МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІЦІЙНА РОБОТА

другий (магістерський) рівень вищої освіти

на тему:

« Обґрунтування ефективних інструментів розробки користувацьких інтерфейсів інтернет-магазину »

Виконав: студент гр. Іт-61
спеціальності 126 «Інформаційні системи та технологій»

Степаненко В.В.
(прізвище та ініціали)

Керівник: Желєзняк А.М.
(прізвище та ініціали)

ДУБЛЯНИ 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Рівень вищої освіти другий (магістерський)
Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____
(підпис)

д.т.н., професор, Тригуба А. М.
(вч. звання, прізвище, ініціали)

“____” _____2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Степаненко Валерій Вікторович
(прізвище, ім’я, по батькові)

1. Тема роботи *«Обґрунтування ефективних інструментів розробки користувацьких інтерфейсів інтернет-магазину»*
керівник роботи *к. е н., доцент., Желєзняк А.М.*
(наук.ступінь, вч. звання, прізвище, ініціали)

затверджені наказом Львівського НАУ № 133 / к - с від 28.04.2023 р
2. Строк подання студентом роботи 15.01.2024 р.
3. Вихідні дані: *вихідні дані та вимоги до користувацьких інтерфейсів, опис інструментів та технологій для реалізації компонентів користувацького інтерфейсу, характеристика об’єкту дослідження, опис бібліотек мов програмування, науково-технічна і довідкова література.*
4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)
Вступ
1. Аналіз стану питання в теорії та практиці та постановка завдання
2. Обґрунтування, вибір та реалізація інструментів вирішення задачі .
3. Результати вирішення задачі реалізації користувацького інтерфейсу .
4. Охорона праці та безпека в надзвичайних ситуаціях .
5. Визначення ефективності .
Висновки
Бібліографічний список
5. Перелік графічного матеріалу
Графічний матеріал подається у вигляді презентації

6. Консультанти розділів

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата		Відмітка про виконання
		завдання видав	завдання прийняв	
1, 2, 3, 5				
4				

7. Дата видачі завдання 28.04.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Відмітка про виконання
1	*Отримання завдання. Вивчення рекомендованої літератури по темі роботи. Написання першого розділу .*	*10.04.2023 - 31.05.2023*	
2	*Проектування та опис технічного завдання, вибір та обґрунтування мови програмування (написання другого розділу).*	*1.06.2023 - 31.08.2023*	
3	*Вибір та обґрунтування інструментів реалізації , програмна реалізація поставленого завдання (написання третього розділу)*	*1.09.2023 - 31.10.2023*	
4	*Написання розділу «Охорона праці та безпека у надзвичайних ситуаціях»*	*1.11.2023 - 20.11.2023*	
5	*Оцінка ефективності поставленого завдання (виконання п'ятого розділу)*	*21.11.2023- 15.12.2023*	
6	*Завершення оформлення основної частини, написання висновків та підготовка презентаційного матеріалу*	*16.12.2023 - 31.12.2023*	
7	*Завершення роботи в цілому. Підготовка до захисту кваліфікаційної роботи*	*1.01.2024- 15.01.2024*	

Здобувач _____ Степаненко В.В.
(підпис)

Керівник роботи _____ Желєзняк А.М.
(підпис)

УДК 004.758.5:339.371.4

Кваліфікаційна робота: 75 сторінок текстової частини, 13 таблиць, 14 рисунків, 32 джерела літератури, 1 додаток.

«Обґрунтування ефективних інструментів розробки користувацьких інтерфейсів інтернет-магазину» Степаненко Валерій Вікторович – Кваліфікаційна робота. Кафедра інформаційних технологій. Дубляни, Львівський національний університет природокористування, 2024 р.

Розглянуто користувацькі інтерфейси на прикладі інтернет-магазину з продажу велосипедів. Проаналізовано предметну область, існуючі аналоги та визначено функціональні вимоги до сайту. Сформовано список сторінок, компонентів та технологій для їх реалізації.

Використано визначені інструменти, визначені їх переваги та недоліки. Обґрунтована потреба в застосуванні інших бібліотек.

Здійснено аналіз травматичних ситуацій при виконанні різних робіт у сфері використанням комп’ютерної техніки, викладено питання охорони праці.

Ключові слова: користувацькі інтерфейси, інтернет-магазин, дизайн.

ЗМІСТ

ВСТУП

В сучасному світі, де електронна торгівля стала невід'ємною частиною бізнесу, розробка інтерфейсу інтернет-магазину набула великої актуальності. За останні десятиліття спостерігається стрімкий розвиток онлайн-торгівлі, що зумовлено зростанням попиту на зручний та ефективний користувацький інтерфейс. З підвищенням конкуренції в інтернеті, створення привабливого та інтуїтивно зрозумілого інтерфейсу стає ключовим для залучення та утримання клієнтів. Сучасні інтернет-магазини повинні пропонувати не лише якісні товари, а й забезпечувати зручність та задоволення від покупки, що визначає актуальність дослідження в області розробки користувацьких інтерфейсів.

Наукова новизна дослідження полягає у визначенні інструментів, які поєднують в собі сучасні тенденції розробки інтерфейсів для інтернет-магазинів, з фокусом на мобільній адаптації, персоналізації, швидкості завантаження та використанні AI. Також, дослідження розглядає попередній розвиток інтерфейсів, включаючи перехід від простих HTML, CSS та JavaScript до більш складних та інтуїтивних інтерфейсів, що відображає еволюцію сучасного електронного бізнесу.

Поняття "користувацький інтерфейс" з'явилося в реакції на зростаючу складність та різноманітність веб-сторінок та додатків. Коли розроблялися перші веб-ресурси, вони складалися з простого HTML-коду, що відображав інформацію, але не надавав користувачам інтерактивності. Завдяки розвитку технологій, виникла необхідність в більш складних інтерфейсах, які забезпечували зручну навігацію та інтеграцію користувачів з веб-сайтами. Pulli Petri та Antoniac Peter надають наступне визначення поняттю "користувацький інтерфейс": це система елементів, які дозволяють користувачам ефективно взаємодіяти з веб-ресурсами, включаючи кнопки, меню, форми, анімацію та інші компоненти [1].

Мета кваліфікаційної роботи полягає у визначенні та обґрунтуванні ефективних інструментів для розробки користувацьких інтерфейсів інтернет-магазину.

В ході виконання кваліфікаційної роботи було визначено наступні завдання:

1. Проаналізувати існуючі підходи, дослідити та оглянути популярні інтернет-магазини “Allo” та “Rozetka”.
2. Обґрунтувати основні користувацькі інтерфейси та компоненти для реалізації.
3. Визначити інструменти для реалізації користувацьких інтерфейсів.
4. Реалізувати компоненти користувацьких інтерфейсів.
5. Оцінити ефективність інструментів розробки користувацьких інтерфейсів.

Загалом кваліфікаційна робота складається з п’яти розділів. У першому досліджено теоретичні основи користувацьких інтерфейсів та їх використання в інтернет магазинах, вивчено їх використання на прикладі інтернет-магазинів Allo та Rozetka. Другий розділ містить обґрунтування вибору мови програмування та здійснено моделювання інтернет-магазину велосипедів. Третій розділ присвячено аналізу інструментів реалізації компонентів, їх практичного застосування для реалізації компонентів користувацького інтерфейсу, а також оцінювання результатів. Четвертий розділ містить інформацію щодо охорони праці. В п’ятому розділі визначено влив підбору інструментів розробки користувацьких інтерфейсів на ефективність.

РОЗДІЛ 1
АНАЛІЗ СТАНУ ПИТАННЯ В ТЕОРІЇ ТА ПРАКТИЦІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Теоретичні основи користувацьких інтерфейсів

Основним призначенням користувацького інтерфейсу інтернет-магазину є надання користувачам зручної та ефективної можливості взаємодії з вмістом та функціоналом веб-ресурсу. Це включає в себе:

- Навігацію по сторінці та взаємодію з різними елементами інтерфейсу;
- Пошук і перегляд інформації;
- Виконання операцій та дій, таких як введення даних, вибір товарів, відправлення замовлень тощо;
- Забезпечення приємного та продуктивного користувацького досвіду.

Методи і засоби взаємодії постійно змінюються. Куклінова Тетяна Вікторівна визначає наступні сучасні тенденції інтернет-торгівлі в Україні, які подані в таблиці 1.1. [2].

Таблиця 1.1. - Сучасні тенденції інтернет-торгівлі в Україні

Назва тенденції	Опис
1	2
Мобільна адаптація	Зростаюча кількість користувачів використовує мобільні пристрої для покупок. Інтерфейс магазину повинен бути адаптованим для різних розмірів екранів і працювати ефективно на смартфонах та планшетах
Персоналізація	Важливо розробляти інтерфейси, які дозволяють користувачам налаштовувати свій досвід покупок, враховуючи їхні індивідуальні потреби та вподобання.

Продовження табл.1.1.

