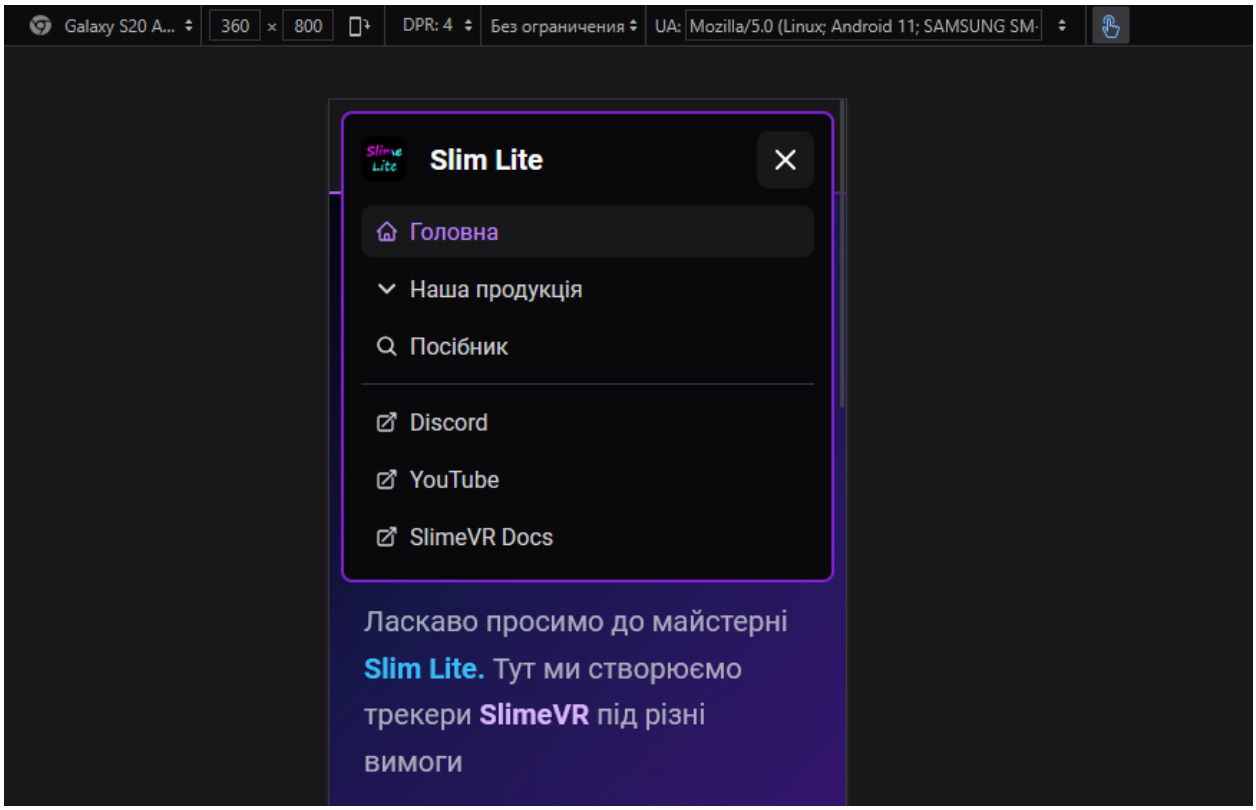


Рисунок 1.36 – Навігація в адаптивній (мобільній) версії вебсайту

Вебсайт є інформаційним ресурсом, основна мета якого – представлення продукції майстерні, а також надання клієнтам детальної інформації про вироби, умови їх експлуатації та можливості модифікації.

Сайт буде поєднувати деякі функції інтернет-магазину та інформаційного порталу, що робить його зручним для різних категорій користувачів.

Основною інноваційною особливістю є наявність розділу "Посібник", який передбачає два режими перегляду: "спрощений" і "просунутий". Це дозволяє користувачам з різним рівнем технічної підготовки знаходити необхідну інформацію в зручному для них форматі. Такий підхід підвищує доступність контенту та покращує користувацький досвід.

Актуальність створення вебсайту зумовлена зростаючим попитом на індивідуальні та кастомізовані електронні вироби. Користувачі все частіше шукають не просто товари, а розширену інформацію щодо їх налаштування, вдосконалення та використання. Додатково, важливими аспектами є

підтримка мультимовності, адаптивний дизайн для мобільних пристроїв та інтерактивні елементи навігації, що забезпечують зручність використання.

Сучасні тенденції веброзробки вимагають не лише якісного контенту, а й високого рівня інтерактивності та персоналізації. Вебсайт враховує ці вимоги, що робить його корисним та зручним для кліентів.

РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Архітектура вебсайту

Під архітектурою вебсайту розуміється логічна й технічна структура, яка визначає взаємозв’язок між його складовими частинами, способи зберігання та обробки інформації, а також принципи взаємодії користувача з інтерфейсом. Грамотно спроєктована архітектура забезпечує зручність користування сайтом, його масштабованість, швидкодію та можливість подальшого розширення функціональності.

У контексті розробки вебресурсу для малого бізнесу — майстерні, що спеціалізується на виготовленні електронних виробів, — було визначено необхідність реалізації інформаційної архітектури, яка дозволяє інтуїтивно орієнтуватися в структурі сайту, легко знаходити потрібну інформацію, а також оперативно керувати контентом.

Вебсайт умовно поділяється на такі основні функціональні блоки:

1. Головна сторінка — стартова точка взаємодії з користувачем, що містить логотип, назву підприємства, головне меню з елементами навігації, інтерактивний блок «Наша продукція», мовний перемикач;
2. Каталог товарів — динамічний блок, в якому реалізовано відображення товарів у вигляді карток з фото, назвою та ціною. Кожна картка містить

посилання на індивідуальну сторінку товару, де подається повний опис, технічні характеристики та зображення;

3. Розділ "Посібник" — тематична база знань, розділена на два рівні складності: «спрощений» і «просунутий», що орієнтовані на різні категорії користувачів. Даний розділ реалізується як окрема структура з навігаційним меню всередині;
4. Сторінки інформаційного характеру — «Про нас», «Доставка та оплата», «Гарантії» — статичні розділи, які забезпечують користувача всією необхідною інформацією для прийняття рішення про покупку;
5. Підвальна частина (footer) — структурний елемент, що повторює основні реквізити: логотип, назву, посилання на сторінки, коротку інформацію про розробника та іконки соціальних мереж.

З технічного погляду архітектура сайту розроблялася з урахуванням принципів модульності та адаптивності, що дозволяє легко масштабувати проект та вносити функціональні зміни без істотного перегляду загальної структури. Адаптивна верстка реалізовується на основі медіа-запитів CSS, що забезпечує коректне відображення контенту на різних типах пристроїв.

Усі ключові компоненти сайту реалізуються з урахуванням можливості мультимовної підтримки. Структура зберігання мовних файлів передбачає гнучку інтеграцію нових мов без змін до основного коду.

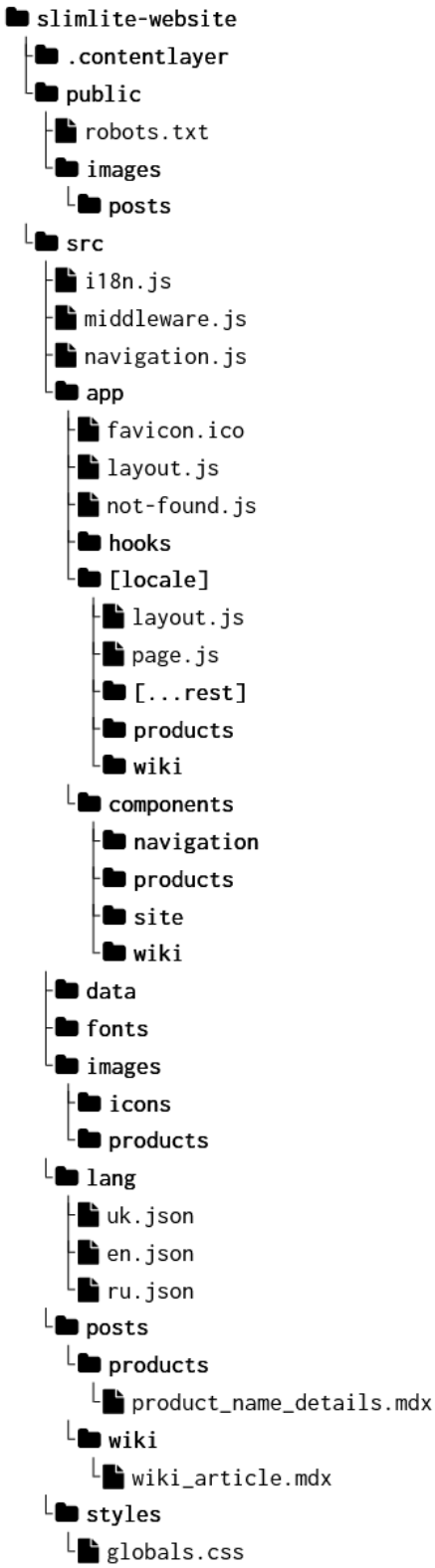


Рисунок 3.1 – Структура файлів та каталогів проєкту

Рисунок 3.1 ілюструє структуру каталогів та файлів вебсайту, який побудовано відповідно до сучасних принципів фронтенд-розробки, включаючи використання компонентного підходу, маршрутизації та багатомовної підтримки. Кореневий каталог проєкту називається «slimlite-website» і містить кілька підкаталогів і файлів, які відповідають за різні аспекти роботи програми. Каталог «.contentlayer», буде використовуватися для інтеграції системи генерації контенту Contentlayer, яка дозволяє працювати з MDX/Markdown файлами як джерелами контенту.

