МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБ-САЙТУ З ВИКОРИСТАННЯМ ІНТЕРАКТИВНОЇ 3D ГРАФІКИ»**

Виконав: студент 4 курсу, групи 6.1220
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
І. В. Дубровний
(ініціали та прізвище)

Керівник доцент кафедри комп'ютерних наук, к.т.н. Добровольський Г.А.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри програмної інженерії, доцент, к.ф.-м.н. Лісняк А. О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., доцент

_____ Шило Г.М.
(підпис)

" ____ " _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Дубровному Іллі Володимировичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка веб-сайту з використанням інтерактивної 3Д графіки
керівник роботи Добровольський Геннадій Анатолійович, к.т.н.
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2024 року № _____

2. Строк подання студентом роботи 01.06.2024

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Реалізація веб-сайту, на якому буде використовуватись інтерактивна 3D-графіка.

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 25.12.2023

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	05.02.2024	
2.	Збір вихідних даних.	10.02.2024	
3.	Обробка методичних та теоретичних джерел.	12.02.2024	
4.	Розробка першого та другого розділу.	13.03.2024	
5.	Розробка третього розділу.	19.04.2024	
6.	Оформлення та нормоконтроль кваліфікаційної роботи бакалавра.	20.05.2024	
7.	Захист кваліфікаційної роботи.	22.06.2024	

Студент _____ І. В. Дубровний
(підпис) (ініціали та прізвище)

Керівник роботи _____ Г.А. Добровольський
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка веб-сайту з використанням інтерактивної 3Д графіки»: 80 с., 25 рис., 30 джерел, 6 додатків.

3Д, ВЕБ-САЙТ, ГРАФІКА, ІНТЕРАКТИВНІСТЬ, ІНТЕРАКТИВНА ГРАФІКА, РОЗРОБКА, РОЗРОБКА ВЕБ-САЙТІВ.

Об’єкт дослідження – інтерактивна 3Д графіка.

Мета роботи: створення веб-сайту магазину, на якому перегляд товарів реалізовано за допомогою інтерактивної 3Д графіки.

Метод дослідження – аналітичний.

Кваліфікаційна робота містить дослідницький компонент з пошуку готових рішень поставленого завдання, тестування та аналіз відповідно до вимог до програмного продукту. Уточнення та розробка архітектури веб-сайту на основі обраного стеку технологій. Практична частина містить етапи проектування та реалізації сайту та подальше його тестування. Внаслідок аналізу типової статистики зроблено висновок, що розроблена програма справляється з завданням без критичних помилок.

SUMMARY

Bachelor’s Qualifying Theses "Development of a website using interactive 3D graphics": 80 p., 25 figures, 30 sources, 6 applications.

3D, DEVELOPMENT, GRAPHICS, INTERACTIVITY, INTERACTIVE GRAPHICS, WEBSITE, WEBSITE DEVELOPMENT.

The object of research is interactive 3D graphics.

Purpose of the study: to create a store website where the viewing of goods is realized using interactive 3D graphics.

The research method is analytical.

The qualification work contains a research component to find ready-made solutions to the task, testing and analysis in accordance with the requirements for the software product. Clarification and development of the website architecture based on the selected technology stack. The practical part includes the stages of designing and implementing the website and its further testing. Based on the analysis of typical statistics, it is concluded that the developed program copes with the task without critical errors.

ЗМІСТ

ВСТУП

Веброзробка – це динамічна галузь, яка постійно розвивається та оновлюється. Нові технології, інструменти та підходи з'являються майже щодня, і важливо бути в курсі цих змін, щоб створювати ефективні та інноваційні вебсайти. Використання інтерактивної 3D-графіки – це лише один з багатьох способів покращити користувацький досвід і зробити вебсайт більш привабливим та незабутнім.

Використання інтерактивної 3D-графіки на вебсайтах має кілька переваг. По-перше, це дозволяє користувачам побачити продукт під різними кутами, що допомагає їм краще зрозуміти, як він виглядає в реальному житті. По-друге, інтерактивні 3D-моделі можуть підвищити впізнаваність бренду, оскільки вони створюють унікальний і незабутній досвід для користувачів.

Однак розробка вебсайту з використанням інтерактивної 3D-графіки – це складний процес, який вимагає глибоких знань у сфері веброзробки та 3D-моделювання. Тому важливо детально розглянути та проаналізувати такі технології та методи їх використання:

- огляд технологій HTML, CSS, React.js, Node.js та Three.js та їх використання при розробці вебсайтів. У цьому розділі будуть розглянуті основні технології, які будуть використовуватися для розробки вебсайту, як ці технології взаємодіють між собою і як їх можна використовувати для створення якісних вебсайтів;
- розробка сайту магазину з використанням цих технологій. У цьому розділі буде розглянуто процес розробки вебсайту, кроки, які необхідно зробити для створення вебсайту, включаючи дизайн, кодування, тестування та реалізацію;
- створення інтерактивних 3D-моделей для представлення товарів на сайті. У цьому розділі буде розглянуто процес створення моделей, які будуть використовуватися на вебсайті. Також буде показано, як

використовувати функціонал Three.js для реалізації 3D-графіки на вебсайті.

Метою цієї роботи є створення вебсайту магазину, який буде використовувати інтерактивні 3D-моделі для представлення товарів. Для реалізації цього проєкту будуть використовуватися технології HTML, CSS, React.js, Node.js і Three.js.

1 АНАЛІЗ ПОСТАВЛЕНОЇ ЗАДАЧІ, ОГЛЯД НАЯВНИХ РІШЕНЬ

1.1 Теоретичні основи використання інтерактивної 3D-графіки на веб-сайтах магазинів

Веб-сайти магазинів є важливими інструментами електронної комерції, які дозволяють підприємствам продавати товари та послуги онлайн. Вони використовують різні технології та дизайнерські рішення для забезпечення зручного та ефективного користувацького досвіду.

Одним з новітніх та найбільш перспективних напрямків у веб-дизайні є використання інтерактивної 3D-графіки. За допомогою 3D-графіки можна створювати вражаючі візуальні ефекти, деталізовані моделі продуктів та інтерактивні сцени, які забезпечують більш занурювальний та ефективний користувацький досвід.

Використання 3D-графіки на веб-сайтах магазинів може мати ряд переваг. Наприклад, 3D-моделі продуктів дозволяють клієнтам бачити товар з різних ракурсів, що може допомогти їм краще оцінити його та прийняти рішення про покупку. Крім того, інтерактивні 3D сцени можуть забезпечити більш занурювальний та енгейджуючий користувацький досвід, що сприяє залученню та утриманню клієнтів.

На сьогоднішній день існує ряд технологій та інструментів, які дозволяють впроваджувати 3D-графіку на веб-сайтах. Одним з найпопулярніших є WebGL (рис. 1.1), JavaScript API, яке дозволяє створювати 3D-графіку безпосередньо у браузері. Інші популярні інструменти включають такі програми для 3D-моделювання, як Autodesk 3ds Max та Substance.

Однак, не дивлячись на всі переваги, використання 3D-графіки на веб-сайтах магазинів також має деякі виклики. Це включає в себе питання про продуктивність, сумісність браузерів, складність розробки та потребу в висококваліфікованих розробниках.

Рисунок 1.1 – Логотип WebGL та логотипи платформ які його підтримують

1.2 Веб-сайти магазинів та їх роль в електронній комерції

Електронна комерція (або e-commerce) є одним із найбільш швидко зростаючих секторів світової економіки. Веб-сайти магазинів відіграють ключову роль у цій галузі, дозволяючи підприємствам продавати товари та послуги онлайн. Вони використовують різні технології та дизайнерські підходи для забезпечення зручності та ефективності користувацького досвіду.

Дизайн веб-сайту (рис. 1.2) має важливе значення для залучення та утримання клієнтів. Добре розроблений сайт з привабливим оформленням, зрозумілою навігацією та зручним інтерфейсом покращує взаємодію з користувачем. Крім того, оптимізація для мобільних пристроїв є критично важливою, оскільки все більше людей здійснюють покупки онлайн за допомогою смартфонів та планшетів. Адаптивний дизайн забезпечує коректну роботу сайту на різних пристроях і розмірах екранів.

Каталог продукції є важливою складовою будь-якого сайту електронної комерції. Він містить детальний опис товарів, їхні ціни, зображення та

інформацію про наявність. Зручне управління каталогом спрощує пошук та придбання товарів. Функція кошика дозволяє користувачам додавати товари до віртуального кошика під час перегляду сайту, а коли вони готові зробити покупку, перейти до оформлення замовлення та здійснити безпечну оплату.

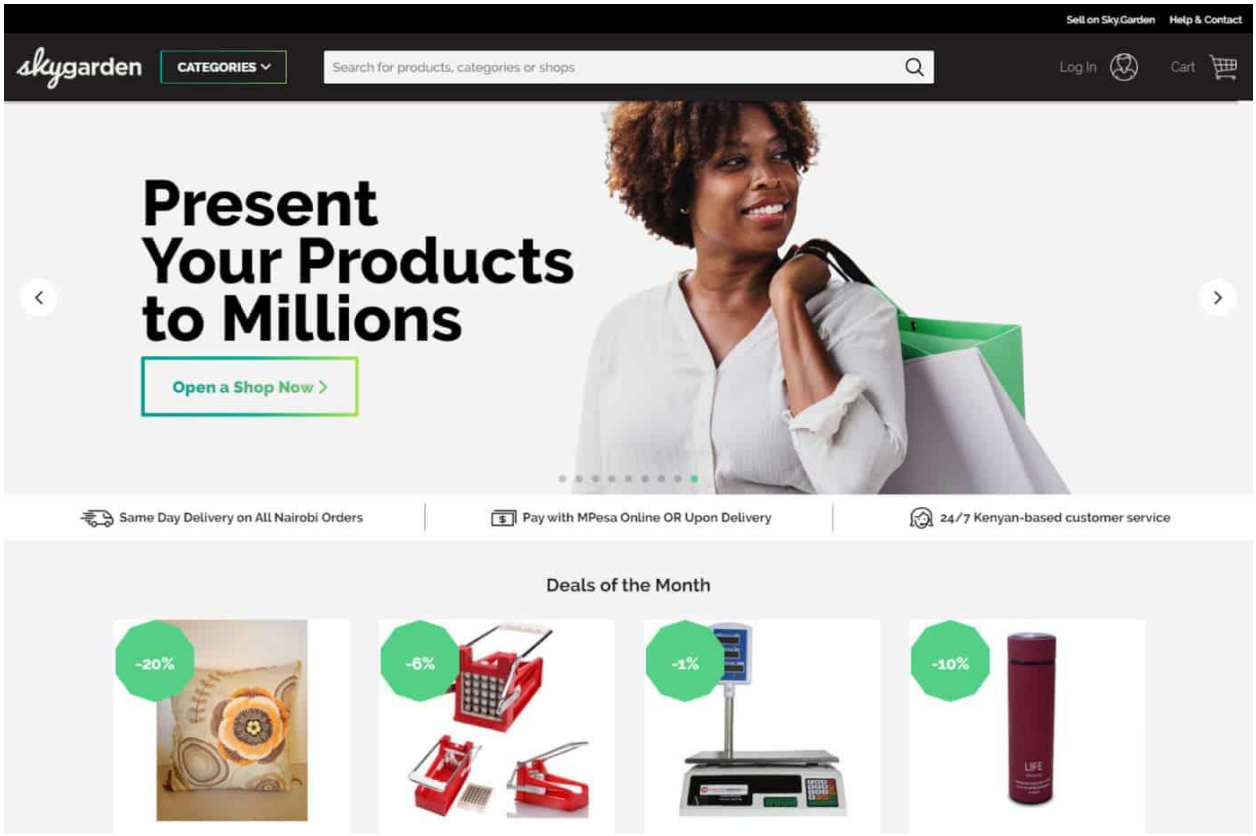

Рисунок 1.2 – Приклад дизайну веб-сайту магазину товарів

Інтеграція надійних платіжних шлюзів є важливим аспектом забезпечення безпечних онлайн-транзакцій. Це гарантує, що платіжна інформація клієнтів шифрується та захищається під час процесу оплати. Відомі та надійні платіжні шлюзи, такі як Stripe, PayPal та інші, інтегруються для надання клієнтам різноманітних варіантів оплати.

1.3 Інтерактивна 3D-графіка в веб-дизайні

Одним з новітніх та найбільш перспективних напрямків у веб-дизайні є використання інтерактивної 3D-графіки. За допомогою 3D-графіки можна створювати вражаючі візуальні ефекти, деталізовані моделі продуктів та інтерактивні сцени, які забезпечують більш занурювальний та ефективний користувацький досвід.

Інтерактивна 3D-графіка відкриває нові можливості для створення веб-сайтів, що захоплюють уяву та забезпечують більш занурювальний користувацький досвід. Використання 3D-графіки в веб-дизайні (рис. 1.3) дозволяє створювати вражаючі візуальні ефекти, деталізовані моделі продуктів та інтерактивні сцени.

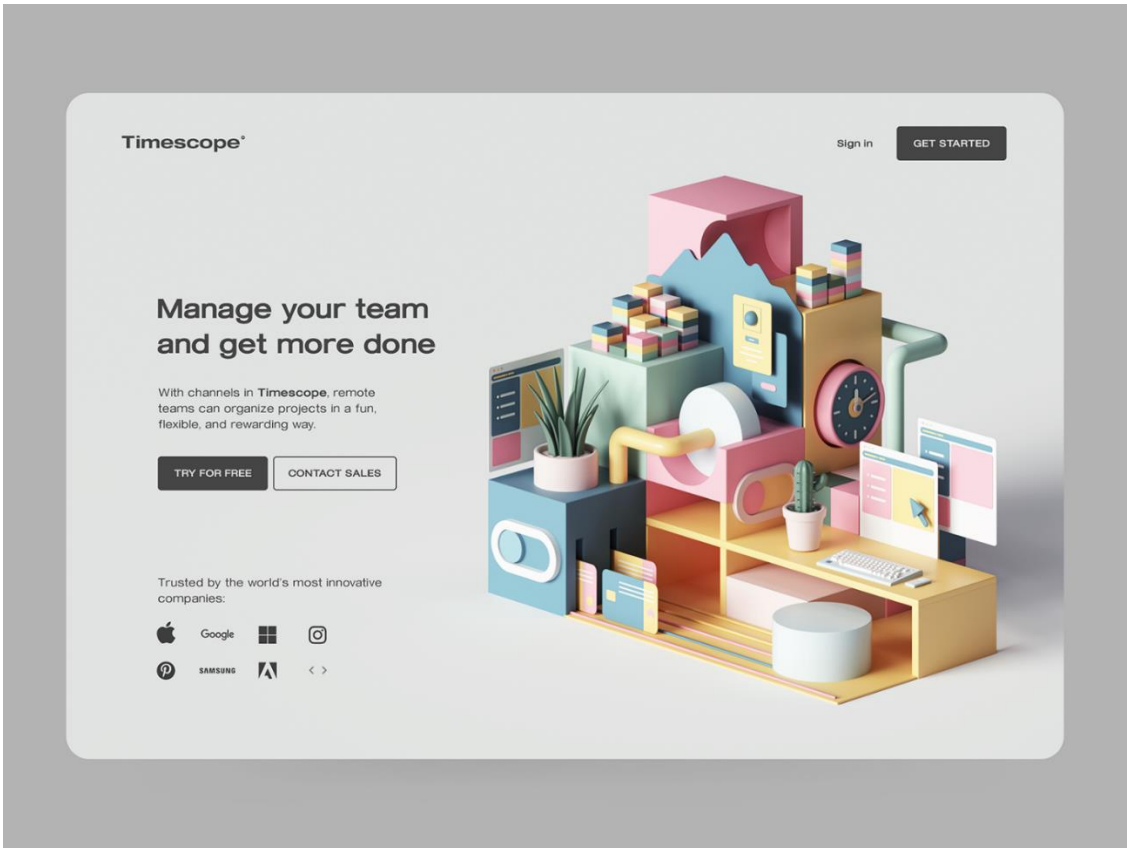

Рисунок 1.3 – Приклад сайту, на якому використовується 3D-графіка

Наразі існують такі приклади впровадження інтерактивної 3D-графіки на веб-сайтах:

– 3D-моделі продуктів, які можна використовувати для створення реалістичних зображень та анімацій, що дозволяють користувачам обертати, масштабувати та переглядати продукти з різних ракурсів. Це може бути особливо корисно для складних або дорогих продуктів, де важливо дати клієнтам чітке уявлення про те, що вони отримують;
– 3D-конфігуратори продуктів дозволяють користувачам персоналізувати продукти, вибираючи з різних варіантів кольору, матеріалу та інших характеристик. Це може допомогти збільшити продажі та покращити задоволеність клієнтів, надаючи їм більше контролю над тим, що вони купують;
– 3D-віртуальні тури (рис. 1.4) дозволяють користувачам віртуально досліджувати магазин або інші простори. Це може бути корисно для магазинів з декількома фізичними локаціями або для магазинів, які хочуть дати клієнтам відчуття того, як це – бути в магазині, не виходячи з дому.