1	2
Швидкість завантаження	Повільність завантаження сайту може призвести до втрати клієнтів. Тому важливо вдосконалювати технічну частину інтерфейсу для підвищення швидкості роботи
Використання технологій штучного інтелекту (AI)	AI може використовуватися для підвищення персоналізації, аналізу поведінки користувачів та підвищення зручності користування інтерфейсом

У таблиці 1.2. подано причини важливості розробки ефективних інструментів для користувацьких інтерфейсів інтернет-магазинів.

Таблиця 1.2. - Причини важливості розробки ефективних інструментів для користувацьких інтерфейсів

Назва причини	Опис
Покращення конкурентоспроможності	Здатність пропонувати привабливий та зручний інтерфейс допомагає вирізнятися серед конкурентів і залучати більше клієнтів
Підвищення задоволення клієнтів	Задоволені клієнти більш схильні повертатися та рекомендувати інтернет-магазин іншим. Гарний інтерфейс сприяє позитивному досвіду покупки
Покращення продажів	Інтуїтивний та зручний інтерфейс збільшує конверсію, оскільки споживачі більш схильні завершити покупку, коли процес є простим та зрозумілим

При розробці користувацьких інтерфейсів потрібно також працювати з UX (взаємодія користувача). UI (інтерфейс користувача) та UX (взаємодія користувача) є двома важливими аспектами веб-дизайну та розробки програмного забезпечення, і вони часто використовуються разом. Однак ці терміни мають різні значення та орієнтації, які детально розписані в таблиці 1.3.

Таблиця 1.3. – Порівняльна характеристика UI та UX дизайну

	Означення	Складові	Задачі
UI	UI визначає зовнішній вигляд продукту та як користувач може з ним взаємодіяти	UI включає в себе елементи дизайну, такі як кольори, шрифти, графіка, ікони та інші візуальні елементи, що створюють зовнішній вигляд продукту	Головна мета UI - забезпечити користувачам ефективний та естетичний спосіб взаємодії з продуктом. Дизайн UI має бути привабливим та легким для розуміння
UX	UX охоплює всі аспекти взаємодії користувача з продуктом, включаючи відчуття, які він отримує від використання продукту	UX включає в себе вивчення користувачів, створення прототипів, тестування користувацької дії та забезпечення, щоб весь процес взаємодії був зручним та задовільним для користувача	Головна мета UX - забезпечити користувачам приємний та ефективний досвід використання продукту. Це включає в себе розуміння потреб користувачів, врахування їхніх очікувань та спрощення взаємодії з продуктом

Отже, хоча UI та UX взаємопов’язані, вони відрізняються в тому, що UI фокусується на зовнішньому вигляді продукту, тоді як UX зосереджується на тому, як користувачі взаємодіють з продуктом, над процесом та кроками, необхідними для досягнення певної задачі та яким чином цей процес можна зробити максимально комфортним та ефективним.

Науковці також визначають значну різницю між користувацькими інтерфейсами та компонентами в коді. UI відповідає за те, як користувачі взаємодіють з веб-сайтом або веб-додатком, а компоненти відповідають за те, як веб-сайт або веб-додаток функціонує. У таблиці 1.4. наведено ключові відмінності.

Таблиця 1.4. - UI елементи та компоненти в коді

	UI елементи	Компоненти в коді
Ціль	створити зручний і інтуїтивно зрозумілий спосіб для користувачів взаємодіяти з веб-сайтом або веб-додатком	забезпечити функціональність веб-сайту або веб-додатка
Формат	зазвичай складаються з візуальних елементів, таких як кнопки, меню та форми	Компоненти можуть бути візуальними, а також функціональними. Наприклад, компонент може бути просто елементом HTML, або він може включати в себе JavaScript для виконання певного завдання.
Розробка	UI розробляється дизайнерами та UX-фахівцями	Компоненти розробляються розробниками
Взаємодія	Користувач взаємодіє з UI	Компоненти можуть взаємодіяти з користувачами, але вони також можуть взаємодіяти один з одним
Приклади	Прикладом UI є поле введення номера телефону.	Для реалізації номеру телефону можна використати безліч компонентів, наприклад цифровий інпут, візуальна цифрова клавіатура у якій кожна цифра - це кнопка та безліч інших

Проаналізувавши основні характеристики UI, UX та компонентів, випливає висновок, що ці поняття тісно пов’язані і не можуть існувати окремо. Створення UI/UX частин дизайну неможливе без узгодження з розробником.

Тому важливо знайти ефективні інструменти для максимального покриття UI/UX потреб.

Користувацькі інтерфейси веб-сторінок використовуються в широкому спектрі веб-додатків та сервісів. Це охоплює такі області, як:

- Інтернет-магазини. Вони надають можливість користувачам вибирати та купувати товари онлайн;
- Соціальні мережі. Де користувачі спілкуються та діляться контентом;
- Новинні портали. Де відвідувачі читають новини та статті;
- Банківські системи. Де клієнти перевіряють стан рахунків та здійснюють фінансові операції;
- Освітні платформи. Де студенти навчаються та виконують завдання.

Таблиця 1.5. - Тенденції розвитку користувацьких інтерфейсів

Назва	Опис
Мобільна адаптація	Зростання використання смартфонів для доступу до веб-сайтів вимагає мобільної адаптації. Веб-сайти повинні оптимізуватися для різних розмірів екранів і пристосовуватися до вимог мобільних користувачів
Мінімалізм	Сучасні інтерфейси відзначаються простотою та мінімалістичним дизайном. Це сприяє зосередженню на ключовому змісті та полегшує навігацію, що загалом покращую UX - user experience
Персоналізація	Сучасні користувацькі інтерфейси надають користувачам можливість налаштувати інтерфейс під свої потреби. Це може включати вибір тем, налаштування віджетів та інше
Інтерактивність	Сучасні інтерфейси повинні бути інтерактивними та відповідати на дії користувачів без зайвого перезавантаження сторінки

Сучасні користувацькі інтерфейси веб-сторінок відзначаються численними тенденціями, які визначають їхній розвиток та споживчі очікування. У таблиці 1.5. наведено основні тенденції.

Актуальність стану питання розробки користувацьких інтерфейсів веб-сторінок визначається високим попитом на якісні та ефективні інтерфейси. З підвищенням конкуренції в інтернеті, створення привабливого та зручного користувацького інтерфейсу стає важливим для залучення та утримання клієнтів. Сучасні інтернет-магазини та веб-сайти повинні не лише представляти товари та інформацію, але і забезпечувати задоволення та комфорт під час користування. Тому дослідження в галузі розробки користувацьких інтерфейсів веб-сторінок є актуальним та має велике значення для сучасної науки та практики.

У цьому розділі розглянуто теоретичні аспекти користувацьких інтерфейсів веб-сторінок, включаючи їх призначення, інструментарій розробки та сучасні тенденції. Аналіз показує, що розробка ефективних користувацьких інтерфейсів є важливою та актуальною задачею, особливо в контексті зростання популярності інтернет-магазинів та веб-сервісів. У подальших розділах дослідження будуть розглянуті конкретні підходи та рекомендації для розробки користувацьких інтерфейсів, а також результати експериментального дослідження їх ефективності.

1.2. Опис предметної області

Предметна область передбачає дослідження процесу розробки інтерфейсів інтернет-магазинів та намагається знайти оптимальні інструменти для забезпечення їх ефективності. Ця тема актуальна в контексті розвитку електронної комерції та постійно зростаючого попиту на інтернет-магазини як засіб для ефективної взаємодії з клієнтами.

Інтернет-магазин (також відомий як онлайн-магазин, електронний магазин або е-комерція) - це веб-сайт або платформа в Інтернеті, де покупці можуть переглядати, вибирати та купувати товари та послуги через Інтернет. Це віртуальний аналог традиційного магазину, де клієнти можуть придбати товари, але зручніше та доступніше зробити це онлайн.

Інтернет-магазини дозволяють покупцям шукати товари за допомогою пошуку, переглядати описи товарів, порівнювати ціни, дивитися фотографії товарів, додавати товари до кошика покупок, виконувати замовлення та здійснювати оплату. Вони підтримують різні методи оплати, включаючи кредитні картки, платіжні системи, банківські перекази тощо.

Інтернет-магазини допомагають підприємцям та компаніям розширити свою аудиторію та ринок, оскільки вони можуть обслуговувати клієнтів з усього світу. Також це спрощує процес продажу і покупки товарів, що дозволяє клієнтам здійснювати покупки в зручний для них час.

Інтернет-магазини можуть пропонувати різні види товарів та послуг, від одягу і електроніки до продуктів харчування та послуг доставки. Також вони можуть включати різні функції, такі як відгуки клієнтів, системи знижок та програми лояльності, що роблять покупки онлайн більш привабливими для споживачів.

Інтернет-магазини та служби доставки, такі як “Нова Пошта” та “Укрпошта”, дозволяють споживачам здійснювати покупки в онлайн-режимі, що значно спрощує процес купівлі товарів. Перше, що варто відзначити, це

безперервна доступність цих сервісів - вони працюють цілодобово, що дає можливість клієнтам здійснювати покупки в будь-який зручний для них час.

Додатково, завдяки інтернет-магазинам, клієнти можуть вибирати товари зі зручності свого дому чи офісу, уникнувши необхідності відвідування фізичних магазинів. Широкий асортимент товарів, представлений в онлайн-каталогах, надає можливість знайти необхідну продукцію зі специфічними характеристиками.