Публічний каталог містить ресурси, до яких можна отримати доступ безпосередньо з боку клієнта, наприклад, файл «robots.txt», який регулює індексацію сайту пошуковими системами, та зображення, які розміщуються в підкаталозі images/posts. Основна частина логіки програми зосереджена в каталозі src, який містить конфігураційні файли для інтернаціоналізації (i18n.js), файл проміжного програмного забезпечення (middleware.js) та навігації (navigation.js).

У каталозі src знаходиться каталог app, який містить файли, що відповідають за структуру сторінки, обробку помилок (not-found.js) та компонування інтерфейсу (layout.js). У цьому ж каталозі реалізовано локалізацію для користувача за допомогою каталогу «[locale]», який містить специфічні для мови компоненти, сторінки та підрозділи, включно з продуктами та вікі.

Окремий каталог компонентів містить багаторазові елементи інтерфейсу, згруповані за призначенням: навігація, продукти, сайт, вікі. Це демонструє дотримання принципу повторного використання коду. Додаткові ресурси, такі як дані, шрифти та зображення, організовані у відповідних каталогах, де, наприклад, зображення мають підкаталоги для іконок та продуктів.

Багатомовна підтримка забезпечується наявністю файлів перекладу JSON в каталозі «lang», особливо для української (uk.json) та англійської (en.json) мов. Розділ контенту представлений у вигляді Markdown-файлів з

розширенням «.mdx» в каталозі «posts», де окремо зберігаються статті, пов'язані з продуктами, та вікі-документація. Нарешті, директорія «styles» містить файл «globals.css», який відповідає за глобальні стилі в додатку.

Загалом, така структура демонструє системний підхід до організації веб-додатку з урахуванням масштабованості, багатомовної підтримки, розподілу обов'язків та практичного управління контентом. На рисунку 3.2 наведена логічна структура розроблюваного сайту.

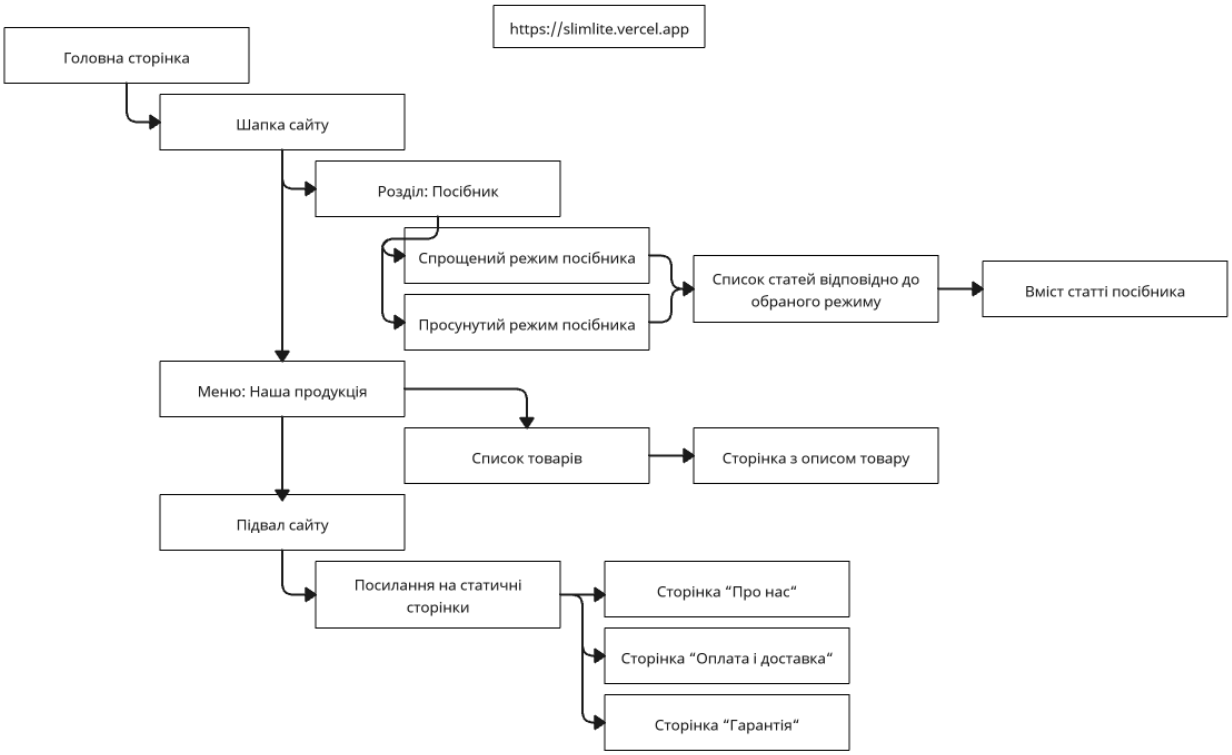


Рисунок 3.2 – Логічна структура сайту

Система навігації побудована за модульним принципом і включає наступні ключові компоненти:

1) Головна сторінка (https://simlite.vercel.app) - виконує функцію вхідного точки та надає загальний огляд ресурсу.
2) Розділ "Посібник" - реалізований у двох варіантах:

а) Спрощений режим (для початкових користувачів)

b) Просунутий режим (для досвідчених користувачів)

3) Секція "Наша продукція" містить:

a) Каталог товарів

b) Детальні картки продуктів з повним описом

4) Підвал сайту (footer) включає:

a) Навігаційні посилання

b) Статичні інформаційні сторінки:

i) "Про нас"

ii) "Оплата і доставка"

iii) "Гарантія"

1. Контентна частина - містить спеціалізовані матеріали посібника.

Структура демонструє продуманий підхід до групування інформації, де основні модулі (навігація, контент, сервісна інформація) логічно відокремлені, що сприяє ефективній орієнтації користувача. Поділ посібника на два режими свідчить про застосування принципу прогресивного розкриття інформації, що дозволяє адаптувати контент під різні рівні підготовки користувачів.

3.2. Проєктування інтерфейсу користувача

Структура розробленого вебсайту базується на сучасних стандартах веб-дизайну та принципах ергономіки. Основу інтерфейсу складають три ключові компоненти: заголовок (хедер), основний контентний блок та підвал (футер).

У розділі "Посібник" додатково реалізовано бічне меню (сайдбар), що формує чотирикомпонентну структуру інтерфейсу.

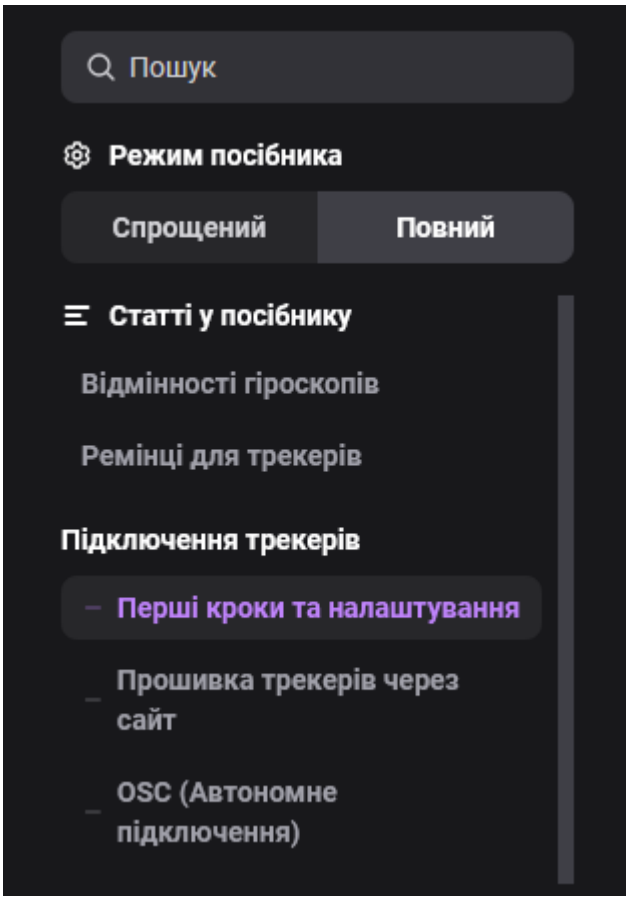


Рисунок 3.3 – Сайдбар з елементами керування у розділі «Посібник»

Хедер виконує кілька важливих функцій:

1. Ідентифікаційну - чітко вказує на приналежність до конкретного вебресурсу
2. Навігаційну - забезпечує швидкий доступ до основних розділів сайту
3. Функціональну - містить елементи керування (у даному випадку - перемикач мови)

Особливістю реалізації є розміщення перемикача мови у верхній частині інтерфейсу, що відрізняється від традиційного розташування у футері. Таке рішення обґрунтоване принципами візуальної ієрархії та забезпечує покращену видимість елемента, швидкий доступ до функціоналу та постійну доступність (завдяки fixed-позиціонуванню).