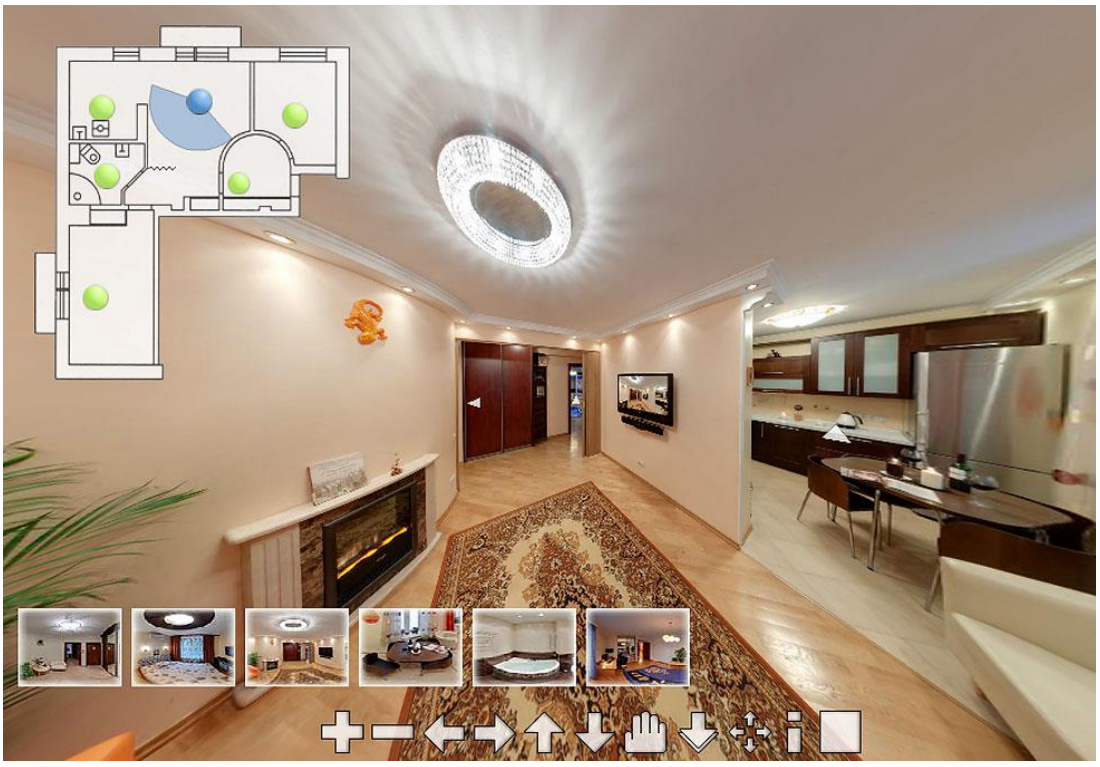

Рисунок 1.4 – Приклад 3D-віртуального туру

– 3D-ігри та інтерактивні елементи можуть допомогти залучити користувачів та зробити їхній досвід на веб-сайті магазину більш приємним. Це може бути особливо корисно для магазинів, які продають продукти для дітей або для магазинів, які хочуть створити більш захоплююче та інтерактивне середовище для покупок.

1.4 Переваги та недоліки використання 3D-графіки на веб-сайтах магазинів

Перевагами використання 3D-графіки на веб-сайтах є:

– покращення візуального досвіду: 3D-графіка може допомогти створити більш візуально привабливий та захоплюючий веб-сайт, що може призвести до збільшення конверсії та продажів;
– підвищення залученості користувачів: 3D-ігри та інтерактивні елементи можуть допомогти залучити користувачів та змусити їх довше перебувати на веб-сайті;
– покращення розуміння продукту: 3D-моделі продуктів та конфігуратори можуть допомогти клієнтам краще зрозуміти та візуалізувати продукти, перш ніж їх купити, що може призвести до меншої кількості повернень та більшого задоволення клієнтів;
– підвищення конкурентоспроможності: Використання інтерактивної 3D-графіки може допомогти магазинам виділитися на тлі конкурентів та створити більш унікальний та запам'ятовується досвід для покупок.

Але, у використання 3D-графіки на веб-сайтах є і свої недоліки, такі як:

– збільшення витрат: Розробка та впровадження інтерактивної 3D-графіки може бути дорогою;
– збільшення складності: Розробка та підтримка веб-сайту з інтерактивною 3D-графікою може бути складною;

– проблеми сумісності: Не всі браузери та пристрої однаково підтримують 3D-графіку.

Тому треба розуміти, що хоча інтерактивна 3D-графіка може бути потужним інструментом для веб-сайтів магазинів, який може допомогти покращити візуальний досвід, підвищити залученість користувачів та покращити розуміння контенту, вона також потребує більших ресурсів для впровадження та підтримки на сайті, що може бути критичним в деяких випадках.

1.5 Приклади використання інтерактивної 3D-графіки на веб-сайтах

На сьогоднішній день відомо про декілька компаній, які вже активно використовують інтерактивну 3D-графіку для комерційної діяльності на своїх веб-сайтах. Приклади таких компаній:

– Nike використовує 3D-моделі на своїй сторінці продуктів (рис. 1.5), щоб дозволити користувачам обертати, масштабувати та переглядати взуття з різних ракурсів. Це допомагає потенційним клієнтам краще візуалізувати взуття перед його покупкою;
– Warby Parker використовує 3D-конфігуратор окулярів, щоб дозволити користувачам віртуально приміряти різні окуляри. Це допомагає користувачам знайти окуляри, які їм підходять, не виходячи з дому. Але це стосується тільки звичайних окулярів, а не окулярів для покращення зору;
– Wayfair використовує 3D-віртуальні тури, щоб дозволити користувачам віртуально досліджувати меблі та декор у власному домі. Це допомагає користувачам краще уявити, як меблі виглядатимуть у їхньому просторі;

– Sephora використовує 3D-моделі продуктів та віртуальні пробники, щоб дозволити користувачам віртуально приміряти косметику. Це допомагає користувачам знайти косметику, яка їм підходить, та одразу побачити як це виглядатиме.

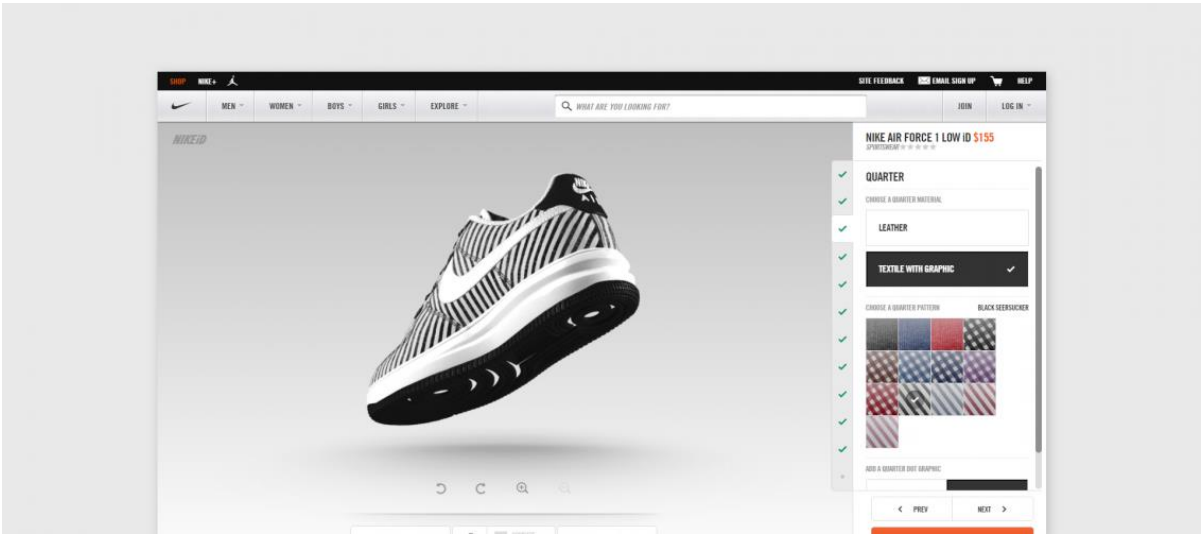

Рисунок 1.5 – Інтерфейс сайту Nike, на якому впроваджено використання інтерактивної 3D-графіки

2 АНАЛІЗ ПРОЦЕСУ РОЗРОБКИ САЙТУ ТА СТВОРЕННЯ 3D-МОДЕЛЕЙ

2.1 Основні сегменти розробки веб-сайту з використанням інтерактивної 3D-графіки

Після визначення ідеї та мети розробки веб-сайту, процес розробки можна розділити на три основні сегменти: розробка фронтенду (від англ. Front end «передній кінець» – зовнішній інтерфейс веб-застосунків), розробка бекенду (від англ. Back end «задній кінець» – внутрішній механізм веб-застосунків), та створення 3D-моделей.

Розробка фронтенду включає в себе створення користувацького інтерфейсу веб-сайту. Це включає в себе дизайн та розробку веб-сторінок, включаючи HTML, CSS та JavaScript, а також використання фреймворків та бібліотек, таких як React.js або Vue.js, для створення інтерактивних елементів та компонентів. Розробка фронтенду також включає в себе оптимізацію веб-сайту для різних пристроїв та браузерів, а також забезпечення доступності та користувацького досвіду.

Розробка бекенду включає в себе створення серверної частини веб-сайту, яка обробляє запити від клієнтської частини, керує базами даних та забезпечує безпеку та продуктивність веб-сайту. Це включає в себе використання мов програмування, таких як Node.js або Python, а також використання баз даних, таких як MongoDB або PostgreSQL. Розробка бекенду також включає в себе створення API для взаємодії з фронтендом та інтеграцію з іншими сервісами та платформами.

Створення 3D-моделей включає в себе створення 3D об’єктів та сцен, які будуть використовуватися на веб-сайті. Це включає в себе використання спеціалізованих програм для 3D-моделювання, таких як Blender або 3DsMax, для створення деталізованих 3D-моделей, текстур та анімацій. Ці моделі потім

можна експортувати в формати, які сумісні з веб-браузерами, та інтегрувати на веб-сайт за допомогою бібліотек, таких як Three.js.

2.2 Детальний огляд сегменту розробки фронтенду

Фронт-енд розробка – це галузь, що динамічно розвивається. Вона охоплює створення візуальної частини веб-сайтів та веб-додатків. Її роль у забезпеченні позитивного досвіду користувачів та загального успіху онлайн-проектів складно переоцінити.

В процесі дослідження було розглянуто та описано ключові технології, фреймворки та бібліотеки, що використовуються в цій галузі. Було наведено приклади з плюсами та мінусами кожного інструменту, що допомогло зробити обґрунтований вибір під час роботи над реалізацією проекту під час дослідження.

2.2.1 Основні технології фронтенд розробки

Якщо не брати до уваги чисельні та різноманітні фреймворки та бібліотеки, на які в наш час спираються багато розробників при створенні своїх веб-застосунків, основними компонентами фронтенд розробки можна назвати такі технології та поняття:

а) HTML (HyperText Markup Language) – це основна мова розмітки веб-сторінок, яка потрібна для визначення структури та вмісту веб-сторінок. Вона описує, як мають бути організовані різні елементи на сторінці, такі як заголовки, абзаци, зображення, списки, посилання тощо. HTML використовує теги для позначення цих елементів, які потім інтерпретуються веб-браузером для візуального відображення сторінки [1].

Переваги HTML:

– простота: HTML не потребує глибоких знань програмування, його легко вивчити та використовувати;

– універсальність: HTML підтримується всіма веб-браузерами, що робить його ідеальним для створення веб-сторінок, доступних для всіх;
– структура: HTML чітко структурує контент веб-сторінки, роблячи її зрозумілою для пошукових систем та інших інструментів.

Недоліки HTML:

– обмежена динамічність: Чистий HTML не підходить для створення динамічного контенту, який змінюється в режимі реального часу. Для цього, зазвичай, використовують деякі фреймворки;
– потреба в CSS: Для візуального оформлення HTML-елементів потрібна додаткова мова стилізації – CSS [2];

б) CSS (Cascading Style Sheets) – це мова стилізації для оформлення зовнішнього вигляду HTML-елементів. CSS дозволяє контролювати візуальні аспекти HTML-елементів, такі як кольори, шрифти, розміри, відступи, вирівнювання, фонові зображення тощо. Це робить веб-сторінки більш привабливими та зручними для користувачів. CSS використовує селектори для вибору елементів, до яких потрібно застосувати стилі, а потім описує ці стилі за допомогою властивостей та значень [3].

Переваги CSS:

– відокремлення: CSS відокремлює візуальне оформлення від структури контенту, роблячи код більш організованим та легким для обслуговування;
– контроль: CSS надає гнучкий контроль над візуальним виглядом веб-сторінок, дозволяючи створювати унікальні дизайни;
– зручність використання: CSS використовує простий синтаксис, який легко вивчити та зрозуміти.

Недоліки CSS:

– складність: Деякі складні візуальні ефекти можуть потребувати більш просунутого знання CSS;

– підтримка: Деякі старі браузери можуть не підтримувати всі можливості CSS [4];

в) JavaScript – це мова програмування для створення сценаріїв веб-сторінок. Вона потрібна для створення динамічного контенту, обробки подій, валідації форм та багато іншого. JavaScript код виконується на стороні клієнта (у браузері). Він може змінювати HTML і CSS, реагувати на дії користувача і взаємодіяти з серверами (рис. 2.1) [5].

Переваги JavaScript:

– інтерактивність: Додає динамічний контент і обробку подій на веб-сторінках;
– широка підтримка: Підтримується всіма сучасними веб-браузерами;
– екосистема: Величезна кількість бібліотек і інструментів для спрощення розробки [6].

Недоліки JavaScript:

– безпека: JavaScript код може бути вразливим до атак, таких як XSS (Cross-Site Scripting);
– продуктивність: Неналежно оптимізований JavaScript може вплинути на продуктивність веб-додатків;
– сумісність: Різні браузери можуть інтерпретувати JavaScript по-різному [7];

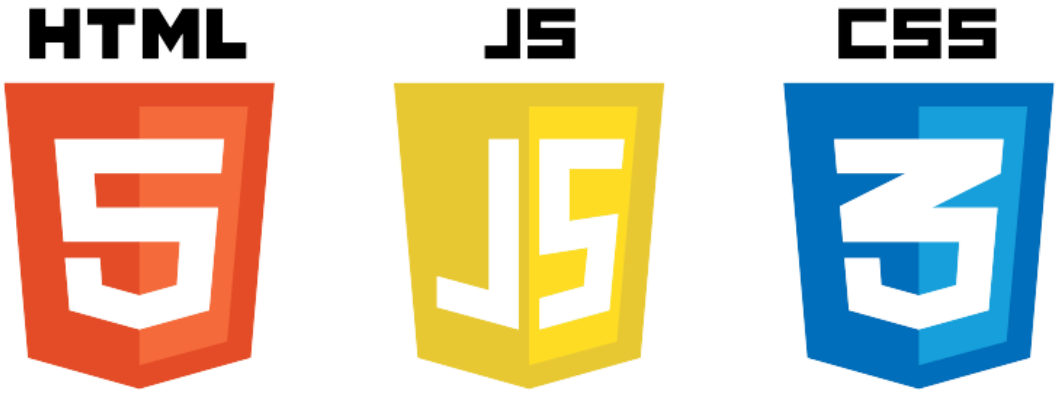

Рисунок 2.1 – Три основні технології, які використовуються в майже кожному веб-проекті: HTML, JavaScript та CSS

г) DOM (Document Object Model) – це програмний інтерфейс для документів HTML та XML. Він представляє структуру документа у вигляді дерева, де кожен вузол є об'єктом. DOM дозволяє JavaScript отримувати доступ до HTML-елементів і маніпулювати ними (додавати, видаляти, змінювати елементи) [8].