Безпосередньо по завершенні вибору товарів, клієнти можуть скористатися послугами доставки для отримання свого замовлення. “Нова Пошта” та “Укрпошта” надають широкий спектр послуг, включаючи швидку доставку, можливість відстеження посилок та надійну упаковку. Це дозволяє споживачам мати впевненість у тому, що їх товар буде доставлений надійно та вчасно.

Історія розвитку веб-користувацьких інтерфейсів (веб-інтерфейсів) свідчить про надзвичайно швидкий та складний процес змін у способах, якими користувачі взаємодіють із веб-сайтами та веб-додатками. Згідно з Гончар В. М. та Римар П. В. виділено п’ять основних етапів за часом: початок інтернету (1990 - 2000), зростання складності (2000 - 2010), епоха соціальних мереж і мобільних пристроїв (2010 - 2015), розширення можливостей (2015 - до сьогодні) та майбутнє (після 2021). Опис кожного етапу та ключові моменти подані в таблиці 1.6 [3].

Розвиток веб-користувацьких інтерфейсів відображає не лише технологічні зміни, але й зростаючі очікування користувачів щодо зручності та ефективності взаємодії з веб-середовищем. А також він підсилений боротьбою за клієнта, внутрішньою конкуренцією. Розробники постійно шукають нові способи покращити цю взаємодію, забезпечити безпеку та забезпечити доступність для різних користувачів.

Таблиця 1.6. – Основні етапи історії розвитку технологій для створення веб-сторінок

Назва етапу	Опис етапу
Початок інтернету (1990-2000)	Перші веб-сайти використовували текстові інтерфейси та гіпертекстові посилання. Графічні інтерфейси з'явилися з винайденням браузера Mosaic і пізніше Netscape Navigator. HTML (HyperText Markup Language) був основним мовним інструментом для створення веб-сторінок
Зростання складності (2000-2010)	Веб-сайти стали більш складними з використанням CSS (Cascading Style Sheets) для кращого дизайну. Вперше використовувалися JavaScript для створення динамічних інтерфейсів. AJAX (Asynchronous JavaScript and XML) дозволив веб-сторінкам асинхронно взаємодіяти з сервером без перезавантаження сторінки
Епоха соціальних мереж і мобільних пристроїв (2010-2015)	Соціальні мережі вели до зростання інтерактивності та ролі зображень та відео на веб-сайтах. Мобільні пристрої вимагали адаптивних дизайнів та веб-додатків. Single-Page Applications (SPA) стали популярними завдяки фреймворкам, таким як Angular, React і Vue
Розширення можливостей (2015-до сьогодні)	Продовжується розвиток SPA та багато інших веб-технологій. Підвищена увага приділяється швидкості завантаження та продуктивності веб-сайтів. Глибока інтеграція веб-додатків із штучним інтелектом (AI) та Інтернетом речей (IoT).
Майбутнє (після 2021)	Розвиток веб-інтерфейсів включає в себе більше голосового та жестового керування, а також віртуальну реальність (VR) і доповнену реальність (AR). Зростає популярність Progressive Web Apps (PWA), які об'єднують можливості веб-додатків та веб-сайтів

Веб-сторінки мають різні інтерактивні елементи, які називаються UI (User Interface) компонентами. Основні UI компоненти веб-сторінки подані в таблиці 1.7.

Таблиця 1.7. - UI компоненти веб-сторінки

Назва компоненту	Опис UI компоненту
1	2
Меню навігації (Navigation Menu)	Меню навігації містить посилання на різні сторінки або розділи веб-сайту. Воно допомагає користувачам швидко переходити між різними частинами сайту
Панель пошуку (Search Bar)	Цей компонент дозволяє користувачам шукати конкретний контент або інформацію на веб-сайті
Шапка (Header)	Шапка розташовується у верхній частині сторінки і зазвичай містить логотип, заголовок сайту та інші важливі елементи, такі як контактна інформація або посилання на соціальні мережі
Підвал (Footer)	Підвал сторінки розташовується у нижній частині і може містити додаткову інформацію про сайт, посилання на сторінки з контактами, правила конфіденційності тощо
Кнопки (Buttons)	Вони застосовуються для обролення певних подій, таких як надсилання форми, відправлення коментарів або переходу на іншу сторінку
Поля вводу (Input Fields)	Поля вводу дозволяють користувачам вводити текстову інформацію, таку як ім’я, адресу електронної пошти або пароль
Списки (Lists)	Списки використовують для відображення набору даних або елементів впорядкованого списку
Зображення (Images)	Зображення додають візуальний контент на сторінку і можуть використовуватися для ілюстрації тексту або привертання уваги користувачів

Продовження табл.1.7.

1	2
Текстовий контент (Text Content)	Текст веб-сторінки поділений на заголовки, абзаци, списки та інші текстові елементи
Форми (Forms)	Форми дозволяють користувачам надсилати інформацію на сервер, таку як реєстраційні дані, коментарі або результати пошуку
Посилання (Links)	Посилання дозволяють користувачам переходити між різними сторінками сайту або виходити на інші ресурси в Інтернеті
Слайдери (Sliders)	Ці компоненти дозволяють користувачам вибирати значення з певного діапазону, наприклад, налаштовувати гучність аудіо або регулювати яскравість зображення
Карти (Maps)	Використовуються для відображення географічних мап або місць розташування
Відеопрогравачі (Video Players)	Дозволяють відтворювати відео на сторінці
Анімація (Animation)	Анімація може використовуватися для покращення візуального враження сторінки

І це лише базові UI компоненти, і їх використання може варіюватися в залежності від конкретного дизайну веб-сторінки та її функціональності.

1.3. Огляд і аналіз існуючих аналогів

Одними з найбільших та найпопулярніших інтернет-магазинів в Україні є “Rozetka” та “Allo”. Вони спеціалізуються на продажу електроніки, побутової техніки, комп'ютерів, мобільних пристроїв та інших товарів. Мають велику мережу власних магазинів, які дозволяють оглянути і забрати товар особисто. Підтримують більшість видів оплати та доставки. Rozetka заснована у 2008 році, Allo на ринку з 2002 року. Вони набули великої популярності завдяки інтернет-магазинам. Тому доцільно розглянути користувацькі інтерфейси на їхньому прикладі. Адже вони вже зробили багато редизайнів на основі зібраної статистики використання функціоналу а також відгуків користувачів.

На рисунках 1.1. та 1.2. зображено головні сторінки інтернет-магазинів Allo та Rozetka відповідно. На кожній сторінці зверху бачимо компонент Header. Він містить логотип компанії, який одночасно є посиланням на головну сторінку, кнопку “Каталог”, яка дозволяє переглянути каталог товарів, поле пошуку, особистий кабінет, обрані товари та кошик.

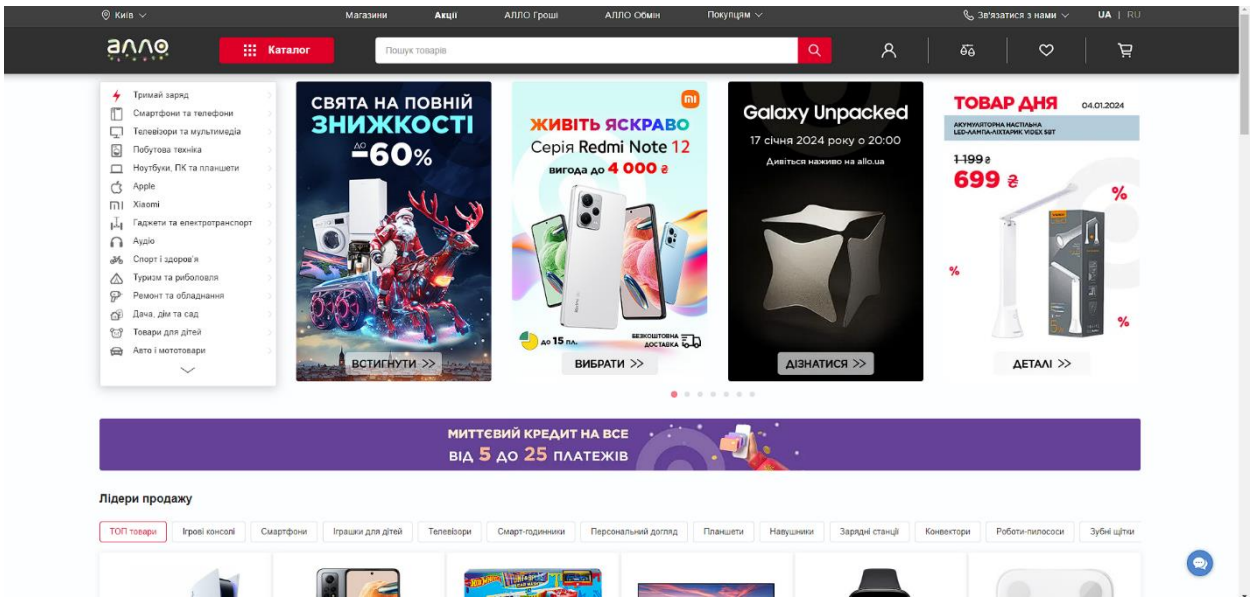


Рисунок 1.1. - Головна сторінка інтернет-магазину Allo

Header головної сторінки інтернет-магазину Allo також має кнопку “порівняння”. Цей функціонал дозволяю користувачам додати певні товари до порівняння і у швидкому доступі порівняти їхні характеристики.