Під час розробки вебсайту було враховано загальноприйняті принципи, що застосовуються у сфері веброзробки, зокрема ті, що спрямовані на забезпечення структурної логіки, зручності користування, адаптивності інтерфейсу та інтуїтивної навігації. Стандартна архітектура сучасного вебресурсу передбачає наявність кількох ключових структурних компонентів: верхнього колонтитула (хедера), нижнього колонтитула (футера), основного контентного блоку, а також, за потреби, додаткових навігаційних елементів, зокрема бічної панелі (сайдбара). Кожен з вказаних елементів виконує певну функціональну роль та сприяє створенню цілісного, зручного для користувача інтерфейсу.

Хедер є однією з найважливіших зон інтерфейсу, оскільки саме з ним користувач взаємодіє на початковому етапі ознайомлення з ресурсом. Він виконує функції ідентифікації сайту, надає доступ до основного меню навігації та до додаткових функцій. У контексті реалізованого вебсайту хедер має фіксоване позиціонування (sticky), що забезпечує його постійну видимість при прокручуванні сторінки. Це значно покращує юзабіліті, оскільки навігаційні елементи завжди доступні незалежно від позиції користувача на сторінці. Однією з особливостей реалізації стало розміщення перемикача мови саме у верхньому колонтитулі. У більшості випадків елементи мовного перемикання розташовуються в нижній частині сторінки, однак з огляду на принципи доступності та логіку поведінки користувача, обране рішення є більш доцільним. Воно дозволяє миттєво знайти та змінити мову інтерфейсу, що особливо важливо для мультимовних ресурсів, орієнтованих на широку аудиторію.

Окрему увагу було приділено реалізації навігації у розділі «Посібник», який за своєю природою містить велику кількість статей, що мають бути легко доступними для користувача. У звичайних умовах на широких екранах для цього використовується стаціонарна бічна панель (сайдбар), яка дозволяє швидко переходити між категоріями та окремими статтями. Проте на пристроях з обмеженою шириною дисплея, таких як смартфони чи планшети, використання постійно видимої бічної панелі може призвести до зменшення корисного простору для основного контенту. З метою збереження повноцінного доступу до навігації та оптимізації інтерфейсу для різних типів пристроїв, було реалізовано мобільну версію сайдбара у вигляді висувної панелі, що активується за допомогою плаваючої кнопки.

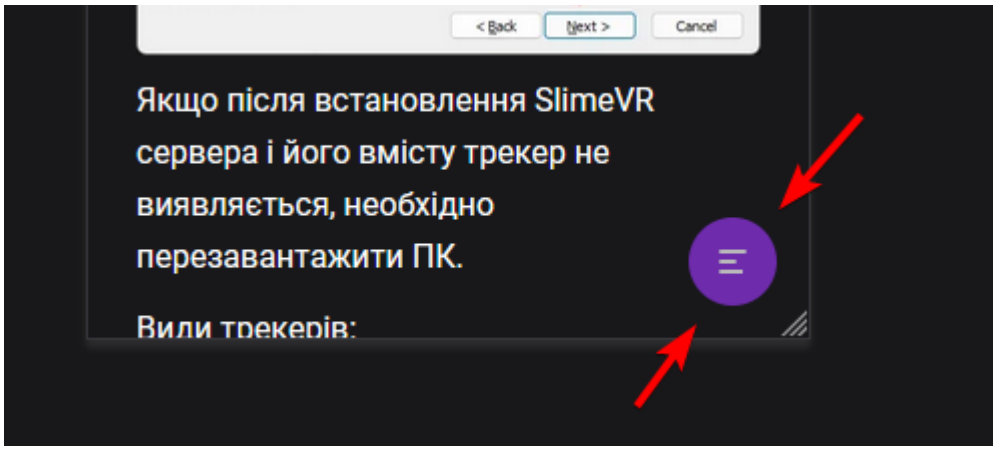


Рисунок 3.4 – Плаваюча кнопка визову висувної панелі розділу «Посібник»

Кнопка, продемонстрована на рисунку 3.4, постійно присутня на екрані, має достатню контрастність і не перекриває основний контент, завдяки чому забезпечується безперешкодний доступ до навігації навіть на мобільних пристроях. При натисканні на неї користувач отримує можливість викликати меню, яке зсувається з лівого боку екрану та містить посилання на інші статті або категорії. Такий підхід дозволяє зберегти повну функціональність розділу «Вікі» без шкоди для зручності користування на екранах будь-якого розміру.

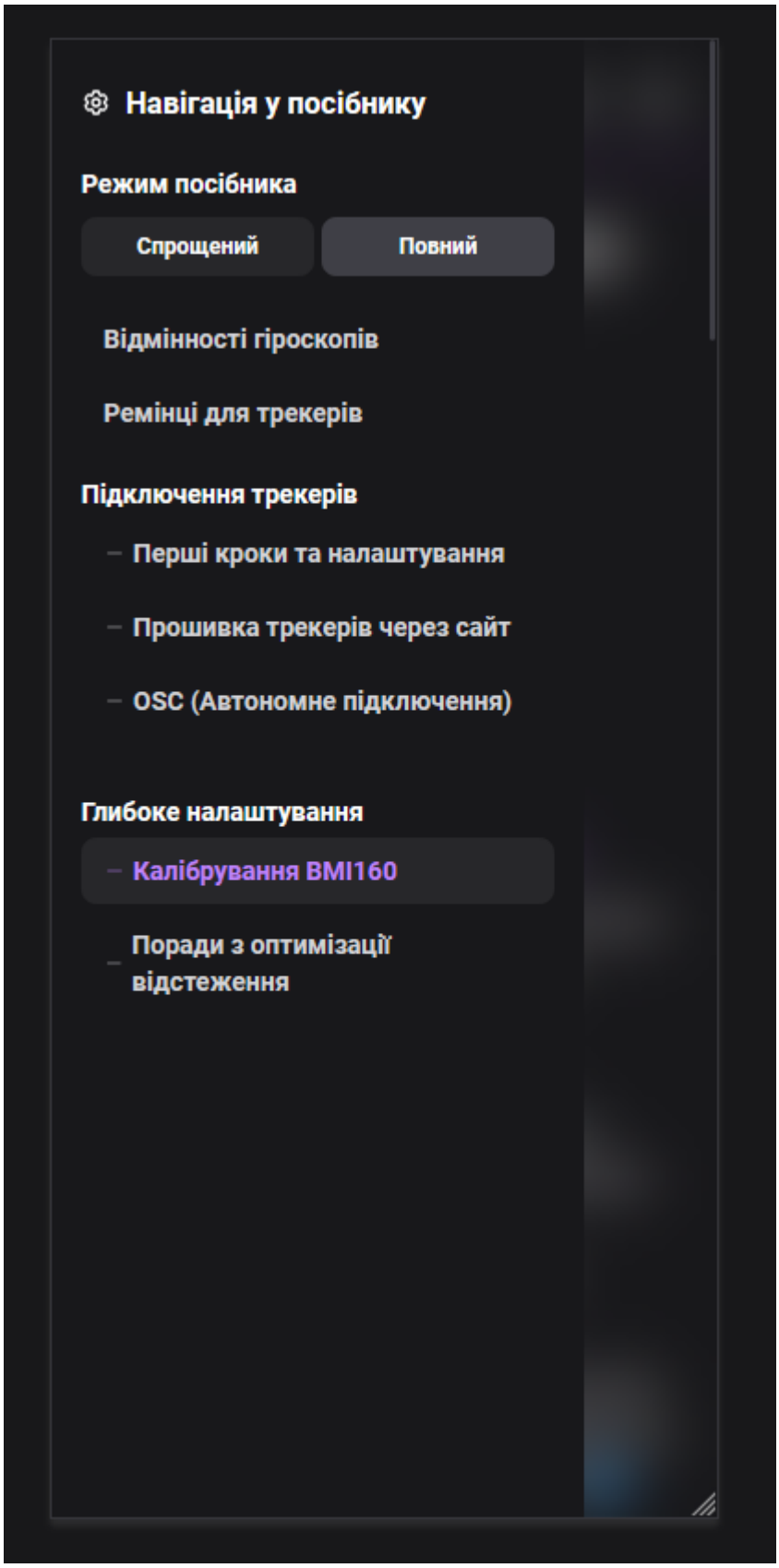


Рисунок 3.5 – Зовнішній вигляд висувної панелі з елементами керування в розділі «Посібник»

Загалом реалізовані рішення свідчать про дотримання сучасних стандартів у галузі вебдизайну та фронтенд-розробки, зокрема принципів адаптивності, доступності, гнучкості інтерфейсу та орієнтації на потреби кінцевого користувача. Усі елементи були спроєктовані з урахуванням кращих практик UX/UI-дизайну, що сприяє створенню ефективного, зручного та інтуїтивно зрозумілого вебінтерфейсу.

3.3. Розробка шаблонів основних сторінок

При розробці вебсайту часто виникає необхідність відображати інформацію в єдиному форматі, в той час як сам контент змінюється залежно від конкретного наповнення. Створення сторінок з однаковим зовнішнім виглядом вручну є трудомістким завданням, особливо у зв'язку з подальшими оновленнями. Шаблони використовуються для оптимізації цього процесу. Шаблони дозволяють створювати динамічні сторінки, тобто сторінки, які мають однакову структуру інтерфейсу, але різний вміст.