Переваги DOM:

- динамічність: Дозволяє динамічно змінювати структуру і вміст веб-сторінки;
- інтерактивність: Полегшує обробку подій (наприклад, кліки миші, введення даних).

Недоліки DOM:

- складність: Маніпуляції з DOM можуть бути складними і призводити до помилок;
- продуктивність: Часті маніпуляції з DOM можуть сповільнити роботу додатку;

д) HTTP (HyperText Transfer Protocol) і HTTPS (HTTP Secure) – це протоколи для передачі даних між веб-браузером і сервером. HTTP працює за схемою запит-відповідь: браузер відправляє запит на сервер, а сервер повертає відповідь. HTTPS додатково використовує шифрування для забезпечення безпеки передачі даних.

Переваги:

- універсальність: Використовується для передачі даних між клієнтом і сервером;
- безпека: HTTPS забезпечує шифрування даних, що покращує безпеку.

Недоліки:

- обмеження швидкості: Велика кількість запитів може сповільнити роботу додатку;
- нестабільність: Проблеми з мережею можуть впливати на доступність ресурсу;

е) AJAX (Asynchronous JavaScript and XML) – це технологія для асинхронного обміну даними між браузером і сервером без перезавантаження сторінки. AJAX використовує об'єкт XMLHttpRequest для відправки асинхронних запитів на сервер і обробки відповідей, оновлюючи лише частину веб-сторінки.

Переваги:

– асинхронність: Дозволяє оновлювати частини веб-сторінки без перезавантаження;
– покращений UX: Забезпечує більш плавну та швидку взаємодію з користувачем.

Недоліки:

– складність: Використання AJAX може ускладнити код;
– проблеми з SEO: Динамічний контент може бути проблематичним для пошукових систем;

ж) Web APIs (Application Programming Interfaces) – це інтерфейси для взаємодії з різними функціями браузера або веб-додатку.

Web APIs надають методи і властивості для виконання різних завдань, таких як доступ до геолокації, аудіо та відео, збереження даних у локальному сховищі тощо.

Переваги:

– розширені можливості: Дозволяють взаємодіяти з апаратними засобами (наприклад, камери, геолокація);
– сумісність: Багато API підтримуються основними браузерами.

Недоліки:

– сумісність: Деякі API можуть бути недоступні в певних браузерах або версіях браузерів;
– безпека: Використання деяких API може вимагати додаткових заходів безпеки.

Також, ще можна виділити такі два напрями, які не є такими технологіями в прямому сенсі, як бібліотеки або якісь мови програмування,

але натомість є наборами ідей та методик, які, при використанні під час створення веб-продукту, значно покращують кінцевий результат. Йдеться про саме такі технології, як:

- Responsive Design: це підхід до створення веб-сторінок, які добре виглядають і функціонують на різних пристроях і розмірах екрану. Responsive Design використовує гнучкі макети, медіа-запити та інші техніки для адаптації веб-сторінок до різних умов;
- Performance Optimization: це набір практик і методів для покращення швидкодії веб-сторінок. Оптимізація продуктивності включає мінімізацію файлів, кешування, оптимізацію зображень та інші техніки для зменшення часу завантаження сторінки.

І, на останок, важливо згадати таку річ як Version Control Sysrtems(VCS). VCS – це системи для відстеження змін у коді та спільної роботи над проектами. Найпопулярніша VCS – Git.

VCS зберігає історію змін у проекті, дозволяючи розробникам працювати над різними частинами коду паралельно, об'єднувати зміни і відстежувати версії.

Переваги:

- спільна робота: Полегшує співпрацю над проектами з великою кількістю розробників;
- відстеження змін: Можливість відстежувати зміни та повертатися до попередніх версій.

Недоліків у цих систем не дуже багато, головним можна назвати тільки можливість виникнення конфліктів при об'єднанні змін, особливо у великих командах.

В цілому, можна сказати, що ці технології є фундаментальними для фронтенд розробки, забезпечуючи основні можливості для створення веб-додатків та інтерфейсів.

2.2.2 Що таке фреймворки. Основні фреймворки, які використовуються для фронтенд розробки

Фреймворки – це структури, що надають базову основу для розробки програмного забезпечення. Вони містять заздалегідь написані компоненти та інструменти, що допомагають розробникам створювати додатки ефективніше та з меншою кількістю коду. Фреймворки забезпечують структуру проекту та встановлюють стандарти для коду, що сприяє кращій організації та підтримці.

Основні характеристики фреймворків:

- структура: Визначає організацію та архітектуру коду;
- компоненти: Надає готові рішення для типових задач (напр. робота з даними, управління станом);
- інструменти: Включає допоміжні інструменти для тестування, збірки, налагодження тощо;
- керівництво: Направляє розробника у використанні кращих практик і стандартів кодування;
- розширюваність: Дозволяє додавати власні компоненти та розширювати функціональність.

Фреймворки дозволяють зосередитися на бізнес-логіці, зменшують час розробки і забезпечують високий рівень повторного використання коду.

Основні фреймворки для фронтенд розробки включають React, Angular, Vue.js та Svelte. Кожен з них має свої особливості, переваги та недоліки:

а) React – це бібліотека для побудови інтерфейсів користувача, створена Facebook. Вона дозволяє створювати компонентні, декларативні та динамічні веб-додатки (рис. 2.2) [9].

React використовує компонентний підхід, де інтерфейси поділяються на окремі компоненти. Кожен компонент може мати свій стан (state) і властивості (props).

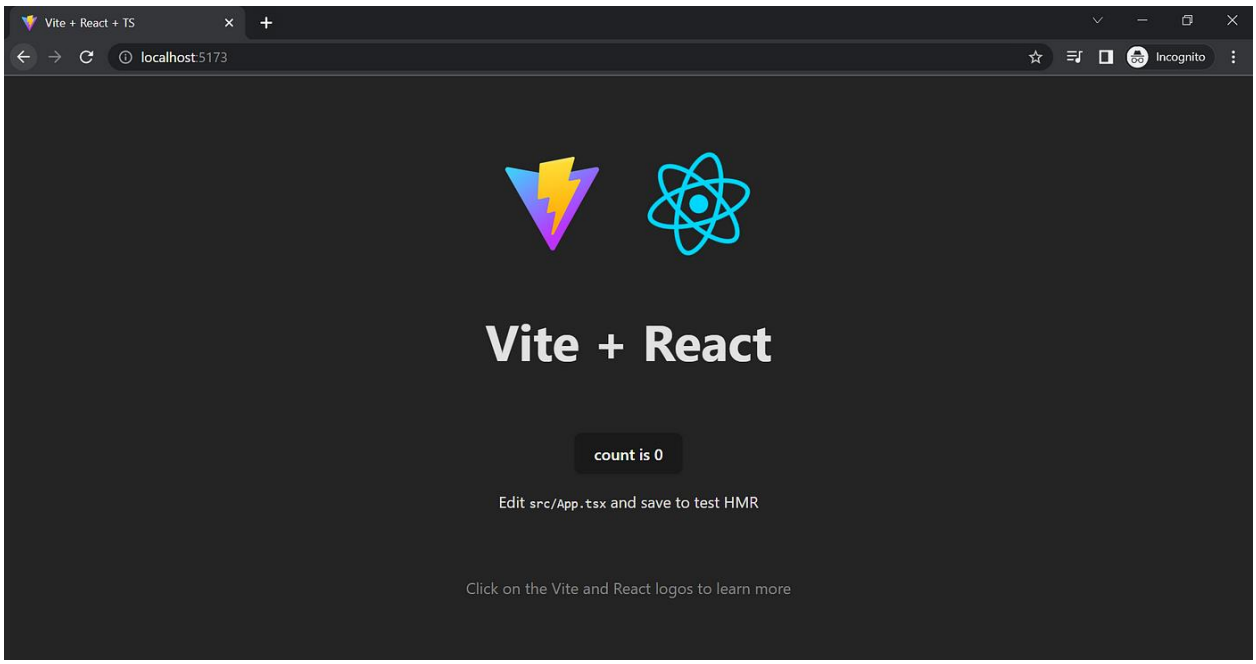

Рисунок 2.2 – Стартовий шаблон сторінки, який використовує React.js та інструмент збірки Vite.js

React використовує віртуальний DOM для ефективного оновлення інтерфейсу. [10, 11]

Переваги:

– компонентний підхід: Забезпечує повторне використання коду і полегшує управління великими додатками;
– віртуальний DOM: Підвищує продуктивність, зменшуючи кількість прямих маніпуляцій з DOM;
– велика спільнота: Велика кількість додаткових бібліотек і підтримка [12].

Недоліки:

– крута крива навчання: Потрібен час для освоєння концепцій, таких як JSX, компоненти та стани;
– відсутність стандартів: багато бібліотек і підходів можуть ускладнювати вибір оптимальних рішень [13, 14].

б) Angular – це фреймворк для побудови веб-додатків, створений Google. Він використовує TypeScript і надає повний набір інструментів для розробки складних додатків.

Angular використовує компонентний підхід і двостороннє зв'язування даних (two-way data binding). Він має потужні інструменти для створення та управління великими додатками, включаючи маршрутизацію, форми, HTTP-сервіси тощо [15].

Переваги:

– повний набір інструментів: Все, що потрібно для розробки складних додатків, включено у фреймворк;
– TypeScript: Забезпечує типізацію, що допомагає виявляти помилки на етапі розробки;
– підтримка Google: Гарантує довгострокову підтримку та розвиток.

Недоліки:

– складність: Велика кількість концепцій і інструментів може бути важкою для освоєння новачкам;
– великі розміри: Додатки на Angular можуть мати великі розміри, що може впливати на продуктивність.

в) Vue.js – це прогресивний фреймворк для побудови інтерфейсів користувача. Він легко інтегрується з іншими бібліотеками або проектами і призначений для поступового впровадження.

Vue.js використовує декларативний синтаксис для опису інтерфейсу та компонентів. Він забезпечує реактивність за допомогою двостороннього зв'язування даних і має просту інтеграцію з іншими інструментами [16, 17].

Переваги:

– простота: Легко вивчати і використовувати, з чіткою і зрозумілою документацією;
– гнучкість: Можливість використовувати Vue частково або повністю в проекті;
– реактивність: Вбудована реактивність забезпечує швидку і плавну роботу.

Недоліки:

– мала спільнота: Менша спільнота порівняно з React або Angular, що може ускладнити пошук ресурсів та підтримки;
– обмежена підтримка: Може мати менше корпоративних користувачів та підтримки порівняно з React або Angular.

г) Svelte – це сучасний фреймворк для побудови інтерфейсів користувача. Він відрізняється тим, що виконує більшість роботи під час компіляції, а не в браузері.

Svelte компілює компоненти у чистий JavaScript під час збірки проекту, що зменшує розмір і покращує продуктивність додатків [18, 19].

Переваги:

– продуктивність: Висока продуктивність за рахунок компіляції в чистий JavaScript;
– простота: Проста і зрозуміла синтаксична структура;
– маленькі розміри: Згенерований код є дуже компактним.

Недоліки:

– мала спільнота: Відносно новий фреймворк з меншою кількістю ресурсів і підтримки;
– обмежена екосистема: Можливість інтеграції з існуючими інструментами може бути обмеженою.

д) Ember.js – це фреймворк для створення амбіційних веб-додатків. Він має всеохоплюючий підхід до розробки і включає в себе багато готових рішень.

Ember.js використовує двостороннє зв'язування даних і компонентний підхід. Він надає інструменти для управління станом, маршрутизацію та інші функції, необхідні для побудови великих додатків [20, 21].

Переваги:

– зручність: Багато готових інструментів і стандартів, що полегшують розробку;
– конвенції над конфігурацією: Зменшує кількість рішень, які розробник повинен приймати;

– спільнота: Сильна і активна спільнота з хорошою документацією.

Недоліки:

– складність: Може бути складним для новачків через велику кількість вбудованих функцій;
– великий розмір: Може призвести до великих розмірів додатків, що впливає на продуктивність.

Ці фреймворки надають потужні інструменти для розробки фронтенд додатків, кожен з яких має свої унікальні особливості, переваги та недоліки. Вибір фреймворку залежить від конкретних вимог проекту та особистих уподобань розробника.

2.2.3 Що таке бібліотеки? Основні бібліотеки, які використовуються у фронтенд розробці

Бібліотеки у фронтенд розробці – це колекції готового коду, які можна використовувати для виконання певних завдань або додавання функціональності до веб-додатків. Вони забезпечують зручний і швидкий спосіб вирішення типових проблем, полегшуючи роботу розробників та прискорюючи процес розробки.

Основні характеристики бібліотек:

– спеціалізація: Бібліотеки, як правило, зосереджуються на вирішенні конкретних задач (наприклад, робота з DOM, обробка запитів HTTP);
– повторне використання: Код бібліотеки можна використовувати у багатьох проектах;
– модульність: Легко інтегруються в проекти, дозволяючи використовувати лише необхідні функції;
– полегшення роботи: Зменшують кількість написаного коду, прискорюють процес розробки та підвищують продуктивність.

Основними бібліотеками, які використовуються в фронтенд розробці можна назвати jQuery, Lodash, Axios, D3.js та Moment.js.

а) jQuery – це швидка, невелика та багатофункціональна JavaScript бібліотека, яка спрощує обробку HTML документів, обробку подій, анімації та AJAX взаємодії.

jQuery використовує простий синтаксис для вибору елементів DOM та маніпулювання ними. Вона також надає методи для обробки подій, анімацій та роботи з AJAX запитами [22].

Переваги:

- простота: Легко вивчати та використовувати завдяки простому синтаксису;
- сумісність: Вирішує проблеми сумісності між різними браузерами;
- велика спільнота: Багато плагінів та розширень.

Недоліки:

- великий розмір: Додавання jQuery до проекту збільшує розмір сторінки;
- застарілість: Багато завдань, які вирішує jQuery, тепер можна виконувати за допомогою сучасного JavaScript без використання бібліотек.

б) Lodash – це JavaScript бібліотека, яка надає зручні утиліти для роботи з масивами, об'єктами, числами, рядками тощо.

Lodash пропонує великий набір функцій, які допомагають виконувати повсякденні задачі з маніпулювання даними, такі як клонування об'єктів, глибоке копіювання, сортування, пошук, тощо.

Переваги:

- універсальність: Великий набір функцій для різних типів даних;
- продуктивність: Оптимізований код для швидкого виконання;
- модульність: Можна імпортувати тільки потрібні функції, зменшуючи розмір бандла.

Недоліки:

- залежність: Використання Lodash може збільшити розмір проекту;

– сучасні альтернативи: Багато функцій Lodash тепер доступні в нативному JavaScript;

в) Axios – це об'єктно-орієнтована JavaScript бібліотека для виконання HTTP-запитів, яка працює як на клієнтській, так і на серверній стороні.

Axios використовує проміси для обробки асинхронних запитів до серверу, надаючи простий API для відправки запитів GET, POST, PUT, DELETE та обробки відповідей.

Переваги:

– простота: Інтуїтивно зрозумілий API для роботи з HTTP-запитами;
– підтримка промісів: Легке управління асинхронними операціями;
– перехоплювачі: Можливість додавання перехоплювачів запитів та відповідей.

Недоліки:

– розмір: Використання Axios додає ваги до проекту;
– сучасні альтернативи: Fetch API в сучасних браузерах надає схожий функціонал.

г) D3.js (Data-Driven Documents) – це JavaScript бібліотека для створення динамічних, інтерактивних візуалізацій даних з використанням стандартів SVG, HTML5 та CSS3.

D3.js надає інструменти для зв'язування даних з DOM елементами та маніпулювання ними, що дозволяє створювати складні графіки та візуалізації.

Переваги:

– потужність: Можливість створювати складні та інтерактивні візуалізації;
– гнучкість: Підтримка різних форматів даних і можливість маніпулювання DOM;
– велика спільнота: Багато прикладів та документації.

Недоліки:

– складність: Вимагає значного часу на вивчення;

– виконання: Може бути повільною при обробці великих об'ємів даних.

д) Moment.js – це бібліотека для роботи з датами та часом у JavaScript, яка полегшує їх маніпулювання, форматування та відображення.