У верхній частині з меншим шрифтом розташована додаткова інформація, зокрема вибір геолокації, список магазинів, Алло гроші, Алло обмін, вибір мови, зв’язок і додаткова інформація покупцям.

Основний вміст сторінки можна поділити на дві частини. В лівій частині знаходиться каталог товарів. В праві знаходиться слайдер з банерами, що містять інформацію про актуальні акції. Знизу знаходиться список товарів з категорії “Лідери продажу”.

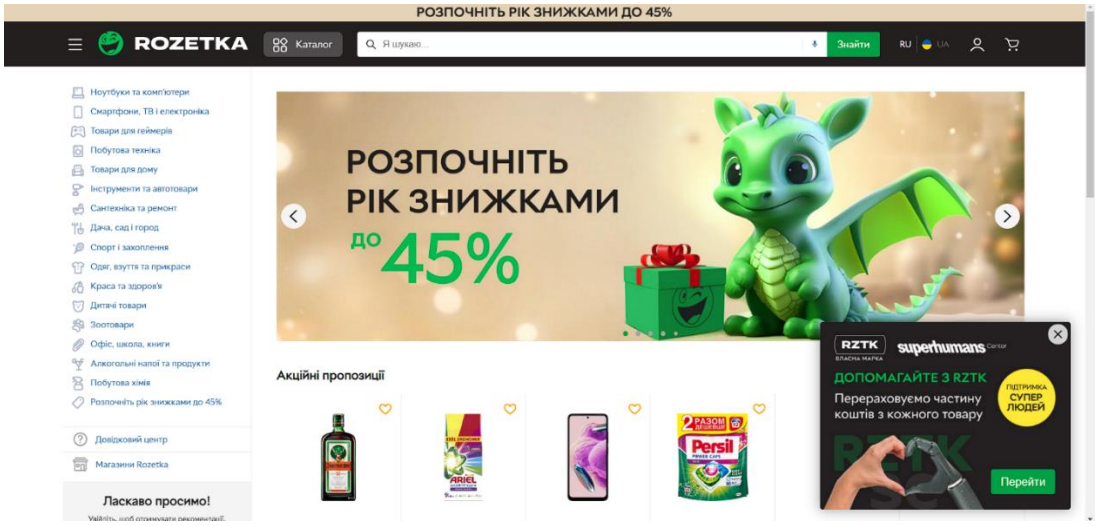


Рисунок 1.2. - Головна сторінка інтернет-магазину Rozetka

Header головної сторінки інтернет-магазину Rozetka містить менше інформації. Більша частина додаткової інформації захована у кнопці меню, або як її ще називають - гамбургер. Зверху знаходиться полоска з текстом “Розпочніть рік знижками до 45%”, яка є посиланням на акційні товари.

Основний вміст сторінки можна поділити на дві частини. В лівій частині знаходиться каталог товарів. В праві знаходиться банер, що містить інформація про поточну акцію, а також список акційних товарів.

На рисунку 1.3. зображена нижня частина (або Footer) головної сторінки інтернет-магазину Allo. Вона містить 5 списків, а саме: АЛЛО, Інформація, Вигода, Комфорт та Бізнес. Ще нижче доступні посилання на сертифікати безпеки та офіційних представників.

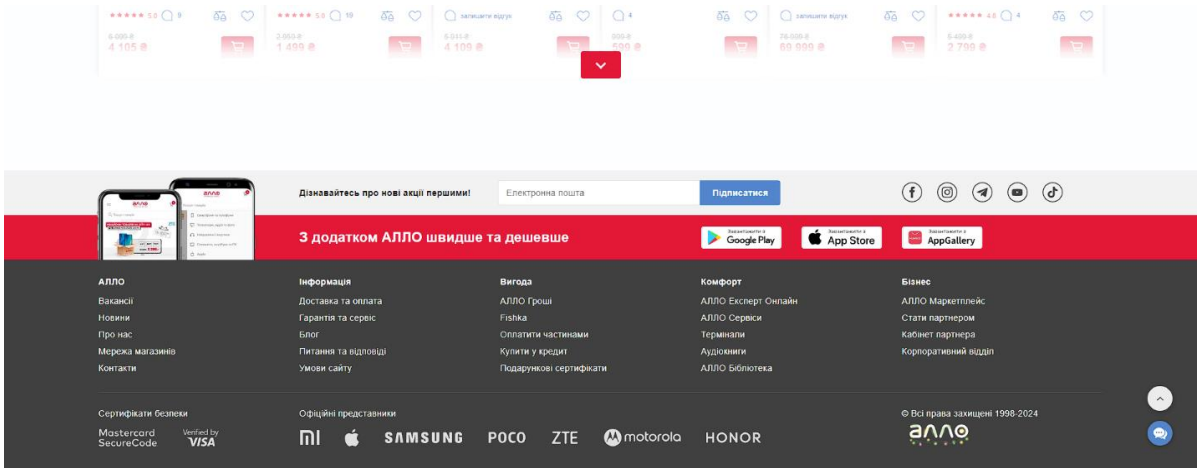
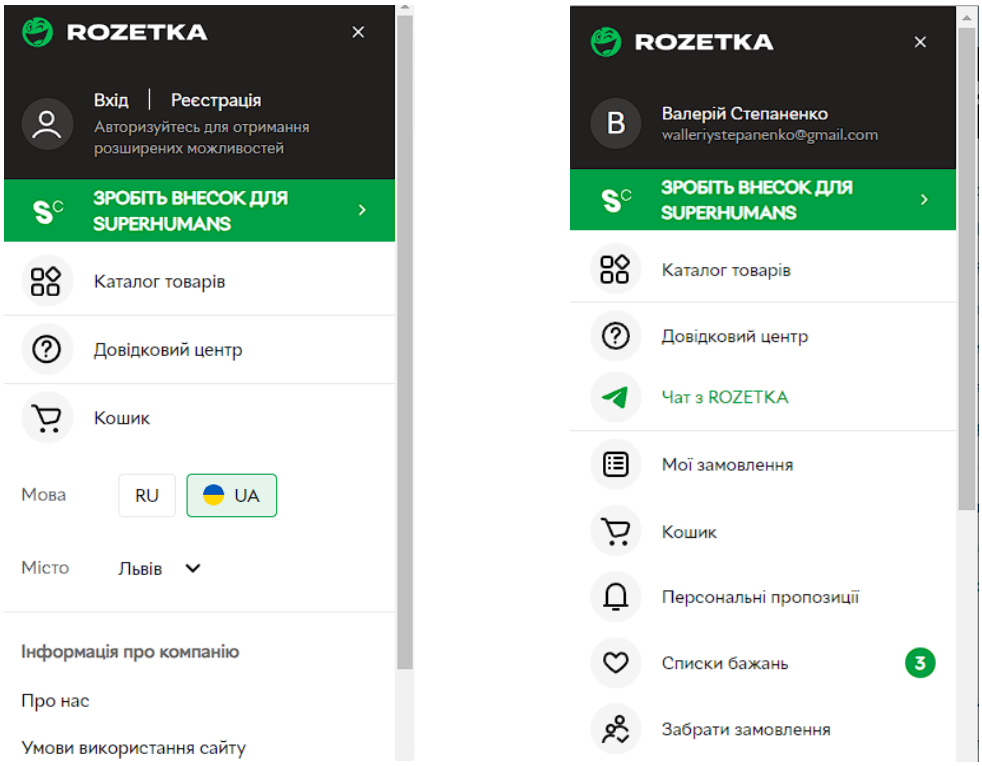


Рисунок 1.3. - Нижня частина головної сторінки інтернет-магазину Allo

Інтернет-магазин Rozetka не має Footer. Натомість ця інформація перенесена у бокову панель або Sidebar, який зображений на рисунку 1.4. Зверху дублюється особистий кабінет, трошки нижче знаходиться вибір мови, геолокації і нижче знаходиться розділи “Інформація про компанію”, “Допомога”, “Сервіси”, “Партнерам”.



а) б)

Рисунок 1.4. - Користувацька частина сторінки інтернет-магазину Rozetka

а) для незареєстрованих користувачів

б) для зареєстрованих користувачів

На рисунку 1.4. б) зображена користувацька частина сторінки інтернет-магазину Rozetka для зареєстрованих користувачів. Перший елемент списку відображає ім’я користувача та його email. Фактично цей блок є посиланням на сторінку користувача. Також появились такі елементи, як “Мої замолвення”, “Кошик”, “Персональні пропозиції” та “Списки бажань”.