У зв'язку з реалізацією цього веб-сайту було створено три шаблони. Перший шаблон є базовим для всього сайту. Він містить основні HTML-елементи, що використовуються на кожній сторінці: метадані, основний контент сторінки, компонент панелі завантаження та компонент довідки, що відповідає за реалізацію багатомовної підтримки.

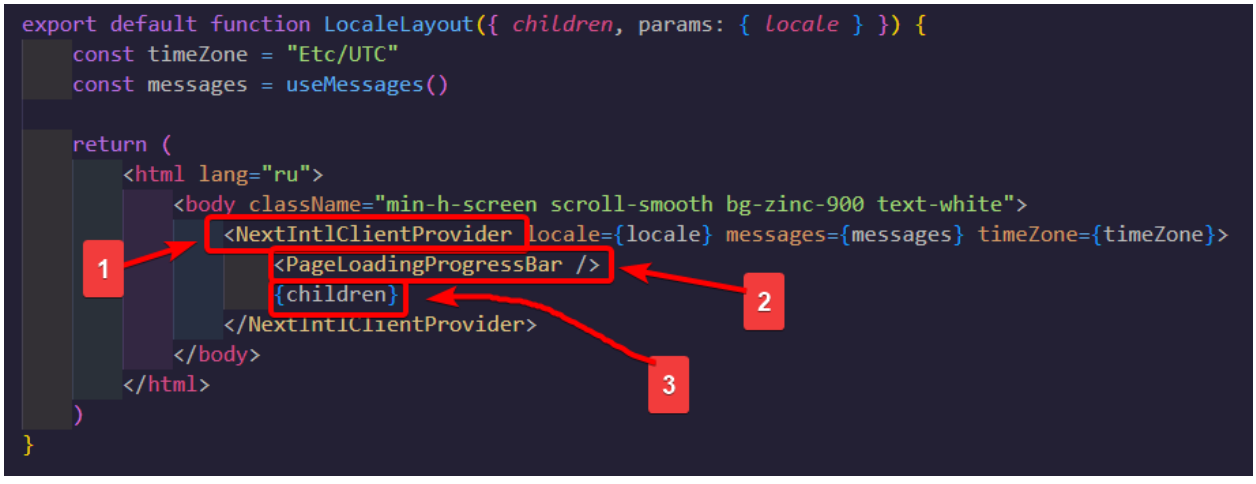


Рисунок 3.6 – Код головного (першого) шаблону вебсайту та його склад

На рисунку 3.6 зображено код та склад головного шаблону вебсайту. Число «1» на рисунку вказує на блок коду, відповідаючого за впровадження локалізації (мультимовності), число «2» - на компонент панелі завантаження,

число «3» вказує на об’єкт «children», замість якого бібліотека React автоматично включить контент підконтент відвіданої сторінки у шаблон.

Приклад метаданих, присутніх в файлі головного шаблону за межами layout функції:

```
{
    title: title,
    description: description,
    openGraph: {
      title: title,
      description: description,
      url: siteUrl,
      siteName: title,
      images: [
        {
          url: imageUrl,
          width: 512,
          height: 512,
        },
      ],
      locale: "uk_UA",
      type: "website",
    },
    icons: {
      icon: "/favicon.ico",
    },
    twitter: {
      card: "summary",
      title: title,
      description: description,
```

```
        images: [imageUrl],
    },
    metadataBase: new URL("http://localhost:3000")
}
```

Другий шаблон призначений для сторінок, пов'язаних з продукцією майстерні. Структура включає елементи для відображення фону, зображення та назви продукту. Основна частина сторінки відведена для динамічного контенту - опису конкретного товару.

Усі шаблони будуються за одним принципом - є певна функція, що містить код, у який вбудовується об’єкт «children» (як на рисунку 3.7).

Третій шаблон використовується для сторінок розділу «Посібник». Він складається з обов'язкових елементів, таких як верхній і нижній колонтитули, екран завантаження, який зникає після завершення завантаження контенту, і навігаційні компоненти, адаптовані до типу використовуваного пристрою - комп'ютера або мобільного телефону. Основну частину шаблону займає контент статей розділу «Посібник», а всі допоміжні структурні елементи розміщені в потрібних місцях. Вміст, що відображається на сторінці, автоматично інтегрується у відповідну структуру шаблону.

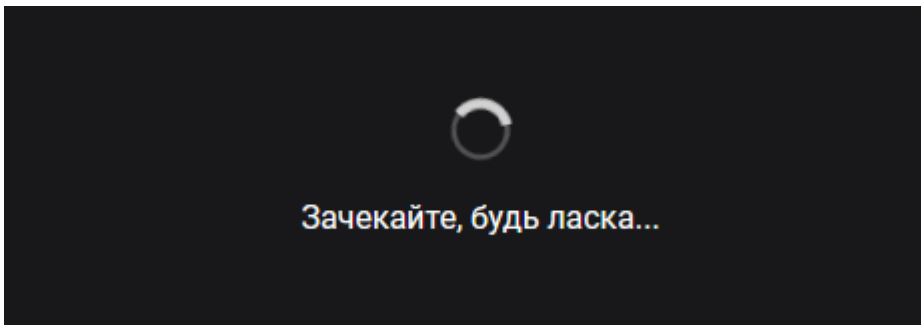


Рисунок 3.7 – елемент екрану завантаження у розділі «Посібник»

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1. Налаштування середовища розробки

Середовищем розробки для цього проєкту є редактор Microsoft Visual Studio Code. Microsoft Visual Studio Code - це універсальний, гнучкий та зручний редактор, який підтримує широкий спектр мов програмування, має інтеграцію з системами контролю версій та розширюється до великої кількості додатків. Налаштувавши його під власні потреби, можна розпочинати роботу над новим проєктом без жодних додаткових підготовчих дій - достатньо лише відкрити потрібну директорію в редакторі.Його популярність серед фронтенд- і бекенд-розробників пояснюється широкими можливостями розширення за допомогою плагінів, підтримкою багатьох мов програмування, швидкою роботою та інтеграцією з системами контролю версій.

Редактор був індивідуально налаштований для підвищення ефективності роботи. Серед встановлених розширень:

1. Prettier - автоматичний форматер коду, що забезпечує єдиний стиль написання;
2. MDX - підтримка підсвічування синтаксису для розширеного Markdown;
3. Lorem Ipsum - генератор тексту-заповнювача для тестування розмітки;
4. indent-rainbow - візуалізація рівнів відступів;
5. Violet Midnight - тема для інтерфейсу редактора.

Prettier налаштовано як інструмент форматування коду за замовчуванням для файлів HTML, JavaScript, Markdown і CSS, що зменшує кількість синтаксичних помилок і прискорює перевірку стилів коду. У файлі конфігурації Prettier були встановлені параметри, що відповідають вимогам до стилю кодування в цьому проекті.

покроковий алгоритм для розробки продукту:

1. Планування проекту, вибір інструментів та технологій
2. Встановлення текстового редактора та плагінів

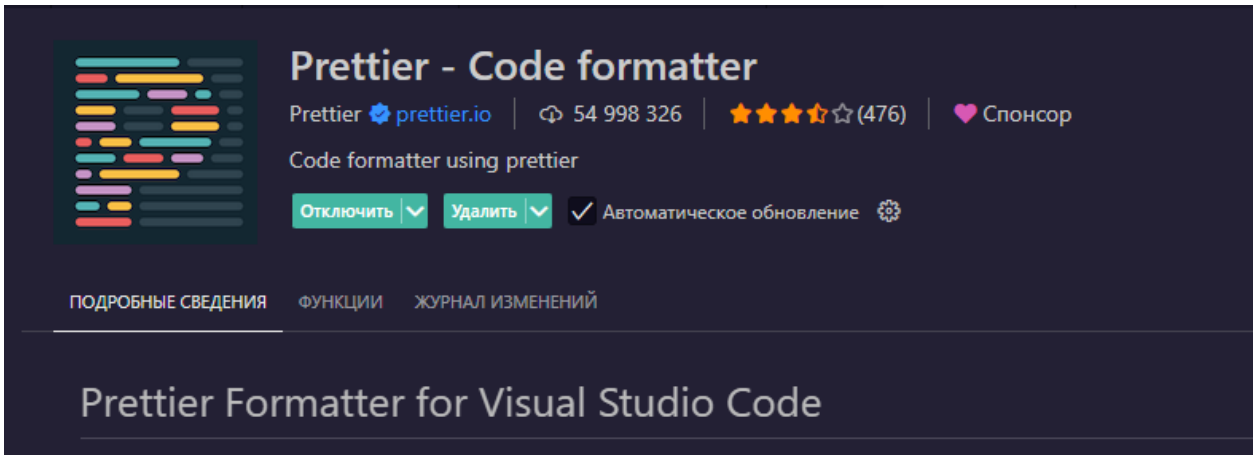


Рисунок 4.1 – Приклад встановленого плагіну «Prettier»

3. Встановлення рантайму (Node.js)

На рисунку 4.1 зображено приклад встановлюваного плагіну для відповідних стилів кодування.

На рисунку 4.2 зображено вікно встановлення рантайму (Node.js)

На рисунку 4.3 зображено створення нового проєкту із використанням скрипта.