Moment.js надає функції для парсингу, валідації, маніпулювання та відображення дат та часу в різних форматах.

Переваги:

– простота: Легко використовувати для роботи з датами та часом;
– форматування: Широкий набір форматів для відображення дат;
– міжнародні формати: Підтримка локалізації.

Недоліки:

– розмір: Бібліотека доволі велика, що збільшує розмір проекту;
– альтернативи: Є нові, більш легкі бібліотеки, такі як Day.js та date-fns, які надають схожий функціонал.

Ці бібліотеки є основними інструментами для фронтенд розробки, кожна з яких забезпечує вирішення специфічних задач, полегшуючи та прискорюючи процес розробки веб-додатків.

2.2.4 Висновки стосовно технологій, які будуть використані при створенні фронтенд сегменту програмного продукту

В ході дослідження було проаналізовано основні технології, фреймворки та бібліотеки для роботи з фронтенд розробкою, а саме такі основні технології:

– HTML (HyperText Markup Language);
– CSS (Cascading Style Sheets);
– JavaScript;
– DOM (Document Object Model);
– HTTP (Hypertext Transfer Protocol) і HTTPS (HTTP Secure) ;
– AJAX (Asynchronous JavaScript and XML);

– Web APIs (Application Programming Interfaces);
– Responsive Design;
– Performance Optimization;
– Version Control Systems (VCS).

Фреймворки:

– React.js;
– Angular;
– Vue.js;
– Svelte;
– Ember.js.

Бібліотеки:

– jQuery;
– Lodash;
– Axios;
– D3.js;
– Moment.js.

Виходячи з того, що для більшості сучасних методів розробки потрібна більша частина основних технологій, згаданих вище, а решта просто роблять весь процес розробки більш комфортним, питання стоїть тільки при виборі фреймворку та бібліотек для розробки.

Для розробки фронтенду в цьому проекті було обрано фреймворк React.js, через те що він є гарним вибором для розробників всіх рівнів навичок, а також через наявний досвід роботи з ним. Також, було обрано інструмент збірки проекту Vite.js, який робить процес початкової підготовки проекту дуже компактним та зручним, а також одразу робить папку проекту Git-репозиторієм, який можна одразу загрузити на свій GitHub.

2.3 Огляд сегменту розробки бекенду

Розробка бекенду включає створення серверної частини веб-додатків, яка обробляє запити від клієнтів, працює з базами даних, виконує бізнес-логіку та забезпечує обробку даних і взаємодію з іншими сервісами. Основні технології для розробки бекенду включають мови програмування, фреймворки та бібліотеки.

2.3.1 Наявні варіанти мов програмування для бекенд розробки

Вибір мови програмування для бекенду залежить від специфіки проекту, досвіду розробників та наявних ресурсів. Основними мовами програмування, які використовуються в наш час для веб-розробки, а саме для бекенду, є:

- JavaScript (Node.js);
- Python;
- Java;
- Ruby;
- PHP;

а) JavaScript: Node.js це середовище виконання JavaScript на сервері, яке використовує рушій V8 від Google.

Node.js дозволяє використовувати JavaScript для написання серверного коду, що забезпечує асинхронне та неблокуюче виконання коду [23].

Переваги:

- єдність мови: Використання однієї мови для фронтенду і бекенду;
- висока продуктивність: Асинхронна модель підходить для обробки великої кількості одночасних запитів;
- велика спільнота: Безліч пакетів і бібліотек через npm.

Недоліки:

- складність асинхронного коду: Може бути важко управляти асинхронним кодом без відповідних знань;

– нестабільність пакетів: Велика кількість пакетів з різною якістю та підтримкою;

б) Python – це високорівнева мова програмування, відома своєю простотою та читабельністю коду.

Python використовується для написання серверного коду, який обробляє запити, взаємодіє з базами даних та виконує бізнес-логіку.

Переваги:

– простота синтаксису: Легко вивчати та писати код;
– широка екосистема: Велика кількість бібліотек і фреймворків, таких як Django та Flask;
– активна спільнота: Велика кількість ресурсів і підтримки.

Недоліки:

– менша продуктивність: Інтерпретована природа може бути повільнішою в порівнянні з компільованими мовами;
– менш підходить для високоінтенсивних завдань: Не завжди оптимальний вибір для задач, що потребують високої продуктивності;

в) Java – це об'єктно-орієнтована мова програмування, широко використовувана для розробки корпоративних додатків.

Java дозволяє писати серверний код, який запускається на JVM (Java Virtual Machine), що забезпечує високу продуктивність і портативність.

Переваги:

– портативність: Код Java може виконуватися на будь-якій платформі, де є JVM;
– висока продуктивність: Підходить для високоінтенсивних задач;
– широке використання: Популярність в корпоративному середовищі.

Недоліки:

– складність: Більш складний синтаксис та архітектура в порівнянні з іншими мовами;

– ресурсоємність: Може вимагати більше ресурсів для запуску віртуальної машини;

г) Ruby – це динамічна мова програмування, відома своєю простотою та елегантністю синтаксису.

Ruby використовується для написання серверного коду, часто разом з фреймворком Ruby on Rails, який спрощує розробку веб-додатків.

Переваги:

– продуктивність розробки: Ruby on Rails забезпечує швидку розробку завдяки конвенціям та генераторам коду;
– простота: Легко вивчати і використовувати завдяки читабельному синтаксису.

Недоліки:

– продуктивність: Може бути повільнішою в порівнянні з іншими мовами;
– менша поширеність: Менше бібліотек та ресурсів у порівнянні з іншими мовами;

д) PHP – це мова сценаріїв, яка широко використовується для розробки веб-додатків.

PHP вбудовується в HTML і виконується на сервері для обробки запитів і динамічної генерації веб-сторінок.

Переваги:

– простота використання: Легко вивчати і інтегрувати в веб-сторінки;
– широке використання: Підтримка великою кількістю хостинг-провайдерів і CMS (наприклад, WordPress).

Недоліки:

– безпека: Вимагає уважного підходу до безпеки через часті вразливості;
– продуктивність: Може бути повільнішою в порівнянні з іншими мовами.

2.3.2 Основні фреймворки для бекенд розробки

Фреймворки бекенду надають готові компоненти та структуру для розробки веб-додатків, економлячи час та зусилля розробників. Деякі з найпопулярніших фреймворків:

а) Express.js – це мінімалістичний фреймворк для Node.js, який використовується для створення серверних додатків та API.

Express.js надає простий інтерфейс для створення маршрутів, обробки запитів і відповідей, що дозволяє швидко створювати веб-сервери.

Переваги:

- простота: Легко вивчати і використовувати;
- гнучкість: Можливість додавання різних модулів та плагінів;
- велика спільнота: Велика кількість ресурсів та підтримки.

Недоліки:

- мінімалізм: Вимагає більше налаштувань та додаткових бібліотек для складніших задач;
- асинхронний код: Потрібно вміти працювати з асинхронними функціями;

б) Django – це високорівневий фреймворк для Python, який забезпечує швидку розробку та чистий, прагматичний дизайн.

Django надає потужні інструменти для роботи з базами даних, формами, маршрутами та аутентифікацією користувачів, що дозволяє швидко створювати складні веб-додатки.

Переваги:

- швидкість розробки: Вбудовані інструменти для прискорення розробки;
- безпека: Вбудовані функції для забезпечення безпеки;
- масштабованість: Підходить для великих і складних проектів.

Недоліки:

- складність: Більш крута крива навчання;

– великий розмір: Може бути надмірним для простих проектів;

в) Flask – це мікрофреймворк для Python, який забезпечує мінімальну основу для створення веб-додатків.

Flask надає базові інструменти для створення маршрутів та обробки запитів, дозволяючи розробникам додавати необхідні функції через розширення.

Переваги:

– простота: Легко вивчати і використовувати;
– гнучкість: Легко розширюється завдяки модульній структурі;
– малий розмір: Менше надмірностей у порівнянні з великими фреймворками.

Недоліки:

– обмеженість: Вимагає більше ручної роботи для реалізації складних функцій;
– менша спільнота: Менше ресурсів у порівнянні з Django;

г) Spring – це потужний фреймворк для Java, який використовується для створення корпоративних додатків.

Spring надає широкий набір інструментів для роботи з базами даних, безпекою, аутентифікацією та іншими аспектами веб-додатків.

Переваги:

– масштабованість: Підходить для великих і складних проектів;
– безпека: Вбудовані функції для забезпечення безпеки;
– широка екосистема: Багато модулів для різних задач.

Недоліки:

– складність: Вимагає значних знань для ефективного використання;
– ресурсоємність: Вимагає більше ресурсів для розробки та виконання;

д) Ruby on Rails – це фреймворк для Ruby, який забезпечує швидку розробку веб-додатків завдяки використанню конвенцій.

Rails надає багато готових рішень для роботи з базами даних, формами, аутентифікацією та іншими аспектами веб-додатків.

Переваги:

– швидкість розробки: Завдяки конвенціям та генераторам коду;
– продуктивність: Легко вивчати і використовувати;
– активна спільнота: Велика кількість ресурсів та підтримки.

Недоліки:

– продуктивність: Може бути повільнішою в порівнянні з іншими фреймворками;
– залежність від конвенцій: Може бути важко вносити нетипові зміни;

е) Laravel – це PHP фреймворк, який надає елегантний синтаксис і потужні інструменти для розробки веб-додатків.

Laravel включає інструменти для маршрутизації, роботи з базами даних, аутентифікації та інших аспектів розробки веб-додатків.

Переваги:

– елегантність: Зрозумілий і чистий синтаксис;
– швидкість розробки: Потужні інструменти для прискорення розробки;
– велика спільнота: Багато ресурсів та підтримки.

Недоліки:

– продуктивність: Може бути менш продуктивним у порівнянні з іншими фреймворками;
– залежність від версій: Часті оновлення можуть вимагати внесення змін до коду.

2.3.3 Висновки стосовно технологій, які будуть використані при створенні бекенд сегменту програмного продукту

Розробка бекенду включає широкий спектр технологій, фреймворків і бібліотек, кожен з яких має свої переваги та недоліки. Вибір конкретних інструментів залежить від специфіки проекту, вимог до продуктивності, масштабу, безпеки та інших факторів.

В ході дослідження було проаналізовано низку мов програмування та фреймворків, які підходять для розробки бекенд частини веб-додатку, виявлено їхні переваги та недоліки, і на основі цього для роботи було обрано JavaScript з програмною платформою Node.js, яка дозволяє швидко та зручно керувати бекенд складовою проекту, використовувати програмні пакети React.js та Vite.js, а також підключати програмні модулі, потрібні для створення функціоналу в веб-додатку.

Що стосується вибору бази даних (БД), в якій зазвичай знаходиться велика кількість інформації про контент, користувачів додатку тощо, в цій роботі було прийнято рішення не вводити повноцінне підключення БД, тому що це не є ціллю дослідження, та залишити бекенд сегмент веб-додатку на початковому рівні, зімітувавши контент прямо в фронтенд сегменті.

2.4 Огляд сегменту створення 3D-графіки для веб-додатку

В процесі створення та впровадження 3D-графіки у веб-додаток можна виділити такі дві стадії: безпосередньо створення якісної 3D-моделі, з урахуванням моделювання, створення текстур тощо, і інтеграція цієї моделі на сайт з використанням фреймворків або інших технологій.

2.4.1 Аналіз процесу створення 3D-моделей для подальшого використання їх у веб-додатку. Огляд наявних прикладів програмного забезпечення

Створення моделей включає в себе декілька етапів, таких як:

а) моделювання: на цьому етапі створюється базова форма об’єкта. Це може бути зроблено за допомогою різних методів, таких як полігональне моделювання, скульптування або використання сплайнів. Полігональне моделювання передбачає створення об'єкта з багатокутників (полігонів), які

разом утворюють його поверхню. Скульптування дозволяє моделювати об'єкт, як якщо б він був зроблений з глини, створюючи деталі шляхом додавання і видалення матеріалу. Сплайнове моделювання використовує криві для визначення форми об'єкта;

б) текстурування: після створення базової форми об’єкта, на нього наносяться текстури. Текстури – це двовимірні зображення, які накладаються на поверхню 3D-моделі для надання їй деталізації, кольору та реалістичності. Процес текстурування включає розгортку UV-карт, де 3D поверхня об’єкта розгортається в 2D простір для зручного накладання текстур;

в) освітлення: на цьому етапі створюється система освітлення сцени. Світло відіграє ключову роль у створенні реалістичних зображень, так як воно визначає, як будуть виглядати матеріали і текстури при різних умовах освітлення. Розміщуються джерела світла, налаштовуються їх інтенсивність, колір та напрямок;

г) рендеринг: рендеринг є останнім етапом, на якому 3D-модель перетворюється на двовимірне зображення або анімацію. Рендеринг може бути реалістичним (фотореалістичний рендеринг) або стилізованим, в залежності від поставлених завдань. Процес рендерингу включає обчислення світла, тіней, відбиттів та інших ефектів для створення кінцевого зображення.

Проте, рендеринг не буде відігравати важливої ролі в цьому проекті, бо зображення будуть використовуватись тільки як додатковий спосіб демонстрації продукту на веб-сайті, окрім головного – із використанням саме 3D-моделей.

Що стосується програмного забезпечення для створення 3D-графіки, можна навести такі приклади:

– Autodesk 3ds Max – це універсальний комплексний продукт для 3D-моделювання, візуалізації та анімації. Він використовується в архітектурі, промисловому дизайні, ігровій та кіноіндустрії. Autodesk 3ds Max підтримує різні формати файлів, включаючи 3DS, FBX, OBJ, DWG [24, 25];

– Blender – це безкоштовна програма з відкритим кодом та повним набором інструментів для створення 3D-моделей, анімації, композингу та рендерингу (рис. 2.3). Blender підтримує багато форматів файлів (OBJ, FBX, STL тощо);

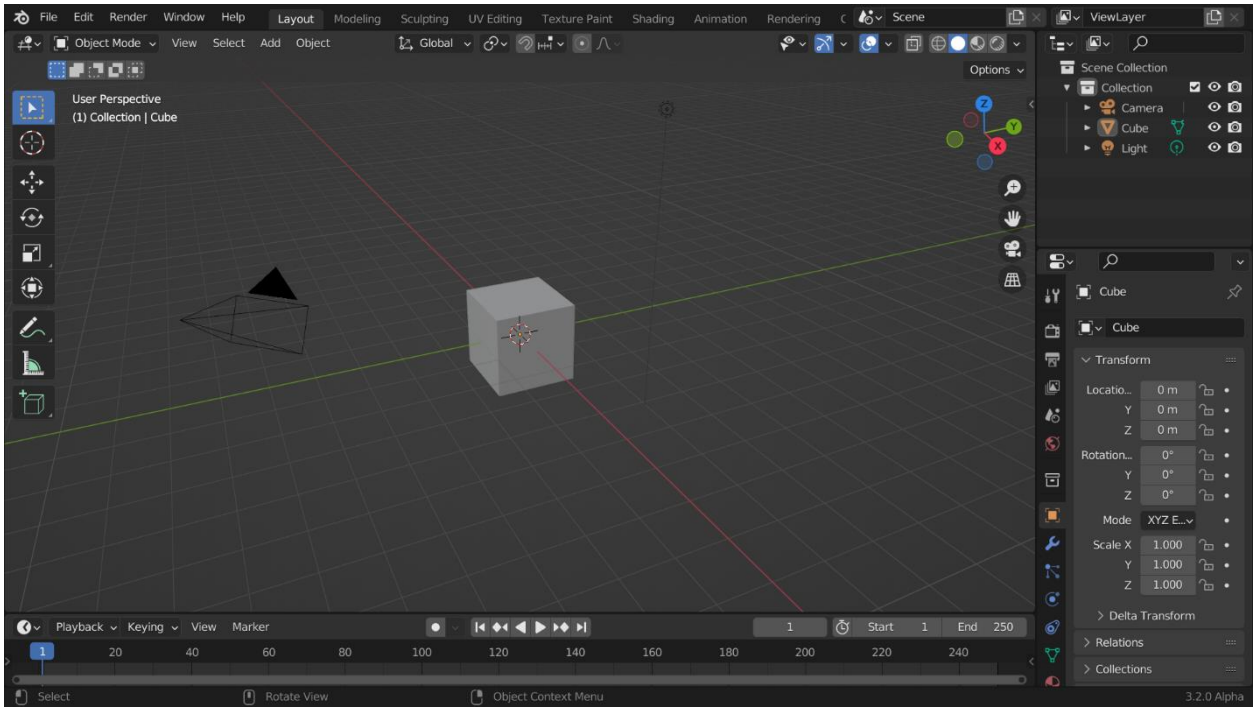

Рисунок 2.3 – Інтерфейс програми Blender

– SketchUp – це проста у використанні програма з інтуїтивно зрозумілим інтерфейсом, призначена для візуалізації та створення 3D-моделей будь-яких об’єктів, включаючи будівлі, меблі, ландшафтні об’єкти тощо. SketchUp має безкоштовну та платну версії;
– ZBrush – це програма для створення 3D-моделей різних рівнів складності. Широко використовується у відеоіграх, кіноіндустрії, мультфільмах, дизайні тощо.