Кожен інтернет-магазин містить асортимент товарів, який доступний у відповідній сторінці. Для порівняння зайдемо у каталог зарядних станцій EcoFlow. На рисунку 1.5. та 1.6. зображені каталоги товарів інтернет-магазинів Allo та Rozetka відповідно. Зверху знаходяться breadcrumbs, які полегшують навігацію. Вміст сторінки можна поділити на дві частини: зліва панель фільтрів, справа список товарів. Всі фільтри є collapsible, більшість опцій реалізована з використанням checkbox. Певні фільтри містять input для пошуку певного значення. Для ціни використано input з типом range.

Для сортування товарів використано select. При наведенні мишка на певний товар виводиться додаткова інформація знизу. Кожен товар містить картинку, назву, опис, ціну, кнопку купити, додати в обране, в кошик, в порівняння. Фактично кожен товар є посиланням на сторінку цього товару. Очевидним покращенням буде можливість сховати панель фільтрів, яка наразі відсутня на обох сторінках.

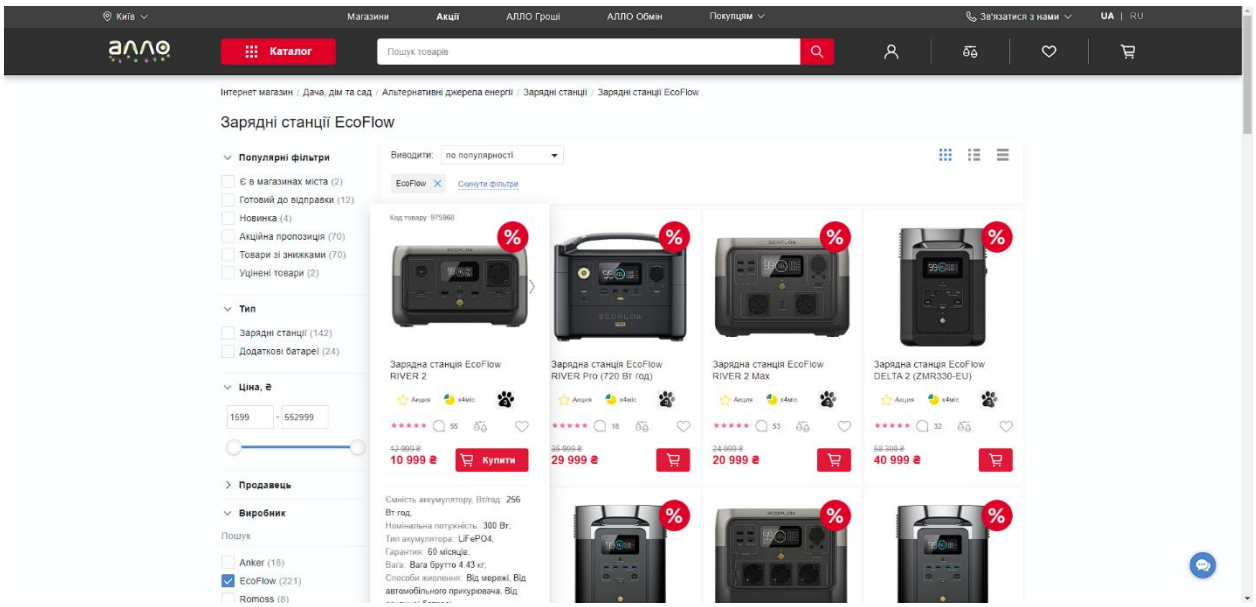


Рисунок 1.5. - Каталог товарів інтернет-магазину Allo

Після детального вивчення кількох сторінок популярних інтернет-магазинів можемо зробити висновок, що всі компоненти, визначені в попередньому розділі активно використовуються.

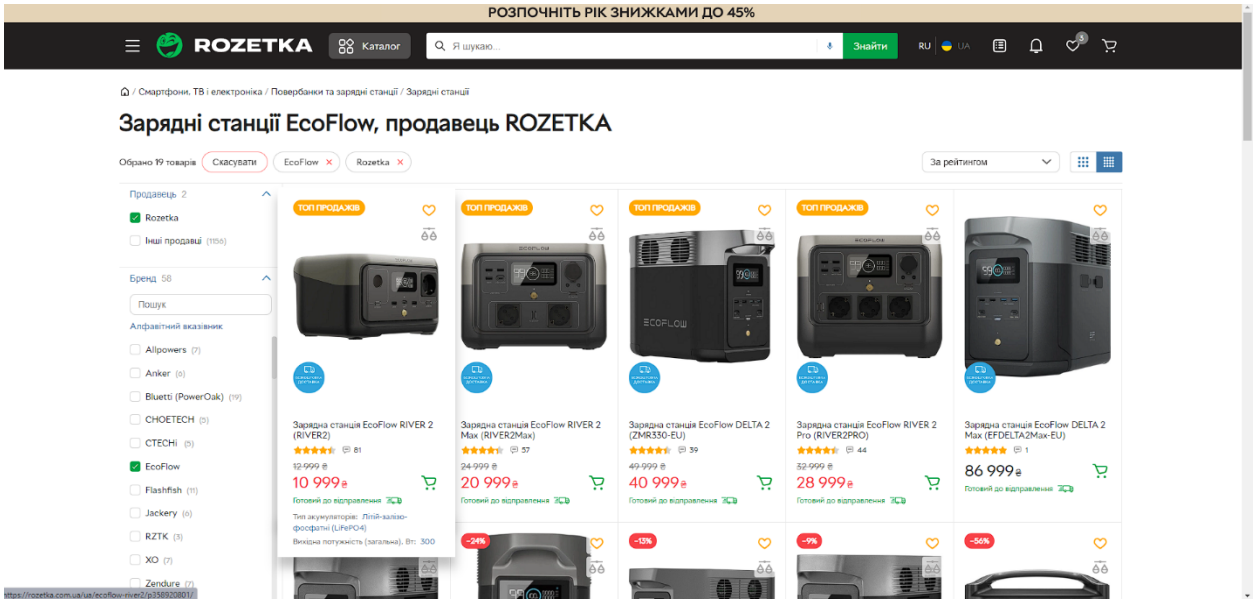


Рисунок 1.6. - Каталог товарів інтернет-магазину Rozetka

При створенні дизайну до уваги беруться такі вимоги як консистенція вигляду, обмежена кількість кольорів та шрифтів. Основні кольори сайту Rozetka – зелений, жовтий та білий. Allo використовує червоний, синій та білий кольори. На жаль, ці сайти не мають підтримки темної теми.

Більшість тексту в обох магазинах мають стиль Arial. Також на rozetka зустрічається власний стиль Rozetka.

При роботі з повільним інтернетом важлива наявність Loader компоненти, яка повідомляє користувачу, що його запит знаходиться в процесі оброблення. На Rozetka при навігації по breadcrumbs над цим компонентом пробігає зелена полоска. Це хороше рішення, але покриває лише навігацію по breadcrumbs та застосування фільтрів. На Allo присутній сірий Loader при завантаженні панелі фільтрів, а також spinner при застосуванні фільтрів.

В результаті можна зробити висновок, що дані популярні інтернет-магазини містять тільки необхідний функціонал у класичному вигляді. Отже на ринку можна створити новий продукт, який візуально буде кращим та якіснішим за існуючі аналоги.

РОЗДІЛ 2
ОБГРУНТУВАННЯ, ВИБІР ТА РЕАЛІЗАЦІЯ ІНСТРУМЕНТАРІЮ ВИРІШЕННЯ ЗАДАЧІ РЕАЛІЗАЦІЇ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ

2.1. Обґрунтування вибору мови програмування

Користувацький інтерфейс (UI) в інтернет-магазинах відіграє ключову роль у взаємодії між клієнтами та продуктами чи послугами, які вони шукають та придбають. Вірно спроектований користувацький інтерфейс може значно полегшити процес вибору та придбання товарів або послуг, роблячи його інтуїтивним та зручним. Однак створення ефективного користувацького інтерфейсу інтернет-магазину вимагає глибокого розуміння потреб користувачів, а також дотримання принципів та стандартів проектування.

У цьому розділі ми розглянемо процес проектування користувацького інтерфейсу для інтернет-магазину та акцентуємо увагу на базових компонентах та принципах, що лежать в його основі. Відповідно до сучасних тенденцій у веб-розробці та виходячи з практичних рекомендацій, ми дослідимо ключові аспекти створення інтуїтивного та ефективного інтерфейсу для забезпечення позитивного досвіду користувачів і збільшення конверсії у віртуальному магазині.

У цьому розділі ми також розглянемо основні концепції, такі як структура і навігація інтерфейсу, використання кольорів та типографіки для підсилення бренду, а також важливість взаємодії між користувачем та інтерфейсом, що може впливати на збереження клієнтів та їхню лояльність. Загалом, цей розділ розкриє ключові аспекти проектування користувацького інтерфейсу інтернет-магазину та запропонує інструменти і рекомендації для створення інтерфейсу, який задовольняє потреби користувачів та сприяє успішному функціонуванню магазину в онлайн-середовищі.

Тривалий період часу беззаперечним набором технологій для реалізації користувацьких інтерфейсів інтернет магазину є набір HTML, CSS та JS. Це основні інструменти для створення веб-інтерфейсів та UI компонентів. HTML забезпечує структуру веб-сторінки, CSS визначає її зовнішній вигляд, а JS додає інтерактивність. Це дозволяє розробникам створювати користувацькі інтерфейси, які є як ефективними, так і привабливими.