На рисунку 4.4 зображено підключення бібліотек.

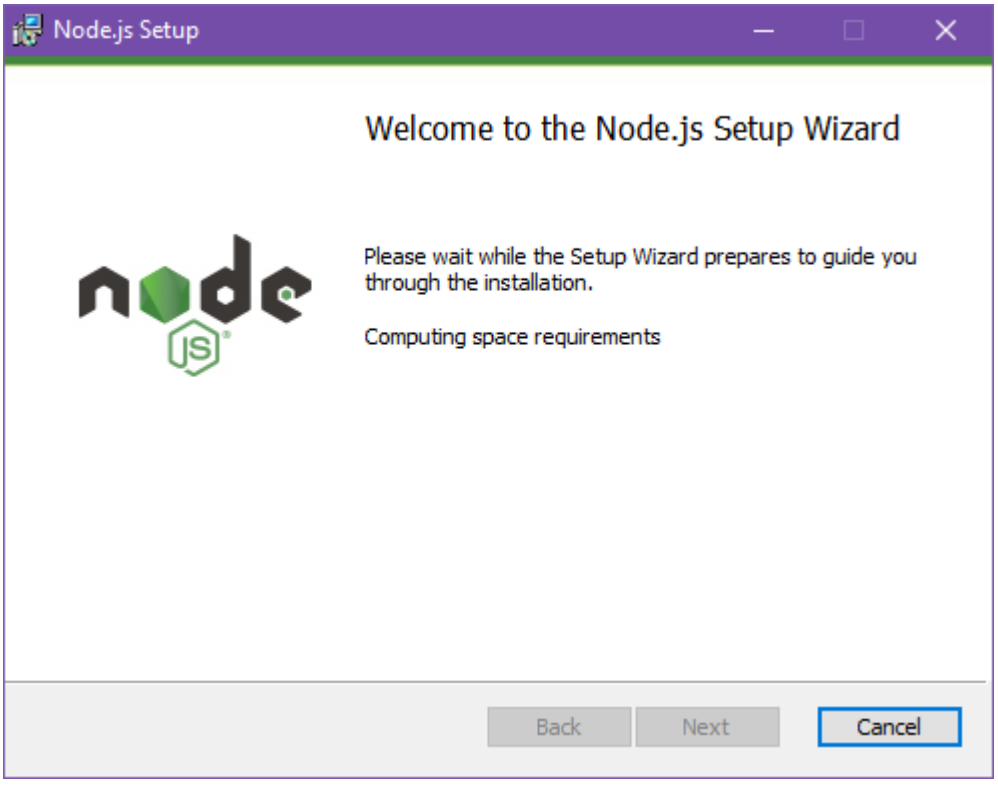


Рисунок 4.2 – Вікно встановлення NodeJS

4. Створення нового проєкту, використовуючи скрипт, наданий фреймворком

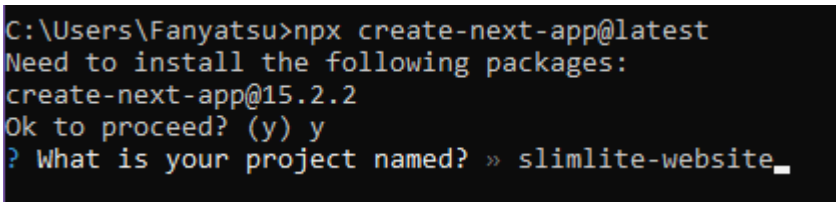


Рисунок 4.3 – Створення нового проєкту Next.js із використанням скрипта «create-next-app»

4.Підключення бібліотек до проєкту за допомогою установки пакетів із пакетного менеджера (npm) рантайму Node.js.

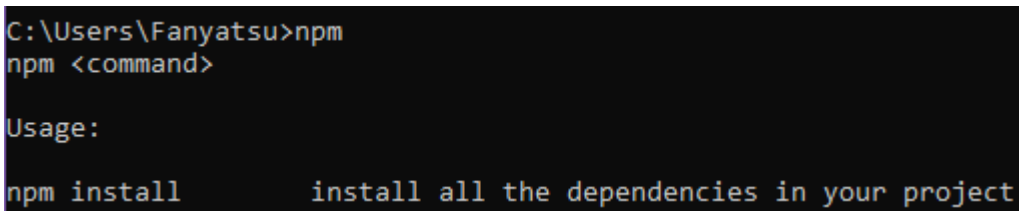


Рисунок 4.4 – Довідка до команди «npm install» пакетного менеджеру «npm»

5. Заповнення конфігураційних файлів:

```
{
  "compilerOptions": {
    "baseUrl": ".",
    "paths": {
      "@/*": ["./src/*"],
      "contentlayer/generated": ["./.contentlayer/generated"],
    },
  },
  "include": [
    ".contentlayer/generated"
  ],
}
```

6. Видалення заздалегідь згенерованих файлів з проекту, які є прикладом вебсайту на вибраному фреймворку.

7. Створення власного вебсайту за допомогою створення нових файлів у проєкті, написання коду в них:

```
import "@/styles/globals.css"
import PageLoadingProgressBar from "@/app/components/PageLoadingProgressBar"
import { getTranslations } from "next-intl/server"
import { NextIntlClientProvider, useMessages } from "next-intl"
import { locales, defaultLocale } from "@/navigation"

const title = "SlimLite"
const siteUrl = process.env.SITE_URL
const imageUrl = `${siteUrl}/images/og-image.png`
```

8. Наповнення вебсайту контентом:

```
---
id: 3
title: SlimLite трекер
description: Легкий, зручний та компактний трекер SlimeVR
image: slimlite-tracker.webp
coverImage: slimlite-tracker-cover.webp
---

SlimeVR — це інноваційний трекер для відстеження рухів
...
```

9. Тестування

Рисунок 4.5 – Тестування вебсайту у режимі розробки (dev environment)

На рисунку 4.5 зображено фрагмент процесу тестування вебсайту.

9. Генерація білда проєкту, готового до хостингу на продакшні

На рисунку 4.6 зображено процес генерації білда проєкту.

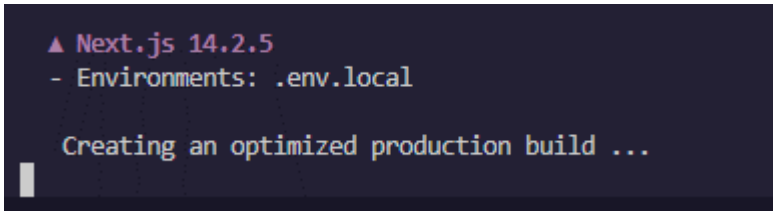


Рисунок 4.6 – Процес генерації білда проєкту через термінал

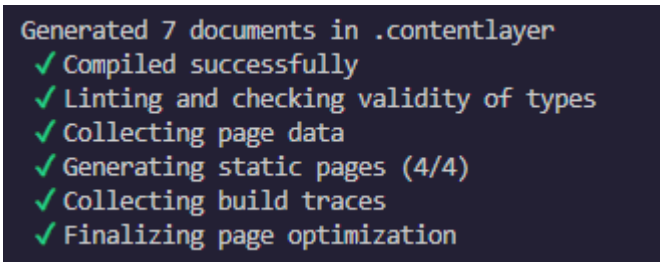


Рисунок 4.7 – Автоматична перевірка коду та генерація завершальних файлів

На рисунку 4.7 зображено процес генерації завершальних файлів.

11. Деплой білда на продакшн

Кожен із цих етапів має своє наукове обґрунтування та відіграє важливу роль у забезпеченні якості та ефективності програмного забезпечення. Нижче

більш детально розглядаються ці кроки, починаючи з етапу планування і закінчуючи деплоєм на продакшн-середовище.

4.2. Реалізація функціональних модулів

Реалізація модулів функціональності веб-додатків базується на принципах модульного дизайну, які передбачають розробку окремих компонентів з чітко визначеними інтерфейсами для взаємодії. Кожен модуль (наприклад, система аутентифікації, каталог продуктів або інтерактивне керівництво) розроблений як окремий блок, який може бути протестований, покращений та інтегрований окремо. Для забезпечення послідовної роботи модулів використовуються API для обміну даними між клієнтами і серверами.

Процес реалізації включає наступні етапи: проєктування модульної архітектури, розробка логіки, інтеграція з іншими компонентами системи і тестування. Наприклад, інтерактивний модуль керівництва може бути реалізований за допомогою React-компонентів з динамічним навантаженням контенту, а система управління товарами може бути реалізована через RESTful API шляхом взаємодії з базою даних. Гнучкі технології проектування (CSS Grid, Flexbox) і умовний дисплей використовуються для адаптації модулів до різних пристроїв. Така стратегія забезпечує масштабованість системи і дозволяє її подальше розширення.

Для ефективної взаємодії між модулями використовуються сучасні інструменти управління станом (наприклад, Redux або Context API у React), що дозволяють уникнути надмірної зв'язаності компонентів. Кожен модуль інкапсулює свою бізнес-логіку, а комунікація між ними відбувається через чітко визначені події або методи. Наприклад, модуль кошика може отримувати дані про товари через глобальний стан додатку, а модуль фільтрації — генерувати події для оновлення списку продуктів.