2.4.2 Аналіз процесу впровадження 3D-графіки на веб-сайт. Огляд варіантів технологій для цього

Після створення 3D-моделей їх можна інтегрувати на веб-сайти для покращення взаємодії з користувачами, презентацій продукції, навчання та інших цілей. Основні етапи впровадження 3D-моделей на сайти це:

а) оптимізація: оптимізація 3D-графіки для веб є важливим аспектом розробки, який впливає на продуктивність та користувацький досвід. Цей процес включає в себе ряд стратегій та методів, які допомагають забезпечити плавність, швидкість та якість 3D-графіки.

Текстури часто займають значний обсяг даних в 3D-графіці. Використання стиснених форматів текстур може значно зменшити розмір файлів, що прискорює завантаження та зменшує використання пам’яті.

LOD (Level of Detail) – це техніка, яка зменшує кількість полігонів в моделях, які далеко від камери (рис. 2.4). Це дозволяє зберегти ресурси та підтримувати високу продуктивність, зберігаючи при цьому візуальну якість.

Існують різні інструменти та сервіси, які можуть допомогти в оптимізації 3D-графіки для веб. Ці інструменти можуть автоматизувати процеси, такі як стиснення текстур, оптимізація геометрії, видалення невикористовуваних матеріалів та інше.

Оптимізація рендерингу включає в себе техніки, такі як використання правильних налаштувань рендерера, оптимізація освітлення та тіней, використання технік рендерингу, які зменшують навантаження на графічний процесор.

Використання веб-стандартів, таких як WebGL, дозволяє забезпечити сумісність з різними браузерами та пристроями.

Це також дозволяє використовувати потужність графічного обладнання користувачів для створення захоплюючих 3D досвідів.

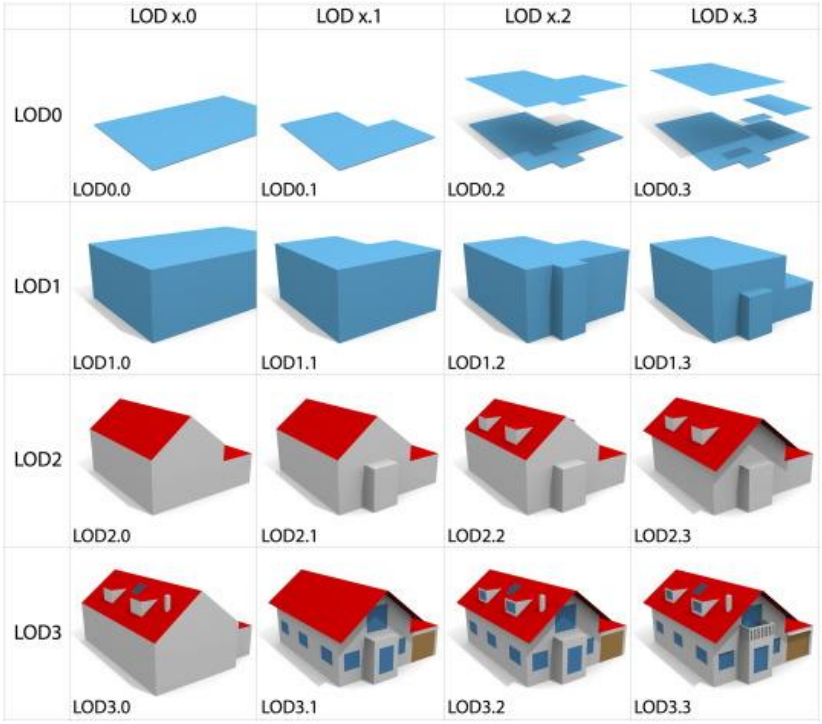

Рисунок 2.4 – Приклад функціонування технології LOD

Ці методи допомагають забезпечити, що 3D-графіка на веб-сайті буде працювати плавно та ефективно, незалежно від пристрою користувача або швидкості його інтернет-з’єднання.

б) вибір технології для рендерингу: існує кілька технологій, які можна використовувати для рендерингу 3D-моделей на веб-сайтах, такі як WebGL, Three.js, Babylon.js та інші. WebGL (Web Graphics Library) є потужним інструментом для рендерингу інтерактивної 3D-графіки безпосередньо в браузері без необхідності встановлення додаткового програмного забезпечення [26, 27].

Three.js – це відкрита JavaScript-бібліотека для створення 3D-графіки в веб-браузерах. Вона легка у використанні, що робить її популярним вибором для розробників, які тільки починають працювати з 3D програмуванням або створюють простіші додатки. Однією з основних переваг Three.js є її простота. Вона має невеликий API, який легко вивчити та використовувати, і вона може бути інтегрована з іншими JavaScript-бібліотеками та фреймворками [28, 29, 30].

Babylon.js – це ще одна відкрита JavaScript-бібліотека для створення 3D-графіки в веб. На відміну від Three.js, Babylon.js спеціально розроблена для розробки ігор та пропонує більш передові функції та функціональність. Однією з основних переваг Babylon.js є вбудовані фізичні та анімаційні двигуни. Це полегшує створення реалістичних рухів та взаємодії між об’єктами, що є важливим для створення захоплюючих ігор та симуляцій.

Обидві бібліотеки, Three.js та Babylon.js, є відмінними виборами для створення 3D-графіки на веб. Three.js легка у використанні, що робить її відмінним вибором для початківців або для простіших додатків. Babylon.js, з іншого боку, є більш складною та функціональною, і спеціально розроблена для розробки ігор та інших захоплюючих додатків.

Існують і інші бібліотеки та інструменти для роботи з 3D у веб, такі як A-Frame, PlayCanvas, GLAM.js, CopperLicht, і т.д. Вибір конкретного інструменту залежить від конкретних потреб проекту, вимог до продуктивності, сумісності з браузерами, а також від досвіду розробника.

в) інтеграція: інтеграція 3D-моделей на сайт включає в себе написання коду, що забезпечує завантаження та відображення моделі. Це може бути зроблено за допомогою JavaScript, HTML5 та CSS. Моделі можуть бути інтегровані як частина інтерфейсу користувача або окремі елементи, такі як інтерактивні продукти на сторінці інтернет-магазину.

г) налаштування взаємодії: після інтеграції 3D-моделей необхідно налаштувати взаємодію з користувачем. Це може включати обертання, масштабування, переміщення моделей, а також додавання анімацій та інтерактивних елементів. Інтерактивність забезпечує кращу взаємодію з користувачами та може бути досягнута за допомогою бібліотек для роботи з подіями миші або сенсорного екрана.

2.4.3 Висновок щодо технологій створення та впровадження 3D-графіки, які будуть використані

Для безпосереднього створення 3D-моделей для цього проекту було обрано програму Autodesk 3DsMAX, бо вона пропонує потужний набір інструментів для моделювання, текстурування та анімації, має зручний та налаштовуваний інтерфейс, підтримує фотореалістичний рендеринг з вбудованим Arnold та сторонніми рендерами (наприклад, Corona Render), а найголовніше те, що є досвід роботи з цією програмою.

Для інтеграції моделей було обрано бібліотеку Three.js. Цей вибір було зумовлено декількома ключовими факторами.

По-перше, Three.js є відмінним вибором для початківців, завдяки своїй простоті та інтуїтивно зрозумілому API. Це робить її доступною для розробників, незалежно від їхнього досвіду роботи з 3D-графікою. Крім того, Three.js має активну спільноту та багатий набір навчальних матеріалів, що сприяє швидкому освоєнню технології.

По-друге, Three.js має достатній функціонал, щоб задовольнити потреби розробки даного проекту. Вона підтримує широкий спектр геометрій, матеріалів, світлових джерел, камер та інших об’єктів, необхідних для створення інтерактивних 3D сцен. Крім того, Three.js дозволяє інтегрувати 3D-графіку з HTML та CSS, що робить її ідеальною для використання в веб-дизайні.

Таким чином, вибір Three.js для реалізації цього проекту було обґрунтовано її простотою, доступністю, гнучкістю та потужним функціоналом, який відповідає потребам розробки веб-сайту з використанням інтерактивної 3D-графіки.

3 РЕАЛІЗАЦІЯ ПРОЕКТУ

3.1 Створення основи сайту. Розробка фронтенду

Як було зазначено в попередньому розділі, проект було поділено на декілька етапів розробки, а саме:

- розробка фронтенду;
- розробка бекенду;
- розробка 3D-моделей.

Код усього проекту представлено в додатку А.

Спочатку було прийнято рішення розробити каркас сайту, на основі якого вже потім можна буде додавати новий функціонал, модулі, ідеї тощо. Так як кінцевим продуктом є веб-додаток, було використано технології React.js та Node.js. За основу було взято шаблон додатку з інструменту збірки Vite.

Структуру проекту було вирішено зробити типовою для додатків на React.js:

а) папка src – є кореневою папкою проекту, в якій зберігається весь вихідний код проекту;

б) папка public – папка, в якій зберігаються всі складі елементи, такі як 3D-моделі, які в подальшому будуть використовуватись на сайті. Все, що не буде знаходитись в цій папці, буде розумітись React.js як модулі, тому буде визивати помилки;

в) папка node_modules – папка, в якій зберігаються всі модулі Node.js та інші завантажуванні бібліотеки та модулі тощо;

г) папки assets, components, pages та syles – папки, в яких зберігаються відповідно якісь графічні ресурси (assets); React-компоненти, які використовуються на сайті (зазвичай це файли з розширенням .jsx, які потім імпортуються в файли сторінок); сторінки (файли веб-сторінок з розширенням

.jsx, між котрими можлива навігація за допомогою модуля Router); стилі (зазвичай файли з розширенням .css, в яких описані CSS стилі для веб-сторінок та компонентів);

д) файл main.jsx – є точкою входу в додаток, відповідає за рендеринг головного компонента у DOM.

Як ми можемо бачити (рис. 3.1), файл main.jsx потрібен для того, щоб правильно оформити роботу файлу додатку – App.jsx.

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

Рисунок 3.1 – Код файлу main.jsx

е) Файл App.jsx – основний компонент додатку, який об’єднує всі компоненти та маршрути в додатку. В ньому прописана логіка компоненту Router цього проекту.

Як можна побачити (рис. 3.2), в файл App.jsx було імпортовано пакет react-router-dom, а саме модулі BrowserRouter, Router та Route, які допомагають налаштувати навігацію між сторінками сайту. Так як це є тестовим дослідницьким проектом, в загальному вигляді було реалізовано тільки основні сторінки MainPage, CartPage, ProfilePage, LoginPage та RegisterPage, та сторінки продуктів ProductPage1, ProductPage2, ProductPage3.

```
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import ScrollToTop from './components/ScrollToTop.jsx';
import MainPage from './pages/MainPage';
import ProductPage1 from './pages/ProductPage1';
import ProductPage2 from './pages/ProductPage2';
import ProductPage3 from './pages/ProductPage3';
import CartPage from './pages/CartPage';
import ProfilePage from './pages/ProfilePage';
import LoginPage from './pages/LoginPage';
import RegisterPage from './pages/RegisterPage';

function App() {

  return (
    <div>
      <Router>
        <ScrollToTop/>
        <Routes>
          <Route path="/" element={<MainPage />} />
          <Route path="/product1" element={<ProductPage1/>}/>
          <Route path="/product2" element={<ProductPage2/>}/>
          <Route path="/product3" element={<ProductPage3/>}/>
          <Route path="/cart" element={<CartPage/>}/>
          <Route path="/profile" element={<ProfilePage/>}/>
          <Route path="/login" element={<LoginPage/>}/>
          <Route path="/register" element={<RegisterPage/>}/>
        </Routes>
      </Router>
    </div>
  )
}

export default App
```

Рисунок 3.2 – Код файлу App.jsx

Для того, щоб покращити розуміння коду та зробити його компактнішим, було прийнято рішення створити декілька React-компонентів, які використовуються майже на всіх сторінках ресурсу, а саме:

ж) хедер (Header.jsx) – це верхня частина веб-сторінки, яка зазвичай містить логотип компанії, навігаційне меню на основні розділи сайту, пошуковий рядок та кнопки або іконки для доступу до кошику, особливого кабінету користувача та інших важливих функцій (зис. 3.3).

Код файлу Header.jsx представлено в додатку Б.

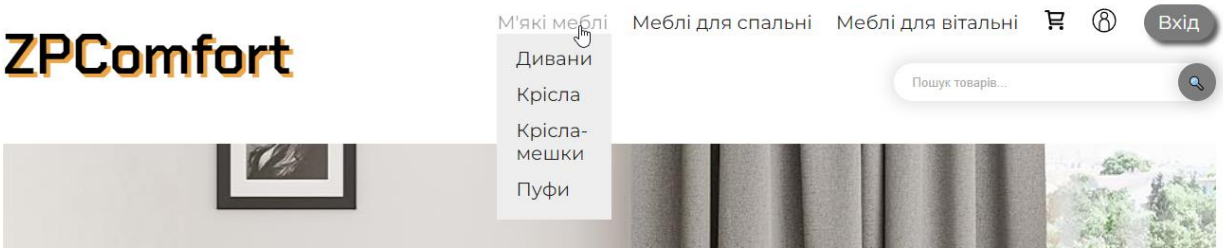

Рисунок 3.3 – Вигляд компоненту Header на сайті

и) футер (Footer.jsx) – це нижня частина веб-сторінки, яка зазвичай містить інформацію про компанію, контактну інформацію, посилання на соціальні мережі тощо (рис. 3.4).

Код файлу Footer.jsx представлено в додатку В.

Рисунок 3.4 – Вигляд компоненту Footer на сайті

к) картки товару (ProductList.jsx) – це елементи на сторінці каталогу або пошуку, які показують основну інформацію про товари, таку як назва, зображення товару, виробник, ціна, характеристики тощо. В проекті було реалізовано компонент ProductList.jsx, який є контейнером для динамічної кількості компонентів ProductCard.jsx, які і є картками товарів. Інформація про товари передається компоненту в масиві, бо повноцінного бекенду з підключенням до бази даних товарів не було реалізовано через те, що це не є основною ціллю дослідження (рис. 3.5).

Код файлу ProductList.jsx представлено в додатку Г.

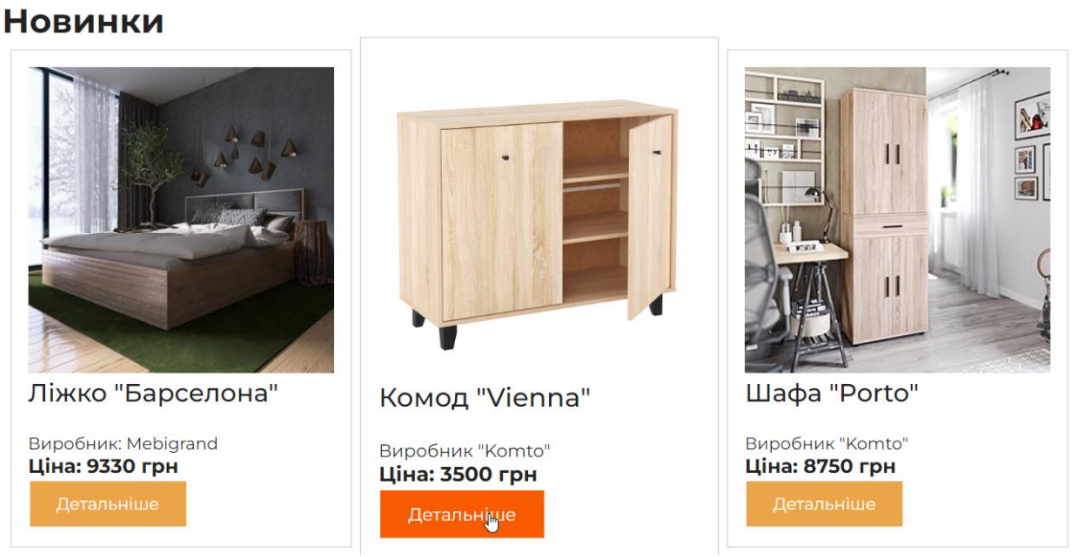

Рисунок 3.5 – Вигляд компоненту ProductList на сайті

л) банер на головній сторінці сайту (Banner.jsx) – великий візуальний елемент на головній сторінці сайту, який привертає увагу користувачів і зазвичай містить важливу інформацію або рекламні повідомлення. Банер може динамічно змінювати зображення, що дозволяє показувати кілька промоакцій, нових товарів або інші важливі повідомлення.