Переваги використання HTML, CSS і JavaScript:

- Простота вивчення: HTML - це мова розмітки, CSS - мова стилів, і JavaScript - мова програмування, і всі вони є відносно простими для вивчення. Це дозволяє новачкам швидко освоювати їх та створювати базові UI компоненти.
- Кросплатформенність: Веб-застосунки, побудовані на HTML, CSS і JavaScript, можуть працювати на різних платформах та браузерах, що робить їх універсальними.
- Зручна розробка та налагодження: Веб-інструменти розробки, такі як браузерні інструменти для розробників, роблять розробку та налагодження HTML, CSS і JavaScript досить зручною. Ви можете переглядати зміни в реальному часі та виправляти їх миттєво.
- Зручна анімація і взаємодія: JavaScript дозволяє створювати складні анімації та інтерактивні ефекти, що поліпшують користувацький досвід.
- Швидкість розгортання: Розгортання веб-сайту або веб-застосунку на базі HTML, CSS і JavaScript відбувається дуже швидко, оскільки вони не вимагають інсталяції на пристрої користувача.

Недоліки використання HTML, CSS і JavaScript:

- Залежність від браузера: Різні браузери можуть інтерпретувати код HTML, CSS і JavaScript по-різному, що може призвести до проблем з сумісністю. Це вимагає додаткових зусиль для забезпечення коректної роботи на різних платформах.

- Безпека: JavaScript може бути використаний для зловмисного впливу на веб-сторінки та викрадання даних користувачів. Це потребує встановлення відповідних заходів безпеки.
- Швидкість завантаження: Велика кількість JavaScript-коду або CSS може призвести до повільного завантаження сторінок, що негативно впливає на швидкість та продуктивність.
- Складність для складних компонентів: При створенні складних UI компонентів, особливо при роботі зі складними даними та станами, код може стати заплутаним та важким для розуміння.
- Залежність від серверної частини: Деякі функції, такі як обробка форм або отримання даних з сервера, вимагають взаємодії з серверною частиною за допомогою API, що може призвести до складнощів у розробці та супутніх проблем.

HTML, CSS і JS підтримуються усіма основними браузерами, включаючи Chrome, Safari, Firefox і Edge. Це означає, що користувацькі інтерфейси інтернет-магазинів, створені за допомогою цих технологій, будуть працювати коректно на всіх цих браузерах.

Chrome є найбільш популярним браузером у світі, і він підтримує всі основні функції HTML, CSS і JS. Це включає такі функції, як:

- Елементи HTML 5, такі як елементи <video> і <audio>;
- CSS 3, такі як властивості flexbox і grid;
- JavaScript, такі як ES6 і ES7.

Safari є другим за популярністю браузером у світі, і він також підтримує всі основні функції HTML, CSS і JS. Крім того, Safari має деякі унікальні функції, які можуть бути корисні для розробників інтернет-магазинів, такі як:

- Підтримка WebGL для створення 3D-графіки;
- Підтримка Touch ID і Face ID для автентифікації користувачів.

Firefox і Edge є менш популярними браузерами, але вони також підтримують усі основні функції HTML, CSS і JS. Крім того, Firefox має деякі

унікальні функції, які можуть бути корисні для розробників інтернет-магазинів, такі як:

- Підтримка WebRTC для голосового і відеозв'язку;
- Підтримка Firefox Reality для створення віртуальної реальності.

Загалом, підтримка HTML, CSS і JS популярними браузерами є дуже хорошою. Це означає, що розробники можуть бути впевнені, що їхні користувацькі інтерфейси інтернет-магазинів будуть працювати коректно на широкому спектрі пристроїв і платформ.

У підсумку, HTML, CSS і JavaScript є потужними інструментами для створення базових UI компонентів, але їх використання вимагає уваги до деталей і керування з питань сумісності та безпеки. Розробники повинні зважати на переваги та недоліки цих інструментів під час вибору підходу до створення веб-інтерфейсів.

2.2. Моделювання інформаційного наповнення інтернет-магазину

Інтернет-магазини стали популярними та важливими платформами для купівлі та продажу товарів різних категорій. Серед них важливе місце посідають інтернет-магазини велосипедів, оскільки велосипеди є екологічно чистим видом транспорту а також хобі. Важливим аспектом функціонування інтернет-магазину велосипедів є інформаційне наповнення, яке включає в себе різноманітну інформацію про продукцію, її характеристики, властивості, технічні деталі, а також іншу інформацію, яка допомагає покупцям приймати обґрунтовані рішення під час покупки велосипедів.

Велосипеди існують у різноманітних варіаціях, призначених для різних стилів катання та умов. Види велосипедів та їхній опис наведені у таблиці 2.1

Таблиця 2.1. – Опис видів велосипедів

Вид	Опис
1	2
Cross Country (XC)	Легкі велосипеди для крос-кантрі, призначені для швидкого пересування по різноманітних місцевостях. Вони добре підходять для довгих дистанцій і трас з різним рельєфом
Downhill (DH)	Спеціалізовані велосипеди для спуску з гір. Важкі та з потужними амортизаторами, вони розроблені для агресивного катання по екстремальних трасах
Road Bikes	Легкі та швидкі, вони призначені для асфальтованих доріг та довгих відстаней. Їхня геометрія спрямована на забезпечення ефективності на гладких поверхнях
Track Bikes (трекові)	Спроектовані для використання на треках із закругленнями. Легкі та мають фіксовану передачу, що робить їх ідеальними для трекового велоспорту
Cruiser Bikes (крузери)	Зручні та стильні велосипеди для повільного круїзування. Зазвичай мають великі колеса та комфортну посадку
Touring Bikes (туристичні)	Розроблені для довгих подорожей. Мають міцну раму та можливість закріплення багажу
Fat Bikes	Мають широкі покришки, призначені для катання по м’якому та нерівному грунту, такому як сніг, пісок чи болото
BMX	Легкі та маневрені велосипеди для використання на спеціалізованих трасах. Часто використовуються для виконання трюків
Trial Bikes	Легкі та з великим контролем, вони призначені для виконання трюків та подолання перешкод у міському середовищі

Продовження таблиці 2.1.

1	2
Enduro Bikes	Комбінують властивості як крос-кантрі, так і даунхілу. Дозволяють кататися як на високих гірських вершинах, так і по агресивних спусках
Шосейний велосипед (шосе)	Оптимізовані для швидкісного катання по асфальту, мають легку конструкцію і аеродинамічну геометрію
Складний велосипед	Має здатність складатися, що робить його зручним для транспортування та зберігання

На рисунку 2.1 показано візуальне зображення видів велосипедів описаних у таблиці 2.1.

Рисунок 2.1. - Види велосипедів за призначенням

І це ще не повний перелік видів велосипедів. Велосипеди для шосе, Downhill, Trial та BMX схожі лише тим, що мають два колеса. Всі інші характеристики, комплектуючі, геометрія повністю відрізняється.

Тому важливо не обмежувати користувача при виборі велосипеда базовими характеристиками, такими як колір, розмір рами, наявність підвіски і кількістю швидкостей, як це реалізовано у класичних інтернет магазинах rozetka, allo, prom.ua.

Перед розглядом конкретних аспектів інформаційного наповнення інтернет-магазину велосипедів, важливо розуміти базові інформаційні потреби покупців. Ткаченко Олександр Андрійович та Ольга Іванівна визначили наступні ключові параметри для велосипедів [6]:

- Технічні характеристики. Покупці бажають знати технічні параметри велосипеда, такі як розмір рами, тип гальма, тип трансмісії, розмір коліс, тип підвіски тощо. Ця інформація допомагає їм визначити, чи відповідає велосипед їхнім потребам;
- Опис продукту. Детальний опис велосипеду, його особливостей та переваг допомагає покупцям отримати загальне уявлення про товар;
- Фотографії та відео. Важливим елементом інформаційного наповнення є фотографії та відео велосипедів. Вони дозволяють покупцям побачити товар з різних кутів та отримати більше інформації про його зовнішній вигляд та конструкцію;
- Відгуки та рейтинги. Відгуки і рейтинги інших покупців можуть бути корисними для визначення якості та задоволеності велосипедом. Ця інформація допомагає покупцям приймати рішення;
- Інструкції та документація. Деякі покупці можуть бажати отримати докладні інструкції щодо збирання, обслуговування та користування велосипедом;
- Посилання. зв’язок з офіційним сайтом виробника надасть доступ до геометрії, полегшить процес реєстрації велосипеда а також забезпечить швидкий зв’язок з виробником у разі гарантійного випадку.