Важливим аспектом є забезпечення безпеки та стабільності роботи модулів. Для цього застосовуються механізми обробки помилок, валідації вхідних даних та лімітування запитів до API. Додатково, для підвищення продуктивності може використовуватися ліниве завантаження (lazy loading) модулів, що особливо актуально для великих додатків. Підхід до реалізації функціональних модулів, заснований на принципах SOLID та DRY, забезпечує гнучкість, легкість підтримки та можливість подальшого масштабування системи.

На сайті використовується файл «middleware.js». Middleware представляє собою критично важливий компонент архітектури веб-додатку, що виконує функцію проміжного обробника між клієнтськими запитами та серверною логікою. Його впровадження дозволяє створити ефективний механізм обробки запитів, який забезпечує безпеку, стабільність та оптимальну продуктивність системи.

Основним призначенням Middleware є централізована обробка вхідних HTTP-запитів перед їх передачею до бізнес-логіки додатку. Це включає комплекс заходів з валідації та нормалізації даних, перевірки автентифікації користувачів, контролю прав доступу, логування системних подій та обробки потенційних помилок. Важливою особливістю Middleware є його здатність формувати ланцюжок обробників, де кожен елемент відповідає за конкретний аспект обробки запиту.

У контексті сучасних JavaScript-фреймворків, зокрема Next.js, Middleware набуває особливого значення. Він ефективно інтегрується з механізмами серверного рендерингу (SSR), статичної генерації сторінок (SSG) та інкрементальної регенерації (ISR). Практична реалізація Middleware дозволяє виносити спільну логіку обробки запитів в окремий модуль, що суттєво спрощує архітектуру додатку та покращує можливості його масштабування.

Технічні переваги використання middleware включають:

1. Підвищення рівня безпеки через централізовану обробку потенційно небезпечних запитів
2. Оптимізацію продуктивності за рахунок кешування та стиснення даних
3. Поліпшення підтримки коду через чітке розділення обов'язків
4. Можливість гнучкої конфігурації обробки запитів для різних маршрутів
5. Спрощення процесу A/B тестування та персоналізації контенту

```
import createMiddleware from "next-intl/middleware"
import { locales, defaultLocale } from "@/navigation"

export default createMiddleware({
    locales: locales,
    defaultLocale: defaultLocale,
})

export const config = {
    matcher: [
        "/",
        "/wiki",
        "/wiki/:path*",
        "/products/:path*",
        "/(en|ru|uk)/:path*",
    ],
}
```

Рисунок 4.8 – Код файлу «middleware.js»

На рисунку 4.8 проілюстровано використання Middleware.js.

Реалізація хука «useClickOutside», який було розроблено для визначення кліків поза елементами інтерфейсу, такими як спливаючі вікна

```
import { useEffect } from "react"

export const useClickOutside = (containerRef, buttonRef,
callback) => {
    const handleClick = (event) => {
        if (buttonRef.current &&
buttonRef.current.contains(event.target)) {
            return
        }

        if (containerRef.current && !
containerRef.current.contains(event.target)) {
            callback()
        }
    }

    useEffect(() => {
        document.addEventListener("mousedown", handleClick)

        return () => {
            document.removeEventListener("mousedown",
handleClick)
        }
    }, [containerRef, buttonRef, callback])
}
```

Рисунок 4.9 – Код React хука «useClickOutside»

На рисунку 4.9 зображено код React хука «useClickOutside».

Мовний перемикач. Він складається з розумного елемента посилання, який виглядає як кнопка в списку мов і клік, на якому змінюється мова, і самого перемикача, який представлений як кнопка, яка розширює список з доступними мовами на вибір. На рисунку 4.10 проілюстровано створення мовного перемикача.

```
const LanguageSwitcher = () => {
    const pathName = usePathname()
    const locale = useLocale()

    const [isDropdownOpen, setIsDropdownOpen] = useState(false)

    const dropdownRef = useRef(null)
    const buttonRef = useRef(null)

    const toggleDropdown = () => {
        setIsDropdownOpen(!isDropdownOpen)
    }

    const flagIconsByLocales = {
        en: <EnglishFlag className="mr-0.5 h-4 w-4" />,
        uk: <UkrainianFlag className="mr-0.5 h-4 w-4" />,
    }

    useClickOutside(dropdownRef, buttonRef, () => {
        if (isDropdownOpen) {
            setTimeout(() => setIsDropdownOpen(false), 50)
        }
    })
```

Рисунок 4.10 – Код мовниого перемикача (LanguageSwitcher.js).

Алгоритми пошуку тексту за статтями розділу «Посібник» та візуальне виділення знайдених результатів. Використовують переваги регулярних виразів та циклів для прокрутки та відокремлення тексту від усього тексту статті.

На рисунку 4.11 зображено використання алгоритму пошуку у статтях в розділі «Посібник».

```
const getSnippet = (rawText, searchQuery) => {
        const normalizedRawText = rawText.toLowerCase()
        const normalizedSearchQuery = searchQuery.toLowerCase()

        const searchIndex = normalizedRawText.indexOf(normalizedSearchQuery)

        if (searchIndex == -1) {
            return ""
        }

        const words = rawText.split(/\s+/)

        let wordIndex = 0
        let charCount = 0

        for (let i = 0; i < words.length; i++) {
            charCount += words[i].length + 1
            if (charCount > searchIndex) {
                wordIndex = i
                break
            }
        }

        const start = Math.max(0, wordIndex - 5)
        const end = Math.min(words.length, wordIndex + 11)

        const snippet = words.slice(start, end).join(" ")

        return snippet
    }

    const getHighlightedText = (text, highlight) => {
        const parts = text.split(new RegExp(`(${highlight})`, "gi"))
        return (
            <>
```

Рисунок 4.11 – Фрагмент коду алгоритму пошуку у статтях.

4.3. Адаптивна верстка і мобільна оптимізація

У сучасних методологіях розробки адаптивний дизайн став обов'язковим стандартом, що обумовлено стрімким зростанням мобільного трафіку, який за останніми даними перевищує 60% від загального обсягу. Ця тенденція вимагає ретельного підходу до проектування інтерфейсів, які повинні коректно відображатись на пристроях з будь-якими розмірами екрану.

У рамках даного проєкту був обраний сучасний підхід до адаптації, що передбачає використання єдиного кодової бази з гнучкою верстkою замість розробки окремого мобільного сайту.

Технічна реалізація адаптивності ґрунтується на комбінації кількох ключових технологій. Основним інструментом став фреймворк Tailwind CSS, який значно спрощує процес створення гнучких інтерфейсів завдяки

вбудованій системі брейкпоінтів та утилітарним класам. Це дозволяє реалізовувати адаптацію без необхідності писати велику кількість власних медіа-запитів. Наприклад, для елементів, які повинні змінювати свою поведінку на різних пристроях, використовуються спеціальні префікси (sm, md, lg, xl), що автоматично адаптують стилі відповідно до розміру екрану.

Для більш складних компонентів, таких як навігаційне меню, був застосований підхід умовного рендерингу. Це передбачає використання React-хуків для визначення розміру екрану та відповідного відображення різних версій компонента. Технічно це реалізовано через підписку на подію зміни розміру вікна та стан, який визначає поточний тип пристрою. Такий підхід дозволяє не лише змінювати стилі, але й повністю перебудовувати структуру компонентів для різних пристроїв.

Важливим аспектом реалізації стало використання сучасних CSS-методологій, зокрема гнучких сіток (Flexbox і Grid), відносних одиниць виміру (rem, vw, %) та принципу прогресивного поліпшення. Це забезпечує плавну адаптацію інтерфейсу без різких змін при зміні розміру вікна. Для елементів, які потребують менш радикальних змін, застосовуються простіші підходи: приховування другорядних елементів на малих екранах, зміна напрямку розташування з горизонтального на вертикальне або пропорційне зменшення розмірів.

Особливу увагу приділено продуктивності рендерингу на мобільних пристроях. Усі адаптивні рішення реалізовані з урахуванням мінімізації перерозрахунків стилів та макету, що досягається за рахунок оптимального використання CSS-властивостей та обмеження кількості умовного рендерингу. Це забезпечує плавну роботу інтерфейсу навіть на слабких мобільних пристроях, що є критично важливим для забезпечення позитивного користувацького досвіду.

4.4. Розгортання сайту на хостингу

Для забезпечення доступності веб-сайту в інтернеті необхідне відповідне хостингове рішення. Сучасні веб-проєкти зазвичай використовують один із таких варіантів: виділені сервери, віртуальні (хмарні) сервери або серверний хостинг. У рамках даного проекту було прийнято рішення про використання безсерверної архітектури (serverless), що є оптимальним варіантом для невеликих веб-сайтів з мінімальними вимогами до ресурсів. Важливою перевагою такого підходу є автоматичне масштабування потужностей при збільшенні навантаження, що забезпечує стабільну роботу сайту.

В якості хостинг-провайдера було обрано платформу Vercel, яка не лише надає сучасні хостингові послуги, але й є розробником фреймворку Next.js, що використовувався для створення даного веб-сайту.