Код файлу Banner.jsx представлено в додатку Д.

м) модуль показу 3D-моделей (FurnitureShopContainer.jsx) – головний модуль додатку, який відповідає за демонстрацію 3D-моделей товарів та забезпечує інтерактивність та взаємодію користувача з моделлю. Модуль

складається з двох контейнерів – FurnitureModel та ProductDescription. В контейнері FurnitureModel за допомогою функціоналу бібліотеки Three.js створено елемент Canvas, на якому демонструється 3D-модель товару. Засобами цієї бібліотеки реалізовано освітлення, завантаження та ініціалізація моделей у форматі .glb, опрацювання текстур, анімацій, відтворення цих анімацій, а також керування моделлю курсором та лівою і правою кнопками миші. У випадку, якщо є якась помилка у шляху до моделі, або моделі не існує в гіпотетичній базі (у випадку цього тестового проекту – у директорії з 3D-моделями), передбачено демонстрацію моделі звичайного куба, який сигналізує про цю проблему. Динамічність зображення у 3D реалізовано через елементи бібліотеки Three.js, а саме через AnimationMixer та Clock, які забезпечують зміну кадрів через фіксовану кількість часу, що й створює ефект плавної анімації.

Як ми можемо побачити (рис. 3.6), створюється елемент loader класу GLTFLoader, який забезпечує завантаження моделей у форматі GLTF та GLB для подальшої роботи з ними. Формат GLB було обрано через те, що він підходить для роботи з 3D у WEB, і представляє собою не лише саму модель, а повноцінну сцену, з текстурами та анімаціями, що дуже підходить для цілей роботи.

Далі можна побачити перевірку на некоректний шлях до моделі: у консоль виводить повідомлення про помилку, а на Canvas демонструється звичайний куб з текстурою, яка повідомляє про помилку (рис. 3.7).

У кінці описана функція animate, яка відповідає за безперервну анімацію сцени, синхронізованою з часом, оновлює всі потрібні елементи такі як mixer (AnimationMixer для анімацій) та controls (для керування моделлю за допомогою миші).

```
const loader = new GLTFLoader();
   loader.load(
     modelPath,
     (gltf) => {
       gltf.scene.position.set(0, -0.33, 0);
       scene.add(gltf.scene);
       const mixer = new AnimationMixer(gltf.scene);
       mixerRef.current = mixer;
       if (gltf.animations && gltf.animations.length) {
         const action = mixer.clipAction(gltf.animations[0]);
         action.paused = true;
         actionRef.current = action;
         setModelLoaded(true);
       }
     },
     undefined,
     (error) => {
       console.error(error);

       const geometry = new THREE.BoxGeometry(1, 1, 1);
       const textureLoader = new THREE.TextureLoader();
       const texture = textureLoader.load(texturePath);
       const material = new THREE.MeshBasicMaterial({ map: texture });
       const cube = new THREE.Mesh(geometry, material);
       cube.castShadow = true;
       cube.receiveShadow = true;
       scene.add(cube);
     }
   );
…
const animate = function () {
     requestAnimationFrame(animate);
     const delta = clock.current.getDelta();
     if (mixerRef.current) {
       mixerRef.current.update(delta);
     }
     controls.update();
     renderer.render(scene, camera);
   };
```

Рисунок 3.6 – Фрагмент коду компонента FurnitureShopContainer.jsx

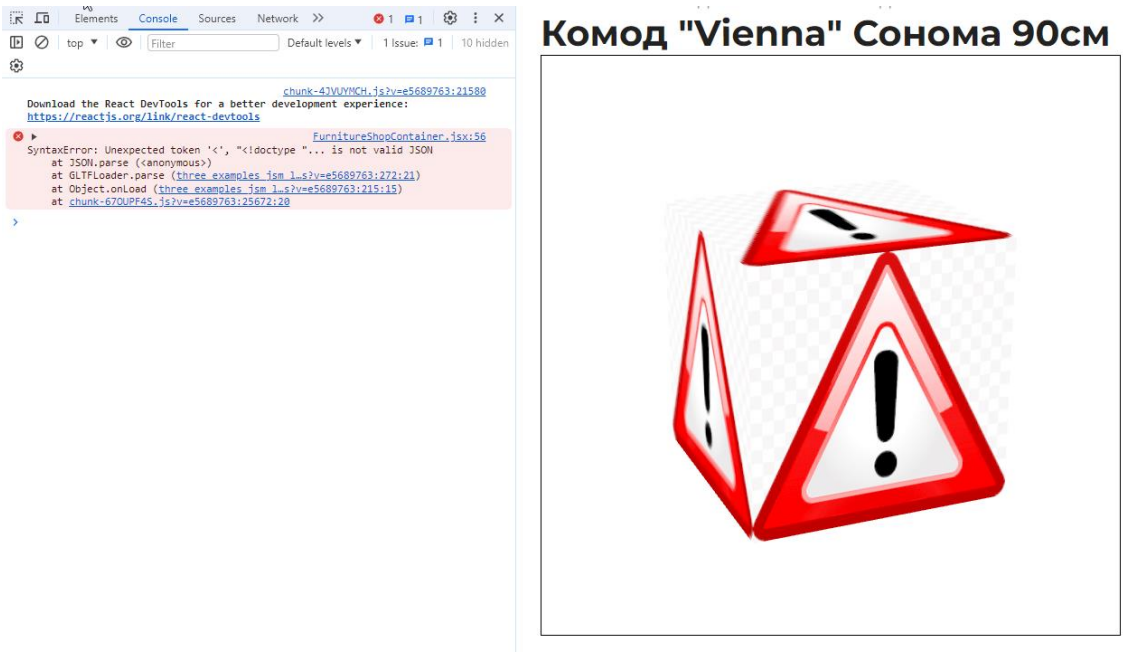

Рисунок 3.7 – Так виглядає компонент FurnitureModel, якщо шлях до моделі вказано некоректно

Для керування відтворенням анімації, яка була завантажена разом з моделлю, було прописано логіку, яка працює при натисканні кнопки, розташованої в контейнері ProductDescription.

Як ми бачимо (рис. 3.8), логіка розрахована на анімації довжиною в 30 кадрів, і на кожне натискання кнопки буде проходити 15 кадрів, тобто кожні два натискання анімація проходить повний цикл і повертається на початок.

У контейнері ProductDescription також указано опис товару, його характеристики тощо. Під описом розташовано кнопку, яка керує анімацією моделі (рис. 3.9).

Код компоненту FurnitureShopContainer.jsx представлено в додатку Е.

```
useEffect(() => {
   if (playAnimation && actionRef.current && modelLoaded) {
     const framesToPlay = 15;
     const frameDuration = 1 / 30;
     const currentTime = (frameCount * framesToPlay * frameDuration) %
actionRef.current.getClip().duration;

     actionRef.current.paused = false;
     actionRef.current.time = currentTime;
     actionRef.current.play();

     setTimeout(() => {
       actionRef.current.paused = true;
       onAnimationEnd();
     }, framesToPlay * frameDuration * 1000);
   }
 }, [playAnimation, frameCount, modelLoaded, onAnimationEnd]);
```

Рисунок 3.8 – Фрагмент коду компонента FurnitureShopContainer.jsx, який відповідає за запуск анімації при натисканні на кнопку

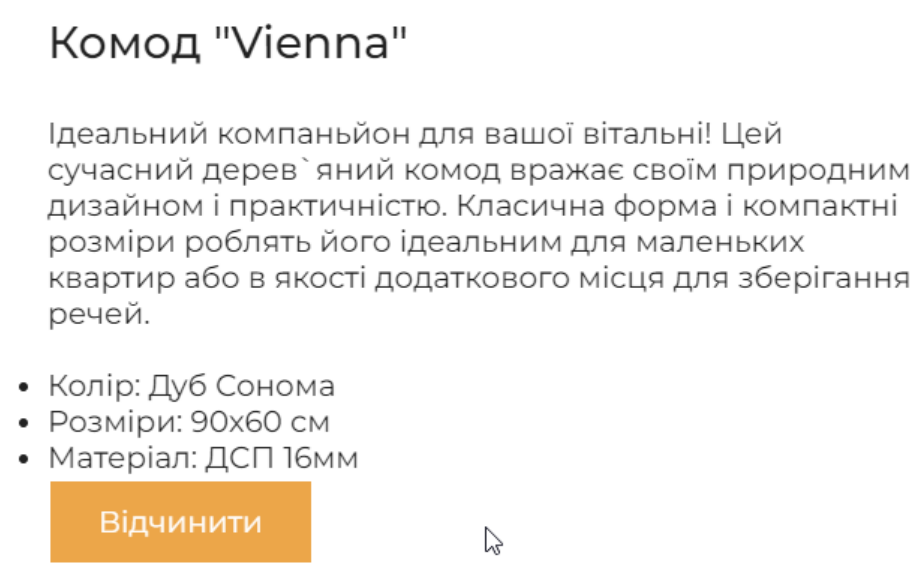

Рисунок 3.9 – Контейнер ProductDescription на сайті

3.2 Процес створення 3D-моделей. Завантаження моделей на сайт

Як було зазначено в попередніх розділах, для створення 3D-моделей товарів було використано програмне забезпечення Autodesk 3DsMAX. Першим етапом було моделювання, створення так званого мешу (від англ. Mesh «сітка» – основна форма моделі, яка представляє собою множину вершин (vertex), ребер (edges) і граней (faces), які разом утворюють поверхню тривимірного об'єкта.) Першим створеним об’єктом є комод. Для нього була створена UV-розгортка (від координат u та v на площині текстури) та налаштовані UV-координати, що дозволяє проводити коректне текстурування.

Далі йде етап накладання текстур (рис. 3.10). Для нього були обрані матеріали деревини дубу для основних панелей, чорний матовий метал для ручок та ніжок, та простий метал для петель дверей.

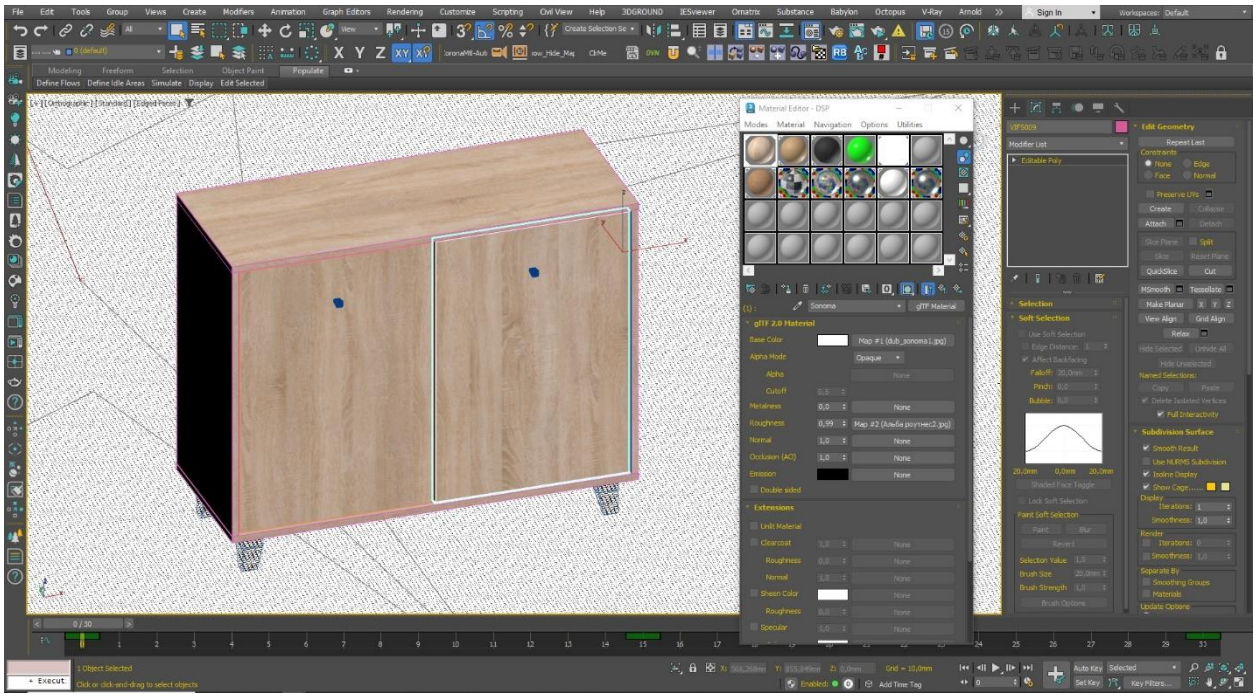

Рисунок 3.10 – Процес накладання текстур на модель

Наступним етапом є створення анімацій, яке робиться в 3DsMAX за допомогою технології Key Frames – відмічання ключових кадрів анімації, на яких змінюються параметри моделі, такі як координати в просторі та кути

повороту, а потім обирається метод переходу між станами моделі у ключових кадрах. Процес анімації керується на шкалі часу знизу, а ключові кадри помічені зеленим кольором. Для кращої точності анімації камеру було закріплено зверху або збоку, а також увімкнено Wire Mode («режим дроту») – модель представлено у вигляді дротового каркасу, без текстур та граней (рис. 3.11, 3.12, 3.13).