Для забезпечення ефективного інформаційного наповнення інтернет-магазину велосипедів потрібно враховувати наступні аспекти:

- Деталізований опис. Кожен велосипед повинен мати детальний та зрозумілий опис, який включає в себе технічні параметри, особливості конструкції, типи компонентів та іншу важливу інформацію;
- Високоякісні фотографії та відео. Фотографії велосипедів повинні бути високої якості і показувати деталі товару. Відео-огляди допомагають покупцям краще ознайомитися з велосипедом;
- Інтерактивні елементи. Можна використовувати інтерактивні елементи, такі як 360-градусні перегляди велосипедів, для більшого залучення покупців;
- Відгуки і рейтинги. Забезпечення можливості залишення відгуків та рейтингів покупцями допомагає покращити довіру до магазину та продукції;
- Інформаційна підтримка. Надання контактної інформації для клієнтського обслуговування та відповідей на питання покупців також є важливим аспектом інформаційного наповнення;
- Актуальність даних та працездатність посилань.

Інформаційне наповнення інтернет-магазину велосипедів відіграє важливу роль у забезпеченні успішної торгівлі велосипедами через Інтернет. Інформація, представлена на сайті, повинна бути докладною, зрозумілою та корисною для покупців, допомагаючи їм приймати обґрунтовані рішення та підвищувати довіру до магазину. Розглянуті аспекти інформаційного наповнення допомагають створити ефективну онлайн-платформу для продажу велосипедів.

2.3. Проектування сторінок та їхнього вмісту

В даному розділі будуть визначені сторінки інтернет магазину, їх візуальний вигляд та наповнення, а також використання компонентів для їх реалізації. Детальний опис кожної сторінки дозволить зрозуміти, яким чином користувач буде взаємодіяти з магазином та як забезпечити йому найвищий рівень ефективності та задоволення від користування веб-сервісом.

При огляді інтернет-магазинів Allo та Rozetka ми визначили потребу Header, Footer та Sidebar компонентів на кожній сторінці. Також потрібна головна сторінка, каталог товарів, сторінка товару, кошик та процес замовлення.

Кожна сторінка веб-сайту буде включати заголовок (header), який буде складатися з рядка елементів. По-перше, в цьому заголовку буде розміщена кнопка для відкриття бічної панелі (сайд бару), яка, у свою чергу, міститиме різноманітні функції, такі як можливість зміни мови, валюти, теми та інші налаштування. Крім того, в бічній панелі буде навігація по різним сторінкам інтернет-магазину.

По центру заголовка буде розташоване поле пошуку, яке користувач може використовувати для швидкого знаходження конкретних товарів чи інформації.

Справа від центрального поля пошуку розташовані дві іконки. Перша ікона представляє кошик для зберігання товарів, які користувач обрав для покупки. Друга ікона вказує на особистий кабінет користувача, де можуть бути доступні особисті налаштування, інформація про замовлення та інші персональні функції.

Отже, заголовок кожної сторінки інтернет-магазину міститиме зручний та функціональний набір елементів для забезпечення зручної навігації та користування всіма можливостями сайту. Він має фіксованим, тобто завжди доступним для користувача, а також мінімальним по висоті, щоб займати мінімум корисного місця.

Нижня частина кожної сторінки веб-сайту, або «footer», також буде важливою для повноцінної користувальницької досвіду. Footer буде включати ряд корисних елементів для користувача. По-перше, в нижній частині сторінки можна розмістити блок з контактною інформацією, яка включатиме адресу електронної пошти, телефон, а також можливість швидкого зв’язку через онлайн-форму.

Додатково, у футері може бути розміщений блок інформації про компанію або інтернет-магазин, включаючи історію, місію, або важливі новини. Це допомагає встановити зв’язок між користувачем і брендом та надає додаткові засоби для залучення аудиторії.

Крім того, у футері можна включити посилання на важливі сторінки, такі як Умови використання, Політика конфіденційності, та інші. Це дозволяє користувачам швидко знаходити необхідну інформацію.

Отже, footer інтернет-магазину буде функціональним блоком, який доповнює заголовок, забезпечуючи повний інформаційний спектр та зручний доступ до важливих ресурсів.

Головна сторінка є ключовою точкою входу в інтернет-магазин. На цій сторінці має бути ретельно продумана навігація, зручний пошук, а також привабливий дизайн. У цьому розділі будуть описані основні елементи головної сторінки, такі як промо-акції, розділи з новинками та популярними моделями велосипедів.

Для полегшення навігації та прискорення процесу вибору товару, на головній сторінці будуть відокремлені розділи з новинками та популярними товарами. Ці розділи міститимуть зображення товарів, назви, технічні характеристики та фактично будуть кнопками, що спрямовують на сторінки з докладнішою інформацією.

Каталог товарів є ключовим елементом інтернет-магазину, який дозволяє користувачам швидко знаходити та оцінювати доступні продукти. У цьому розділі детально розглянемо елементи та функціонал каталогу товарів.

Ліва панель каталогу буде обладнана розгортальними фільтрами, які дають користувачам можливість точно налаштувати параметри пошуку. Можливі фільтри включатимуть категорії велосипедів (гірські, шосейні, міські тощо), бренди, ціновий діапазон, розмір рами, тип гальма, та інші характеристики. Користувачі можуть згортати або розгортати фільтри залежно від своїх потреб.

У верхній частині каталогу розташовані кнопки для сортування товарів. Користувач може вибрати сортування за ціною, рейтингом, новинками чи популярністю. Додатково, інша набір кнопок дозволяє користувачам перемикати вигляд товарів між плиткою та списком, враховуючи їхні особисті вподобання та зручність.

Кожен товар у каталозі представлений якісними зображеннями, які відображають його з різних ракурсів. Під зображеннями розташована інформація про товар, така як його назва, бренд, ціна та короткий опис. Клікнувши на товар, користувач може перейти на його сторінку для отримання детальної інформації.

Щоб забезпечити швидке завантаження сторінки та зручну навігацію, використовується техніка динамічного завантаження товарів при прокрутці сторінки. Це дозволяє уникнути завантаження всього каталогу одразу, що особливо важливо в разі великої кількості товарів.

Кожен товар має опції для додавання до кошика, додавання до списку бажань та порівняння з іншими товарами. Кнопка «Деталі» або «Купити» перенаправляє користувача на сторінку товару для отримання детальної інформації чи здійснення покупки.

Кожен товар повинен мати окрему сторінку, на якій детально розглядаються всі його характеристики. В цьому розділі буде описано, як забезпечити інформативність та переконливість сторінки товару, включаючи використання відгуків, відео оглядів та інших елементів.

На сторінці кожного товару важливо забезпечити докладну інформацію, яка допоможе користувачам прийняти обдумане рішення про покупку. У цьому розділі ми розглянемо ключові елементи сторінки товару:

Зверху сторінки будуть виведені breadcrumbs - інтерактивні «хлібні крихти», які дозволяють користувачам легко навігувати назад до попередніх розділів каталогу. Це спрощує процес повернення до вибору інших товарів або категорій.

Трошки нижче верхня частина сторінки товару буде обладнана табами навігації, які дозволять користувачам легко переміщатися між різними секціями та отримувати швидкий доступ до ключової інформації. Кожен таб буде містити посилання на відповідну секцію сторінки.

Також тут буде міститись інформація про ціну та кнопки додати в кошик та купити. Ці кнопки дозволяють користувачам швидко взаємодіяти з товаром та перейти до процесу замовлення безпосередньо з цієї сторінки.

Зустрічати користувача буде галерея фотографій велосипеда, яка надає користувачам можливість подивитися на товар з різних ракурсів та докладно розглянути його дизайн і компоненти.

Секція «Опис продукту» містить докладний опис велосипеда, його особливостей та переваг. Тут може також бути включена історія бренду, важливі подробиці та особливості дизайну.

Секція «Технічні характеристики» містить вичерпний перелік технічних параметрів велосипеда. Це може включати розмір рами, тип гальм, кількість швидкостей, тип підвіски (якщо це гірський велосипед), та інші характеристики, які важливі для покупців.

Секція «Відгуки та рейтинги» дозволяє користувачам залишити свої враження від товару та оцінити його. Рейтинги та відгуки інших покупців можуть бути впливовими факторими для придбання товару.

Секція «Посилання на сайт виробника» надає можливість користувачам дізнатись інформацію про цей вибір, надану виробником.

На сторінці «Кошик та Процес Замовлення» користувач отримає зведений список всіх доданих товарів до кошика. Кожен товар буде відображений зі своєю назвою, фотографією, ціною та вказаною кількістю. Для зручності користувача буде можливість редагувати кількість товару, а також вибирати колір та розмір, якщо це необхідно.

Розташована поруч з кожним товаром буде кнопка для видалення товару з кошика, щоб користувач міг легко коригувати свій вибір перед оформленням замовлення.

Під зведеним списком товарів вказується загальна вартість усіх доданих товарів. Користувач може переглядати та редагувати вміст кошика, змінюючи кількість товарів або видаляючи їх.

Після завершення вибору товарів та редагування кошика, користувач має можливість перейти до оформлення замовлення. Для цього на сторінці кошика розташована кнопка «Оформити Замовлення». Після кліку на цю кнопку з’явиться модальне вікно, в якому користувач буде запрошений вказати свої контактні дані, обрати спосіб доставки та оплати.

У модальному вікні користувач також матиме можливість перевірити і остаточно підтвердити свій вибір, переглянути загальну вартість замовлення та зробити останній погляд перед підтвердженням покупки. Після введення всіх необхідних даних та підтвердження замовлення, інформація буде відправлена для обробки, і користувач отримає підтвердження та інформацію про його замовлення.