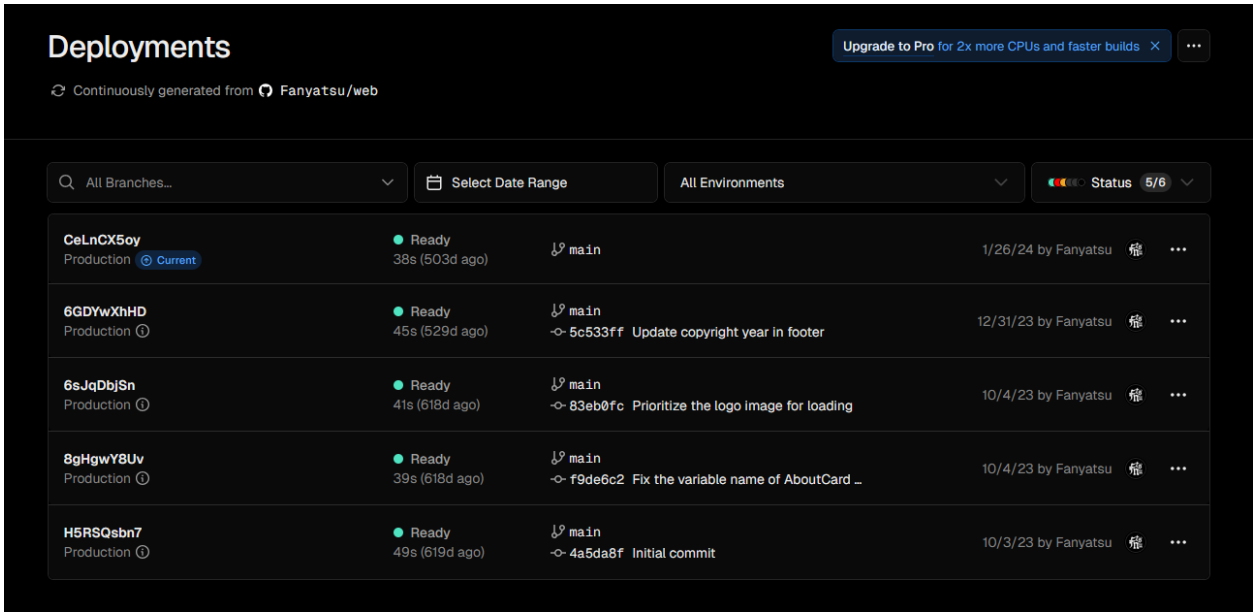
Рисунок 4.12 – Логотип платформи «Vercel»

Процес розгортання включав наступні етапи:

1. Реєстрація облікового запису клієнтом на платформі Vercel
2. Використання GitHub як системи контролю версій для зберігання вихідного коду
3. Вибір та налаштування доменного імена для публічного доступу до сайту

Після налаштування цих параметрів, платформа Vercel автоматично виконує процес збирання проекту: отримує вихідний код з GitHub-

репозиторію, компілює його та розгортає готовий до роботи екземпляр веб-сайту. Весь процес займає мінімальний час (кілька хвилин), що дозволяє швидко забезпечити доступність сайту для кінцевих користувачів.



Риснок 4.13 – Демонстрація історії розгорнутих білдів через платформу «Vercel»

Застосування безсерверної архітектури у рамках розгортання вебсайту на платформі Vercel не лише спростило процес публікації ресурсу, але й забезпечило відповідність сучасним тенденціям у сфері вебінфраструктури. Однією з ключових переваг обраного підходу є висока автоматизація обслуговування — платформа самостійно виконує масштабування, балансування навантаження та моніторинг стану додатку без необхідності втручання з боку розробника або адміністратора. Це дозволяє зосередити ресурси на вдосконаленні функціональності й інтерфейсу сайту, а не на обслуговуванні серверної інфраструктури.

Крім того, Vercel підтримує безперервну інтеграцію (CI) та безперервне розгортання (CD), що значно оптимізує життєвий цикл програмного забезпечення. Будь-які зміни у вихідному коді, збережені у репозиторії GitHub, автоматично ініціюють процес збирання нового релізу та оновлення

сайту. Така схема істотно знижує ризик помилок під час ручного розгортання та забезпечує актуальність контенту для кінцевих користувачів у режимі реального часу.

Ще одним важливим аспектом є висока доступність (high availability) платформи, що досягається завдяки використанню глобальної мережі доставки контенту (CDN), яка дозволяє значно скоротити час відгуку вебресурсу незалежно від географічного розташування користувача. Це особливо важливо для малого бізнесу, який прагне забезпечити належну якість обслуговування клієнтів за мінімальних витрат.

Також варто зазначити відповідність архітектури принципам екологічної стійкості: serverless-рішення працюють на спільній інфраструктурі з оптимізованим використанням ресурсів, що дозволяє зменшити загальне енергоспоживання в порівнянні з традиційними фізичними або віртуальними серверами.

4.5. Потенційні загрози та уразливості

Оскільки в проєкті використовується значна кількість відомих JavaScript-бібліотек, популярність яких виходить за межі окремих спільнот, уразливості в таких бібліотеках трапляються досить часто. У зв’язку з цим надзвичайно важливо підтримувати актуальність усіх компонентів вебсайту шляхом регулярного оновлення npm-пакетів до останніх стабільних версій.

Хостинг-платформа Vercel не надає фіксованих обчислювальних ресурсів, натомість динамічно виділяє мінімально необхідні ресурси для запуску вебсайту. Такий підхід може створювати фінансові ризики в разі атаки на вебресурс з метою його перевантаження або відключення, оскільки система автоматично масштабуватиме виділені ресурси. У нашому випадку дана проблема нівелюється можливістю встановлення обмежень на використання ресурсів.

Існує ймовірність виникнення помилок, що можуть спричинити збої як на стороні клієнта, так і на сервері. З огляду на те, що в межах проєкту не було залучено професійних тестувальників, існує ризик недостатнього тестового покриття, що може призвести до ситуацій, у яких вебсайт стає недоступним — як для окремого користувача, так і глобально. У випадку критичних збоїв Vercel здійснює автоматичний перезапуск процесів через власний менеджер.

Для мінімізації ризиків, пов’язаних із помилками у програмному коді, було досягнуто домовленості щодо безоплатної підтримки функціоналу сайту на початковому етапі його експлуатації.

Застосування численних зовнішніх JavaScript-бібліотек у рамках вебпроекту, з одного боку, забезпечує високу швидкість розробки, доступ до перевірених рішень і загальновизнаних практик, однак, з іншого — створює додаткові виклики у сфері інформаційної безпеки та підтримки стабільності роботи системи. Вразливості в широко використовуваних бібліотеках з

відкритим кодом виявляються регулярно, і навіть незначні оновлення можуть впливати на сумісність або функціональність вебсайту. Тому критично важливим аспектом стало забезпечення постійного моніторингу та оновлення залежностей через систему керування пакетами npm, що дозволяє автоматизовано перевіряти й оновлювати бібліотеки до останніх стабільних версій, а також здійснювати аудит безпеки на основі існуючих баз уразливостей.

Особливості обраної хостинг-платформи Vercel, яка використовує динамічне виділення ресурсів за принципом безсерверної архітектури, забезпечують високу гнучкість, але одночасно створюють потенційні фінансові ризики. У разі DDoS-атаки або іншого типу навмисного перевантаження сайту, система автоматичного масштабування інфраструктури може призвести до неочікуваних витрат. З метою зниження цього ризику було впроваджено обмеження на максимальне використання ресурсів, що дозволяє уникнути непрогнозованого зростання вартості обслуговування ресурсу.

Окремої уваги потребує проблема забезпечення стабільності функціонування вебсайту. У зв’язку з обмеженим складом проєктної команди та відсутністю професійного етапу тестування, покриття функціональності автоматичними тестами є недостатнім. Це створює ймовірність виникнення неочевидних логічних помилок, які можуть спричинити як часткові, так і повні збої у роботі вебресурсу — як на рівні клієнтської частини, так і на рівні обробки запитів сервером. Водночас хостинг-платформа Vercel частково компенсує ці ризики завдяки реалізованій системі автоматичного відновлення — у разі виявлення збоїв виконання, процеси розгортання та обробки запитів автоматично перезапускаються із заздалегідь збережених стабільних версій.

Для додаткової мінімізації негативного впливу можливих збоїв та для підтримки високого рівня надійності вебсайту на етапі впровадження було погоджено рішення про безоплатну технічну підтримку з боку розробника. Це передбачає моніторинг функціональності, оперативне усунення критичних

помилок і поступове впровадження додаткових заходів контролю якості коду, що сприятиме поступовому зниженню ризиків і стабілізації роботи ресурсу в довгостроковій перспективі.

Таким чином, у процесі реалізації проєкту було враховано основні ризики, пов’язані із сучасною практикою використання динамічного хостингу, зовнішніх бібліотек та обмеженої підтримки на ранніх етапах експлуатації. Застосовані превентивні заходи та інструменти дозволяють вважати обрані підходи такими, що відповідають практичним вимогам безпечної, економічно обґрунтованої та стійкої архітектури вебресурсу.

4.6. Використання Cloudflare для захисту від ботів та шифрування трафіку

Cloudflare - всесвітньо відома, і, ймовірно, найкраща і найдоступніша система проксі-сервера і фільтрації для мережевого веб-трафіку, яка має безліч механізмів захисту сайтів від атак ботботів і деяких видів несанкціонованого доступу. Також керує записами DNS, сертифікатами SSL, правилами перенаправлення, кешуванням вмісту, оптимізацією сторінки та багато іншого.