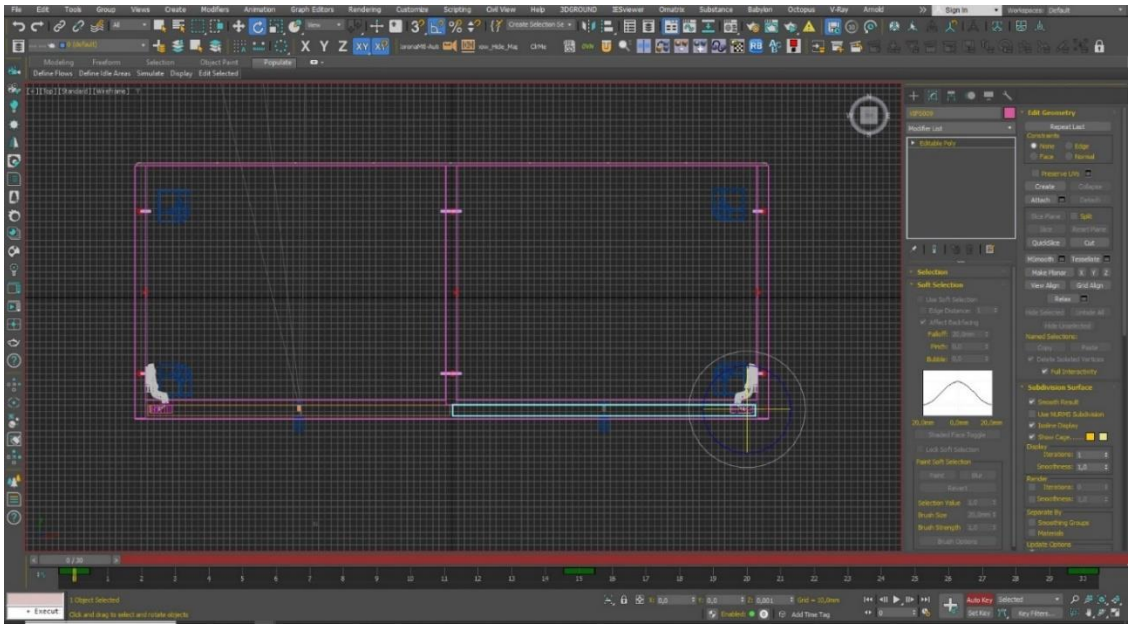

Рисунок 3.11 – Процес анімації моделі, вид зверху

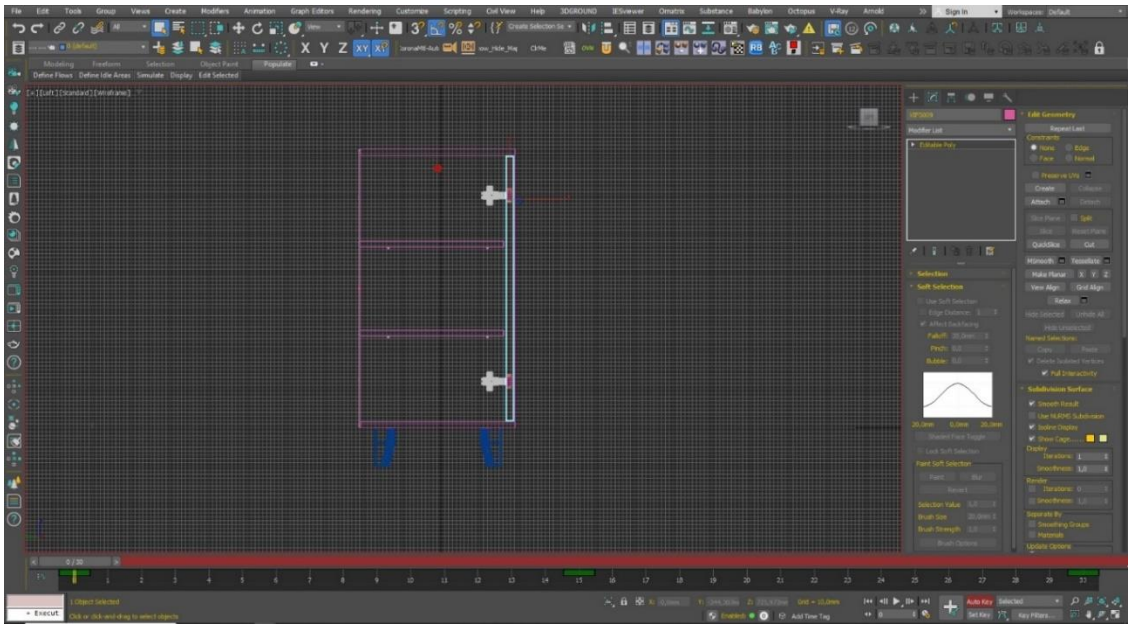

Рисунок 3.12 – Процес анімації, вид збоку

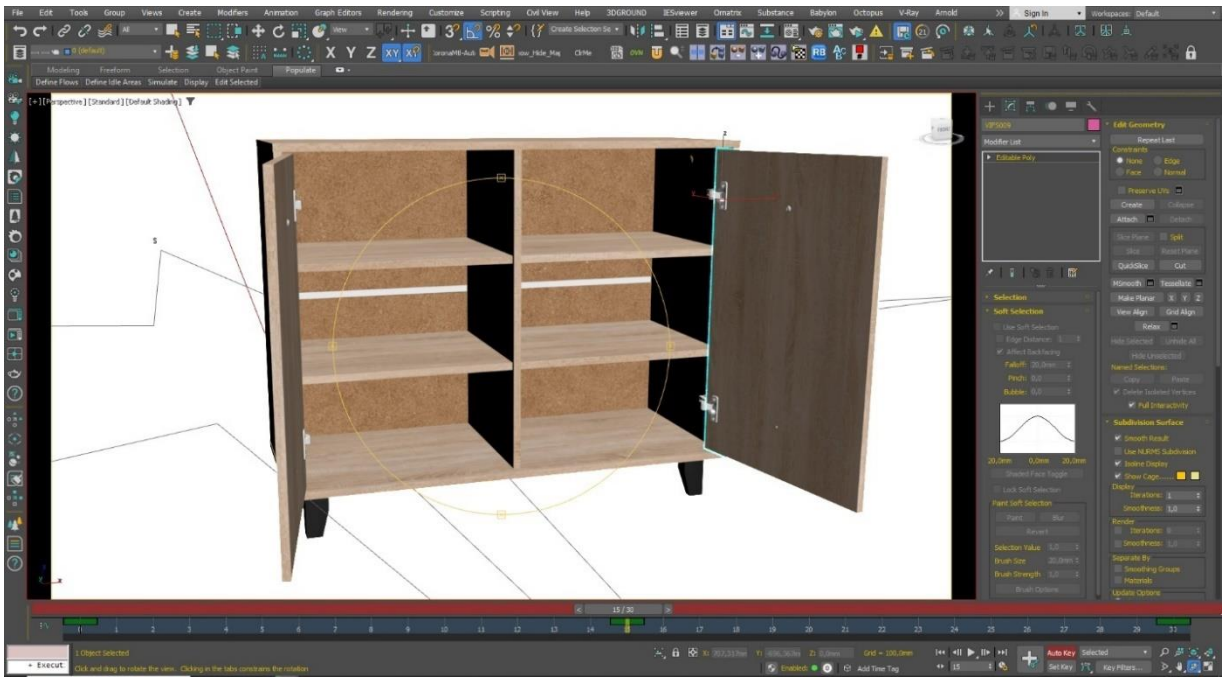

Рисунок 3.13 – Вигляд моделі на 15 кадрі анімації

Останнім етапом є експорт готової моделі у форматі GLTF(GLB) для подальшого використання на сайті (рис. 3.14). Через те, що у оригінальному експорті 3DsMAX не експортуються анімації, було використано плагін Babylon (який також експортує у форматі .babylon, який використовується у бібліотеці Babylon.js, але це не є актуальним для цього проекту).

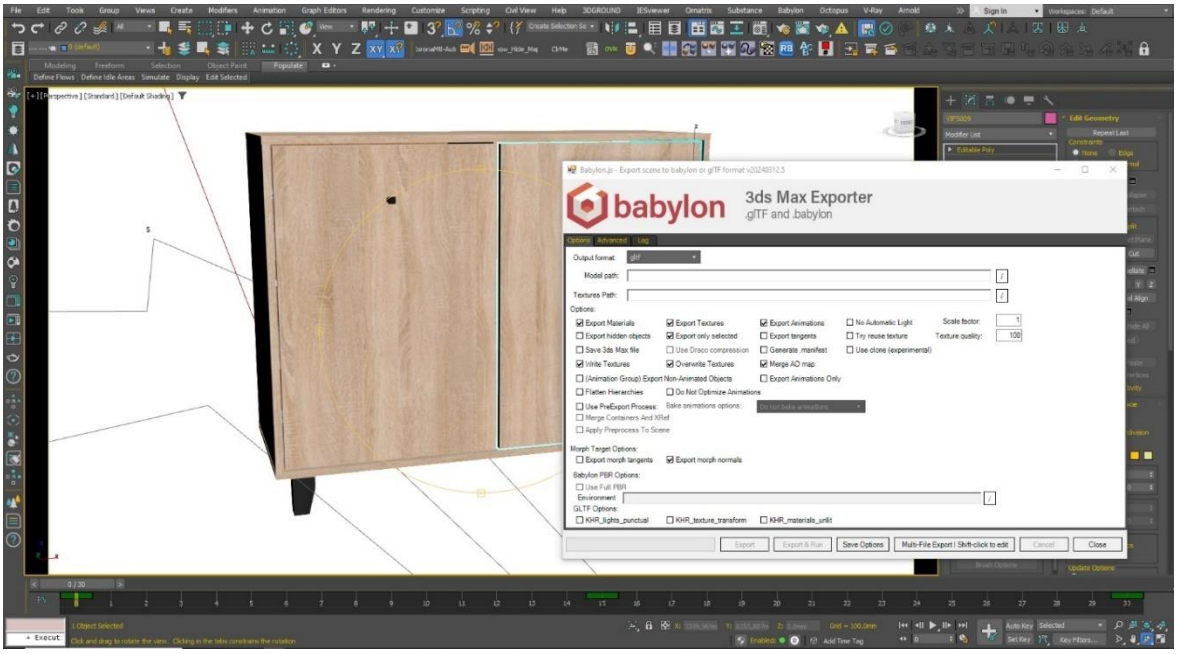

Рисунок 3.14 – Експорт готової моделі у форматі GLTF(GLB)

Модель було експортовано в директорію public у кореневій папці проекту, щоб вона могла бути імпортованою в проект правильно. Після цього, модель можна побачити на сайті (рис. 3.15, 3.16).

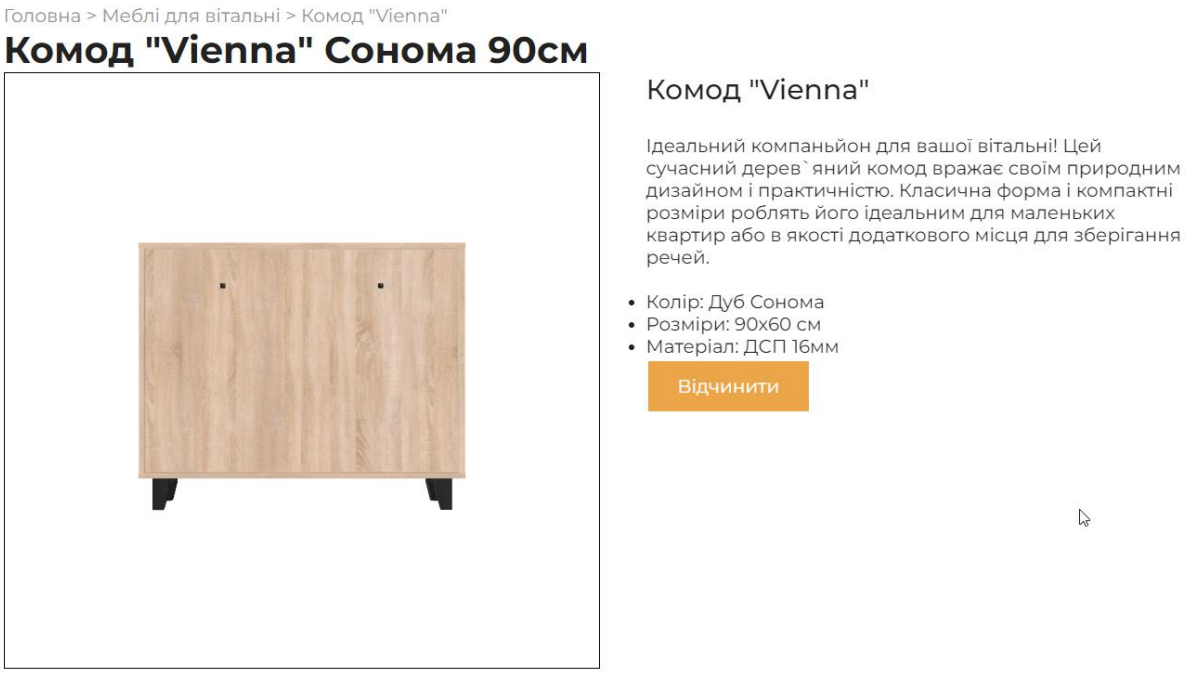

Рисунок 3.15 – Стартовий вигляд моделі при першій ініціалізації

Рисунок 3.16 – Спрацювала анімація відчинення дверей по натисканню кнопки

ВИСНОВКИ

Мета роботи, а саме створення веб-сайту магазину меблів, на якому використовується інтерактивна 3D-графіка для демонстрації товару, була досягнута.

В процесі дослідження були розглянуті та порівняні технології веб-розробки, такі як React.js, Angular, Vue.js, Svelte та Ember.js; технології впровадження та роботи з 3D-графікою у WEB, такі як Three.js та Babylon.js; програмне забезпечення для створення 3D-графіки та анімацій, а саме Autodesk 3DsMAX та Blender. Було обрано React.js, Node.js, Three.js та 3DsMAX, та вибір цих технології було обґрунтовано.

Загалом було створено фронтенд частину сайту магазину меблів, до якої належить головна сторінка, сторінка товарів, корзина, особистий кабінет, сторінка реєстрації та логіну. Були реалізовані усі потрібні модулі, а головне модуль для демонстрації 3D-моделей з анімаціями, та інтерактивної взаємодії з цими моделями. Також, була створена повноцінна 3D-модель комоду, з текстурами та анімаціями, і повністю впроваджена на сайт.

Розробка веб-сайту продемонструвала можливості сучасних веб-технологій для створення інтерактивних 3D-елементів. Використання React.js значно полегшило процес створення динамічного інтерфейсу користувача, що дозволяє легко інтегрувати складні графічні компоненти та забезпечити їхню інтерактивність.

Хоча бекенд частина не була реалізована, обраний стек технологій передбачає можливість майбутнього розширення функціоналу веб-сайту, включаючи інтеграцію з серверними API та базами даних для зберігання та обробки даних користувачів.

В результаті дослідження було доведено, що сучасні веб-технології дозволяють створювати інтерактивні 3D-візуалізації, які можуть бути використані у різних сферах, таких як електронна комерція, освіта, розваги та інші.

ПЕРЕЛІК ПОСИЛАНЬ

1. Jon Duckett. HTML and CSS : Design and Build Websites. Wiley. 2011. 490 с.

2. HTML Living Standard. URL : https://html.spec.whatwg.org/ (дата звернення: 13.03.2024).

3. CSS-Tricks. Animating in 3D with Three.js. URL : https://css-tricks.com/animating-in-3d-with-three-js/ (дата звернення: 13.03.2024).

4. CSS Specification. W3C. URL : https://www.w3.org/Style/CSS/specs.en.html (дата звернення: 13.03.2024).

5. Douglas Crockford. JavaScript: The Good Parts. O'Reilly Media. 2008. 176 с.

6. David Flanagan. JavaScript: The Definitive Guide. O'Reilly Media. 2020. 706 с.

7. Jeremy Keith, Jeffrey Sambells. DOM Scripting: Web Design with JavaScript and the Document Object Model. Apress. 2010. 284 с.

8. Marijn Haverbeke. Eloquent JavaScript: A Modern Introduction to Programming. No Starch Press. 2018. 472 с.

9. Robin Wieruch. The Road to Learn React. Independently Published. 2020. 300 с.

10. Stoyan Stefanov. React Up and Running. O'Reilly Media. 2016. 226 с.

11. Alex Banks, Eve Porcello. Learning React: Functional Web Development with React and Redux. O'Reilly Media. 2017. 350 с.

12. Chris Drackett. Building a React Application with Three.js. LogRocket Blog. URL : https://blog.logrocket.com/building-react-application-three-js/ (дата звернення: 15.03.2024).

13. React.js Official Documentation. URL : https://reactjs.org/docs/getting-started.html (дата звернення: 15.03.2024).

14. Brad Dayley, Brendan Dayley, Caleb Dayley. Learning Angular: A Guide to Building High-Quality Web Apps with Angular and TypeScript. Addison-Wesley Professional. 2017. 640 с.

15. Angular Documentation. URL : https://angular.io/docs (дата звернення: 15.03.2024).

16. Callum Macrae. Vue.js Up & Running. O'Reilly Media. 2018. 174 с.

17. Vue.js Documentation. URL : https://vuejs.org/v2/guide/ (дата звернення: 15.03.2024).

18. Mark Volkmann. Svelte and Sapper in Action. Manning Publications. 2020. 384 с.

19. Svelte Documentation. URL : https://svelte.dev/docs (дата звернення: 15.03.2024).

20. Joachim Haagen Skeie. Ember.js in Action. Manning Publications. 2014. 375 с.

21. Ember.js Guides. URL : https://guides.emberjs.com/release/ (дата звернення: 16.03.2024).

22. Karl Swedberg, Jonathan Chaffer. Learning jQuery: A Hands-On Guide to Building Rich Interactive Web Front Ends. Packt Publishing. 2016. 510 с.

23. Mario Casciaro, Luciano Mammino. Node.js Design Patterns. Packt Publishing. 2020. 520 с.

24. Jos Dirksen. Three.js Cookbook. Packt Publishing. 2015. 364 с.

25. Kenwright. Professional 3D Graphics Using WebGL and Three.js. Independently Published. 2018. 289 с.

26. Paul Lewis. Creating Interactive 3D Graphics with Three.js. Google Developers Blog. URL : https://developers.google.com/web/updates/2013/01/Getting-started-with-Three-js (дата звернення: 19.04.2024).

27. Rachel Smith. Introduction to 3D in the Browser with Three.js. Smashing Magazine. URL : https://www.smashingmagazine.com/2016/01/creating-3d-graphics-in-the-browser-with-three-js/ (дата звернення: 19.04.2024).

28. Three.js Documentation. URL : https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene (дата звернення: 20.04.2024).

29. Jahirul Amin. 3ds Max Projects: A Detailed Guide to Modeling, Texturing, Rigging, Animation and Lighting. Apress. 2015. 460 с.

30. 3ds Max Documentation. Autodesk. URL : https://knowledge.autodesk.com/support/3ds-max/learn (дата звернення: 20.04.2024).

ДОДАТОК А

Посилання на репозиторій проекту на GitHub

https://github.com/Drakenbolt/diplom-proj

ДОДАТОК Б

Код компонента Header.jsx

```
import React from 'react';
import { useNavigate } from 'react-router-dom';
import '../styles/Header.css';


export const Header = () => {
  const navigate = useNavigate();

  const handleMain = () => {
    navigate('/');
  }

  const handleCart = () => {
    navigate('/cart');
  }
  const handleProfile = () => {
    navigate('/profile');
  }

  const handleLogin = () => {
    navigate('/login');
  }

  return (
      <div>
```

```
<div className='headerMenu'>

  <div onClick={handleMain} className='logocontainer' >
    <span className='logoshadow'>ZPComfort</span>
    <span className='logo'>ZPComfort</span>
  </div>

<div className='headerNav'>
  <ul className='nav'>
    <li style={{position: 'relative'}}>
       <a>М'які меблі</a>
       <div className='dropdown'>
         <ul>
           <li><a>Дивани</a></li>
           <li><a>Крісла</a></li>
           <li><a>Крісла-мешки</a></li>
           <li><a>Пуфи</a></li>
         </ul>
       </div>
    </li>
    <li style={{position: 'relative'}}>
      <a>Меблі для спальні</a>
      <div className='dropdown'>
         <ul>
           <li><a>Ліжка</a></li>
           <li><a>Шафи</a></li>
           <li><a>Шафи-купе</a></li>
           <li><a>Тумбочки</a></li>
           <li><a>Нічні столики</a></li>
```

```
            </ul>
          </div>
        </li>
        <li style={{position: 'relative'}}>
          <a>Меблі для вітальні</a>
          <div className='dropdown'>
            <ul>
              <li><a>Столи</a></li>
              <li><a>Стільці</a></li>
              <li><a>Комоди</a></li>
              <li><a>Книжкові шафи</a></li>
            </ul>
          </div>
        </li>
        <li>
          <svg onClick={handleCart} width="25px" height="25px" viewBox="0
0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg">
          <path d="M6.29977 5H21L19 12H7.37671M20 16H8L6 3H3M9 20C9
20.5523 8.55228 21 8 21C7.44772 21 7 20.5523 7 20C7 19.4477 7.44772 19 8
19C8.55228 19 9 19.4477 9 20ZM20 20C20 20.5523 19.5523 21 19 21C18.4477
21 18 20.5523 18 20C18 19.4477 18.4477 19 19 19C19.5523 19 20 19.4477 20
20Z" stroke="#000000" strokeWidth="2" strokeLinecap="round"
strokeLinejoin="round"/>
          </svg>
        </li>
        <li>
          <svg onClick={handleProfile} fill="#000000" width="30px"
height="30px" viewBox="0 0 1024 1024"
xmlns="http://www.w3.org/2000/svg"><path d="M85.333 512C85.333 276.358
276.358 85.333 512 85.333c235.639 0 426.667 191.025 426.667 426.667 0
```

```
235.639-191.027 426.667-426.667 426.667C276.358 938.667 85.333 747.64
85.333 512zM512 128c-212.077 0-384 171.923-384 384 0 142.135 77.222
266.231 192 332.629V628.62c0-77.474 45.885-144.23 111.962-174.575-38.129-
25.737-63.201-69.344-63.201-118.808 0-79.108 64.131-143.238 143.24-
143.238s143.236 64.13 143.236 143.238c0 49.463-25.071 93.071-63.202 118.808
66.078 30.345 111.966 97.101 111.966 174.575v216.009c114.778-66.398 192-
190.494 192-332.629 0-212.077-171.921-384-384-384zm149.333
737.882V628.621c0-82.475-66.859-149.333-149.333-149.333s-149.333 66.859-
149.333 149.333v237.261C408.572 885.278 459.034 896 512 896s103.428-10.722
149.333-30.118zM512 234.667c-55.543 0-100.573 45.027-100.573
100.571S456.457 435.81 512 435.81c55.543 0 100.57-45.028 100.57-
100.572S567.544 234.667 512 234.667z"/></svg>
            </li>
            <li>
              <button onClick={handleLogin} className='loginbutton'
>Вхід</button>
            </li>
           </ul>
           <div className='search-box'>
               <input type="text" placeholder="Пошук товарів..."></input>
               <input type="submit" value="🔍"></input>
           </div>
          </div>
          </div>
       </div>
  )
}

export default Header;
```

ДОДАТОК В

Код компонента Footer.jsx

```
import React from 'react'
import '../styles/Footer.css';

export const Footer = () => {
  return (
    <footer className="footer">
      <div className="container">
        <div className="row">
          <div className="footer-section about">
            <h3 className="logo-text">ZPComfort</h3>
            <p>Ми пропонуємо високоякісні меблі для вашого комфорту.</p>
            <div className="contact">
              <span><i className="fas fa-phone"></i>   123-456-789</span>
              <span><i className="fas fa-envelope"></i>  
info@zpcomfort.com</span>
            </div>
          </div>
          <div className="footer-section links">
            <h3>Швидкі посилання</h3>
            <ul>
              <li><a href="#">Про нас</a></li>
              <li><a href="#">Контакти</a></li>
              <li><a href="#">Блог</a></li>
              <li><a href="#">Каталог</a></li>
            </ul>
```

```
        </div>
        <div className="footer-section social">
          <h3>Соціальні мережі</h3>
          <p>Підписуйтесь на нас у соціальних мережах</p>
          <div className="socials">
          <svg xmlns="http://www.w3.org/2000/svg" fill='white' x="0px" y="0px"
width="50" height="50" viewBox="0 0 30 30">
            <path d="M24,4H6C4.895,4,4,4.895,4,6v18c0,1.105,0.895,2,2,2h10v-9h-
3v-3h3v-1.611C16,9.339,17.486,8,20.021,8 c1.214,0,1.856,0.09,2.16,0.131V11h-
1.729C19.376,11,19,11.568,19,12.718V14h3.154l-0.428,3H19v9h5c1.105,0,2-
0.895,2-2V6 C26,4.895,25.104,4,24,4z"></path>
          </svg>
          <svg xmlns="http://www.w3.org/2000/svg" fill='white' x="0px" y="0px"
width="50" height="50" viewBox="0 0 30 30">
            <path d="M28,6.937c-0.957,0.425-1.985,0.711-3.064,0.84c1.102-
0.66,1.947-1.705,2.345-2.951c-1.03,0.611-2.172,1.055-3.388,1.295 c-0.973-1.037-
2.359-1.685-3.893-1.685c-2.946,0-5.334,2.389-
5.334,5.334c0,0.418,0.048,0.826,0.138,1.215 c-4.433-0.222-8.363-2.346-10.995-
5.574C3.351,6.199,3.088,7.115,3.088,8.094c0,1.85,0.941,3.483,2.372,4.439 c-
0.874-0.028-1.697-0.268-2.416-
0.667c0,0.023,0,0.044,0,0.067c0,2.585,1.838,4.741,4.279,5.23 c-0.447,0.122-
0.919,0.187-1.406,0.187c-0.343,0-0.678-0.034-1.003-
0.095c0.679,2.119,2.649,3.662,4.983,3.705 c-1.825,1.431-4.125,2.284-
6.625,2.284c-0.43,0-0.855-0.025-1.273-
0.075c2.361,1.513,5.164,2.396,8.177,2.396 c9.812,0,15.176-8.128,15.176-
15.177c0-0.231-0.005-0.461-0.015-
0.69C26.38,8.945,27.285,8.006,28,6.937z"></path>
          </svg>
          <svg xmlns="http://www.w3.org/2000/svg" fill='white' x="0px" y="0px"
width="50" height="50" viewBox="0 0 30 30">
```

```
            <path d="M 9.9980469 3 C 6.1390469 3 3 6.1419531 3 10.001953 L 3
20.001953 C 3 23.860953 6.1419531 27 10.001953 27 L 20.001953 27 C
23.860953 27 27 23.858047 27 19.998047 L 27 9.9980469 C 27 6.1390469
23.858047 3 19.998047 3 L 9.9980469 3 z M 22 7 C 22.552 7 23 7.448 23 8 C 23
8.552 22.552 9 22 9 C 21.448 9 21 8.552 21 8 C 21 7.448 21.448 7 22 7 z M 15 9
C 18.309 9 21 11.691 21 15 C 21 18.309 18.309 21 15 21 C 11.691 21 9 18.309 9
15 C 9 11.691 11.691 9 15 9 z M 15 11 A 4 4 0 0 0 11 15 A 4 4 0 0 0 15 19 A 4 4
0 0 0 19 15 A 4 4 0 0 0 15 11 z"></path>
          </svg>
          </div>
        </div>
      </div>
      <div className="footer-bottom">
        &copy; ZPComfort.com | Designed by ZPComfort Team
      </div>
    </div>
  </footer>
 );
}

export default Footer;
```

ДОДАТОК Г

Код компонента ProductList.jsx

```
import React from 'react';
import { useNavigate } from 'react-router-dom';
import '../styles/ProductList.css';

const ProductCard = ({ product }) => {
  const navigate = useNavigate();

  const handleProductClick = () => {
    navigate(product.path);
  };

  return (
    <div className="product-card" onClick={handleProductClick}>
      <img src={product.image} alt={product.name} />
      <h2>{product.name}</h2>
      <p>{product.description}</p>
      <h3>Ціна: {product.price} грн</h3>
      <button>Детальніше</button>
    </div>
  );
};

const ProductList = ({ products }) => {
  return (
    <div className="product-list">
```

```
      {products.map((product) => (
        <ProductCard key={product.id} product={product} />
      ))}
    </div>
  );
};

export default ProductList;
```

ДОДАТОК Д

Код компонента Banner.jsx

```
import React, { useState, useEffect } from 'react';
import '../styles/Banner.css';

function Banner({ images }) {
  const [index, setIndex] = useState(0);

  useEffect(() => {
    const timer = setInterval(() => {
      setIndex((prevIndex) => (prevIndex + 1) % images.length);
    }, 10000);

    return () => {
      clearInterval(timer);
    };
  }, [images]);

  return (
    <div className="banner">
      {images.map((image, i) => (
        <img
          key={image}
          className={`banner-image ${index === i ? 'visible' : ''}`}
          src={image}
          alt="banner"
        />
      ))}
    </div>
  );
}

export default Banner;
```

ДОДАТОК Е

Код компонента FurnitureShopContainer.jsx

```
import React, { useEffect, useRef, useState } from 'react';
import * as THREE from 'three';
import { AnimationMixer } from 'three';
import { GLTFLoader } from 'three/examples/jsm/loaders/GLTFLoader';
import { OrbitControls } from 'three/examples/jsm/controls/OrbitControls';
import '../styles/FurnitureContainerShop.css';
import texturePath from '../assets/texture.png';

const FurnitureModel = ({ modelPath, playAnimation, frameCount,
onAnimationEnd }) => {
  const ref = useRef();
  const mixerRef = useRef(null);
  const actionRef = useRef(null);
  const clock = useRef(new THREE.Clock());
  const [modelLoaded, setModelLoaded] = useState(false);

  useEffect(() => {
    const scene = new THREE.Scene();
    scene.background = new THREE.Color("rgb(255, 255, 255)");

    const camera = new THREE.PerspectiveCamera(25, 640 / 640, 0.1, 1000);
    camera.position.y += 0;
    const renderer = new THREE.WebGLRenderer({ antialias: true });

    renderer.setSize(640, 640);
```

```
renderer.shadowMap.enabled = true;

if (ref.current) {
  while (ref.current.firstChild) {
    ref.current.removeChild(ref.current.firstChild);
  }
  ref.current.appendChild(renderer.domElement);
}

const controls = new OrbitControls(camera, renderer.domElement);
controls.enableDamping = true;
controls.dampingFactor = 0.25;
controls.enableZoom = true;

const loader = new GLTFLoader();
loader.load(
  modelPath,
  (gltf) => {
    gltf.scene.position.set(0, -0.33, 0);
    scene.add(gltf.scene);
    const mixer = new AnimationMixer(gltf.scene);
    mixerRef.current = mixer;
    if (gltf.animations && gltf.animations.length) {
      const action = mixer.clipAction(gltf.animations[0]);
      action.paused = true;
      actionRef.current = action;
      setModelLoaded(true);
    }
  },
  undefined,
  (error) => {
```

```
    console.error(error);

    const geometry = new THREE.BoxGeometry(1, 1, 1);
    const textureLoader = new THREE.TextureLoader();
    const texture = textureLoader.load(texturePath);
    const material = new THREE.MeshBasicMaterial({ map: texture });
    const cube = new THREE.Mesh(geometry, material);
    cube.castShadow = true;
    cube.receiveShadow = true;
    scene.add(cube);
  }
);

const ambientLight = new THREE.AmbientLight(0xffffff, 2);
scene.add(ambientLight);

const light = new THREE.DirectionalLight(0xffffff, 1);
light.position.set(2, 5, 10);
light.castShadow = true;
scene.add(light);

const light2 = new THREE.DirectionalLight(0xffffff, 1);
light2.position.set(-10, -5, -10);
light2.castShadow = true;
scene.add(light2);

camera.position.z = 5;

const animate = function () {
  requestAnimationFrame(animate);
  const delta = clock.current.getDelta();
```

```
      if (mixerRef.current) {
        mixerRef.current.update(delta);
      }
      controls.update();
      renderer.render(scene, camera);
    };

    animate();
  }, [modelPath]);

  useEffect(() => {
    if (playAnimation && actionRef.current && modelLoaded) {
      const framesToPlay = 15;
      const frameDuration = 1 / 30;
      const currentTime = (frameCount * framesToPlay * frameDuration) %
actionRef.current.getClip().duration;

      actionRef.current.paused = false;
      actionRef.current.time = currentTime;
      actionRef.current.play();

      setTimeout(() => {
        actionRef.current.paused = true;
        onAnimationEnd();
      }, framesToPlay * frameDuration * 1000);
    }
  }, [playAnimation, frameCount, modelLoaded, onAnimationEnd]);

  return <div ref={ref} className='furniture-model' />;
};
```

```
const ProductDescription = ({ title, description, features, onPlayAnimation,
isButtonDisabled, buttonText }) => (
  <div className='product-description'>
    <h2>{title}</h2>
    <p>{description}</p>
    <br />
    <ul>
      {features.map((feature, index) => (
        <li key={index}>{feature}</li>
      ))}
    </ul>
    <button onClick={onPlayAnimation}
disabled={isButtonDisabled}>{buttonText}</button>
  </div>
);

export const FurnitureShopContainer = ({ product }) => {
  const [frameCount, setFrameCount] = useState(-1);
  const [isButtonDisabled, setIsButtonDisabled] = useState(false);
  const [buttonText, setButtonText] = useState('Відчинити');
  const [playAnimation, setPlayAnimation] = useState(false);

  useEffect(() => {
    setFrameCount(-1);
  }, [product.modelPath]);

  const handlePlayAnimation = () => {
    setPlayAnimation(true);
    setIsButtonDisabled(true);
    setFrameCount((prevCount) => prevCount + 1);
    setButtonText(buttonText === 'Відчинити' ? 'Зачинити' : 'Відчинити');
```

```
  };
  const handleAnimationEnd = () => {
    setIsButtonDisabled(false);
    setPlayAnimation(false);
  };

  return (
    <div className='furniture-shop-container'>
      <FurnitureModel
        modelPath={product.modelPath}
        playAnimation={playAnimation}
        frameCount={frameCount}
        onAnimationEnd={handleAnimationEnd}
      />
      <ProductDescription
        title={product.title}
        description={product.description}
        features={product.features}
        onPlayAnimation={handlePlayAnimation}
        isButtonDisabled={isButtonDisabled}
        buttonText={buttonText}
      />
    </div>
  );
};

export default FurnitureShopContainer;
```