Рисунок 4.13 – Логотип «Cloudflare»

При створенні цього вебсайту було використано всі перераховані вище технології. Наприклад, управління записами DNS дуже зручне і інтуїтивно зрозуміле, набагато багатше за функціональністю, ніж використання ресурсу Vercel. Для даного сайту важливо мати повністю функціонуючий SSL, тому обрано механізми Cloudflare для отримання довірених SSL сертифікатів, які автоматично переоформлюються до закінчення терміну їх дії. В даному випадку також необхідні перенаправлення для перетворення всього трафіку з незахищеного протоколу HTTP в HTTPS. Кешування Cloudflare охоплює велику частину сайту, оскільки воно майже повністю скомпільоване з файлів, що Cloudflare дуже добре кешує.

Оптимізація застосовується до макетів, коду та зображень, що дозволяє значно зменшити розмір контенту, необхідного для завантаження браузером, ще більше мінімізуючи використання трафіку як клієнта, так і нас. Для захисту від ботів використано каппі-стратегію, наприклад, якщо трафік клієнта підозрілий, у виняткових випадках доступ до сайту може бути повністю закритий, якщо виявлено, що користувач - бот.

Один із ключових елементів забезпечення безпеки, стабільності та швидкодії сучасного вебресурсу — використання сервісів мережевого посередництва та захисту, зокрема платформи Cloudflare, яка є визнаним лідером у сфері проксі-серверних технологій, веб-фільтрації та оптимізації мережевого трафіку. Завдяки своїм розвиненим можливостям, Cloudflare виконує не лише функції розподілу навантаження та прискорення доставки контенту, але й надає ефективні механізми захисту від бот-трафіку, спроб несанкціонованого доступу, DDoS-атак, а також забезпечує централізоване керування доменними записами (DNS), сертифікатами SSL, правилами редиректу, кешуванням і стисканням контенту.

У рамках реалізації даного вебпроєкту було інтегровано весь спектр базових функцій Cloudflare. Зокрема, управління DNS-записами здійснюється безпосередньо через інтерфейс Cloudflare, який відзначається високим рівнем інтуїтивності та гнучкості, що вигідно відрізняє його від аналогічних інструментів, доступних на платформі Vercel. Особливу увагу приділено SSL-захисту: з метою забезпечення конфіденційності та цілісності переданих даних було налаштовано автоматичне отримання та продовження довірених сертифікатів SSL, які Cloudflare видає безкоштовно й оновлює без втручання з боку розробника. Для унеможливлення доступу через незахищене з’єднання протоколу HTTP були створені правила примусового перенаправлення на HTTPS, що є критично важливим для забезпечення безпеки взаємодії з користувачем.

Важливою складовою є також механізми кешування, які дозволяють значно знизити навантаження на серверну частину проєкту. Завдяки тому, що

структура сайту в основному сформована зі статичних файлів, Cloudflare ефективно зберігає та віддає кешовані версії сторінок, що пришвидшує завантаження ресурсу з будь-якої точки світу. Це також позитивно впливає на показники продуктивності, що є важливим чинником для SEO та загального користувацького досвіду.

Комплексна оптимізація контенту на рівні Cloudflare охоплює автоматичне стиснення HTML, CSS, JavaScript, а також адаптивну оптимізацію зображень. Завдяки цьому вдалося зменшити обсяг передаваних даних, що, у свою чергу, позитивно позначається на швидкості відображення сайту, особливо для користувачів з повільним інтернет-з’єднанням або мобільних пристроїв.

З метою забезпечення захисту від автоматизованих атак та небажаного бот-трафіку було впроваджено стратегії фільтрації запитів. Зокрема, Cloudflare надає інструменти поведінкової аналітики для виявлення підозрілих дій. У разі фіксації аномальної активності або порушень політик доступу, реалізується система CAPTCHA-захисту або повне блокування доступу до ресурсу, що дозволяє ефективно нейтралізувати потенційні загрози.

Отже, Cloudflare до архітектури вебсайту істотно підвищила його стійкість до навантажень, швидкодію, безпеку передачі даних та загальну надійність у роботі. Використання можливостей цієї платформи є обґрунтованим рішенням для проєктів малого та середнього масштабу, які прагнуть досягти професійного рівня якості без значного фінансового навантаження.

ВИСНОВКИ

У межах реалізації поставленого завдання було успішно виконано розробку інформаційного вебсайту для майстерні, що відповідає сучасним вимогам до структури, функціональності, дизайну та адаптивності. Основна мета проєкту - створення платформи для представлення асортименту виробів із можливістю ознайомлення з детальною інформацією про кожен товар - була досягнута шляхом розробки інтерфейсних рішень, що забезпечують зручний доступ до відповідного контенту.

Було реалізовано розділи, що відповідають практичним потребам цільової аудиторії, зокрема інформаційні сторінки щодо доставки, оплати, гарантійного обслуговування, а також сторінку з описом діяльності майстерні. Важливим досягненням стало впровадження навчального розділу «Посібник», що функціонує у двох режимах (для початківців і досвідчених користувачів), що відповідає принципам персоналізованого користувацького досвіду та підвищує загальну корисність ресурсу.

Сайт також відповідає вимогам мультимовності, підтримуючи українську та англійську мови, що розширює потенційну аудиторію та дозволяє інтегрувати ресурс у ширший інформаційний простір. Особлива увага була приділена адаптивності - інтерфейс коректно масштабується та змінює компонування залежно від типу пристрою, що є критично важливим в умовах зростаючої мобільності користувачів.

Крім безпосередньої технічної реалізації, у процесі роботи було враховано загальний контекст цифровізації малого бізнесу в Україні. Зокрема, проєкт демонструє можливість створення якісного онлайн-ресурсу з обмеженими ресурсами, враховуючи реальні виклики, з якими стикаються представники малого підприємництва — фінансові обмеження, недостатній рівень цифрової компетентності, складнощі з підтримкою та просуванням

сайту. Водночас розроблений ресурс відповідає очікуванням сучасних користувачів щодо дизайну, зручності навігації, швидкодії та функціональності, що потенційно сприяє підвищенню довіри до бренду та розширенню клієнтської бази.

Оформлення роботи виконано відопвідно до методичних рекомендацій [5]. Результати виконання роботи апробовані на XLVIII Міжнародній науковій студентській конференції[6].

Таким чином, виконаний проєкт підтвердив практичну доцільність впровадження цифрових інструментів у діяльність малих підприємств та може розглядатися як зразок раціонального підходу до розробки інформаційних вебресурсів у рамках обмежених ресурсів та високих вимог з боку цільової аудиторії.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. **Statcounter GlobalStats** (2023). *Desktop vs Mobile vs Tablet Market Share Worldwide* [Електронний ресурс]. – Режим доступу: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet. – Назва з екрана.
2. **Ericsson Mobility Report** (2023). *Ericsson Mobility Report November 2023* [Електронний ресурс]. – Режим доступу: https://www.ericsson.com/en/reports-and-papers/mobility-report. – Назва з екрана.
3. **Pew Research Center** (2023). *Mobile Fact Sheet* [Електронний ресурс]. – Режим доступу: https://www.pewresearch.org/internet/fact-sheet/mobile/. – Назва з екрана.
4. **Google Search Central** (2023). *Mobile-first indexing* [Електронний ресурс]. – Режим доступу: https://developers.google.com/search/mobile-sites. – Назва з екрана.
5. Ольховська О.В. Методичні рекомендації до виконання кваліфікаційної роботи для студентів за освітньою програмою «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» - 58 ст.
6. Збірник тез доповідей Міжнародної науково-технічної конференції студентів, аспірантів та молодих вчених "Комунальна та промислова теплоенергетика" (КНІТ), випуск 2, 2023 р. / [редкол.: О. В. Іванов, П. П. Петров та ін.]. – Полтава : Полтавський університет економіки і торгівлі, 2023. – 150 с. – Режим доступу: https://dspace.puet.edu.ua/bitstream/123456789/13014/3/%D0%97%D0%91%D0%86%D0%A0%D0%9D%D0%98%D0%9A%20%D1%82%D0%B5%D0%B7%20%D0%9A%D0%9D%D0%86%D0%A2%20%D0%B2%D0%B8%D0%BF%D1%83%D1%81%D0%BA%202%202023.pdf
7. Петренко, І. Оптимізація та безпека веб-сайтів: посібник з WordPress. Київ: Освіта, 2022.

8. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures: Dissertation. University of California, Irvine. [Електронний ресурс]. – Режим доступу: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
9. Wang, L., Ranjan, R., Chen, J., Benatallah, B. (2011). Cloud computing: a perspective study. *Future Generation Computer Systems*, **29**(2), 137–146.
10. Doupe, A., Cova, M., Vigna, G. (2011). Why Johnny can't pentest: An analysis of black-box web vulnerability scanners. *Proceedings of the 7th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (DIMVA), 111–131.
11. Yu, M., Xie, Y., Ke, Q., Sun, L. (2019). Cloud-based DDoS attacks and defenses: A review. *Journal of Network and Computer Applications*, **123**, 15–33.
12. ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. International Organization for Standardization, 2011. – 31 p.
13. Google. (2023). Web Vitals. [Електронний ресурс]. – Режим доступу: https://web.dev/vitals/