РОЗДІЛ 3
РЕЗУЛЬТАТИ ВИРІШЕННЯ ЗАДАЧІ РЕАЛІЗАЦІЇ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ

3.1. Аналіз інструментів реалізації компонентів користувацьких інтерфейсів інтернет магазину

Наш дослідницький об’єкт - інтернет-магазин велосипедів, представляє собою складну систему, яка вимагає високоякісного користувацького інтерфейсу, щоб забезпечити комфортність та задоволеність клієнтів під час їхніх покупок. У цьому розділі ми розглянемо технології, методи та інструменти, які були використані для створення інтерфейсу нашого велосипедного інтернет-магазину, а також поділимось важливими кроками розробки.

Програмна реалізація базується на ретельному аналізі потреб користувачів та наших бізнес-цілей. Ми врахували сучасні практики розробки користувацьких інтерфейсів, такі як мінімалізм, адаптивність та інші, для забезпечення зручності та ефективності взаємодії користувачів з магазином.

У цьому розділі ми оглянемо, порівняємо та надамо оцінку результатів програмної реалізації та сформуємо можливості для подальшого вдосконалення нашого інтерфейсу для покращення користувацького досвіду та досягнення більшої конкурентоспроможності на ринку інтернет-торгівлі велосипедами.

Для реалізації користувацьких інтерфейсів доступно багато інструментів, які мають свої переваги та недоліки. Найпопулярніші інструменти:

- HTML, CSS, JavaScript;
- React;
- Bootstrap;
- Material UI.

HTML, CSS та JavaScript визнані найкращими інструментами для створення користувацьких інтерфейсів інтернет-магазинів з численних причин.

HTML, CSS та JavaScript відповідають принципу «Single Responsibility» (одна відповідальність), оскільки кожен з цих мов виконує конкретну функцію в розробці веб-додатків, розділяючи відповідальності за різні аспекти веб-сторінок.

HTML (Hypertext Markup Language) відповідає за структуру та семантику веб-сторінки, розміщення та організацію контенту. Він визначає, як інформація повинна бути представлена на сторінці, але не займається її стилізацією чи логікою взаємодії. Він визначає структуру веб-сторінок, дозволяючи розташовувати та організовувати контент, такий як товари, категорії та інше.

CSS (Cascading Style Sheets) відповідає за вигляд та стилізацію елементів, які визначені в HTML. Відокремлюючи стилі від структури, CSS дозволяє змінювати вигляд сторінки без втрати самої структури. Це робить можливим легке внесення змін у дизайн і забезпечує консистентність вигляду на всій веб-сторінці чи веб-сайті. Він відповідає за стиль і вигляд сторінки, надаючи можливість легко налаштовувати дизайн і розміщення елементів.

JavaScript відповідає за програмну логіку та динаміку взаємодії на сторінці. Він додає інтерактивність, обробку подій та можливість динамічно змінювати вміст сторінки відповідно до дій користувача. JavaScript інтегрується з HTML та CSS, але його основна роль полягає в забезпеченні функціональності та взаємодії, відділеної від структури та стилізації. JavaScript відіграє ключову роль у покращенні інтерактивності сторінок, що дозволяє реалізовувати динамічні функції, такі як додавання товарів до кошика, фільтрація товарів за категоріями, а також валідація форм для зручного оформлення замовлень.

Цей розподіл відповідальностей дозволяє розробникам ефективно працювати над різними аспектами веб-розробки, полегшуючи розуміння,

підтримку та розширення коду. Одночасно це сприяє підтримці принципу модульності та чіткості, що робить код більш читабельним і піддається модифікаціям. Трійця HTML, CSS і JavaScript спільно дозволяє створити вражаючі та ефективні інтерфейси для користувачів Ranjan Alok, Abhilasha Sinha та Ranjit Battewad детально описали всі аспекти реалізації веб аплікацій з використанням даного набору технологій [7].

Існують інші інструменти створення користувацьких інтерфейсів, але деякі з них можуть бути менш ефективними або мають обмеження порівняно з технологіями HTML, CSS та JavaScript:

- Flash. Flash колись був популярним інструментом для створення анімацій та інтерактивності. Проте, з поширенням мобільних пристроїв і забороною використання Flash у багатьох сучасних браузерах, цей метод втратив актуальність. Він не забезпечує таку широку сумісність та доступність як HTML, CSS та JavaScript;
- Вбудовані додатки. Використання вбудованих додатків або плагінів також може бути обмеженим. Користувачам може бути не зручно завантажувати та встановлювати додаткові компоненти для перегляду інтернет-магазину, що може вплинути на їхню прихильність та участь;
- Серверний рендерінг. У деяких випадках веб-сайти можуть використовувати техніку серверного рендерінгу, де сторінка формується на сервері та відправляється на клієнтський браузер. Це може бути причиною меншої динамічності та інтерактивності, оскільки кожна зміна вимагає нового запиту на сервер;
- Інші технології. Інші технології, такі як Silverlight чи JavaFX, були використані у минулому, але вони також стали менш популярними через обмежену сумісність із сучасними пристроями та забезпеченням безпеки.

У порівнянні з іншими підходами, використання HTML, CSS та JavaScript дозволяє забезпечити ефективну розробку, підтримку та розширення функціоналу інтернет-магазину. Їхній відкритий характер, широкі

можливості та зручність роблять їх переважною вибором для більшості розробників у галузі електронної торгівлі.

React - це популярна бібліотека JavaScript для створення інтерфейсів користувача. Вона часто використовується для реалізації базових UI компонентів, таких як кнопки, тексти, форми і т. д. Також це ціла екосистема, яка дозволяє побудувати цілий проект на основі даної бібліотеки, наприклад create-react-app. Є два підходи для роботи з react, а саме SPA та SSR.

SSR використовується для генерації HTML на стороні сервера та відправки готової сторінки користувачеві. При роботі з Single Page Application (SPA) HTML генерується на стороні клієнта з використанням JavaScript.

Один із популярних шаблонізаторів для серверного рендерингу - це EJS (Embedded JavaScript). EJS дозволяє вставляти JavaScript-код прямо в HTML, спрощуючи вбудовування змінних та структурного коду. Це створює динамічні сторінки, які можуть легко пристосовуватися до змінних оточення сервера.

Next.js - це фреймворк React для створення універсальних (Universal) або іншими словами, серверних, додатків. Використовуючи Next.js, можна здійснювати серверний рендеринг React-компонентів, що поліпшує швидкість завантаження та забезпечує більше оптимальний SEO.

Lazuardy Mochammad Fariz Syah та Dyah Anggraini детально вивчили дане питання [8]. Обидва підходи мають свої переваги та недоліки. SSR може поліпшити витрати часу на завантаження сторінки, оскільки клієнт отримує вже готовий HTML від сервера. Це може позитивно впливати на SEO, а також на відчуття користувача при першому завантаженні сторінки. З іншого боку, SPA використовує асинхронні запити для оновлення лише частин сторінки, що може зменшити обсяг передаваних даних та поліпшити взаємодію з користувачем під час використання додатку. Проте, SPA може мати більше проблем із SEO, оскільки пошукові системи можуть мати обмежений доступ до вмісту, який генерується динамічно за допомогою JavaScript.

Для інтернет-магазинів важливо вибирати підхід, який відповідає конкретним потребам проекту. Якщо SEO та швидкість завантаження сторінок є критичними, SSR може бути більш підходящим варіантом. Однак, якщо важлива динамічність та інтерактивність без повторного завантаження сторінок, то SPA може бути оптимальним вибором.

Насправді всі три варіанти, а саме React як представник SPA, Next.js, як представник SSR використовують React для побудови користувацьких інтерфейсів, тому ми можемо знехтувати різницею між ними.

Bootstrap - це відкрите програмне забезпечення, або фреймворк, для розробки веб-сайтів та веб-додатків. Він базується на HTML, CSS та JavaScript і надає набір готових компонентів, стилів та скриптів, що значно полегшує розробку та забезпечує єдність вигляду.

Gaikwad S. Shahu та Adkar здійснили детальний огляд bootstrap фреймворка [9]. Однією з ключових особливостей Bootstrap є його адаптивність (responsive design), що означає, що створені за його допомогою інтерфейси автоматично адаптуються до різних розмірів екранів, що робить його ідеальним вибором для розробки інтернет-магазинів. Це особливо важливо, оскільки користувачі мають різні пристрої, такі як комп’ютери, планшети та смартфони, і їхній комфорт при використанні магазину на будь-якому пристрої є важливим фактором.

Bootstrap також має широкий спектр готових компонентів, таких як кнопки, форми, навігаційні панелі, каруселі, модальні вікна та багато інших, що можна легко і швидко використовувати для створення інтерфейсу магазину. Це робить розробку ефективною, а також дозволяє забезпечити консистентність та професійний вигляд сторінок.

Окрім цього, Bootstrap підтримує сучасні браузери і включає в себе важливі компоненти JavaScript, такі як jQuery, які допомагають реалізовувати динамічні функції без необхідності написання великої кількості коду самостійно.

Узагальнюючи, Bootstrap є потужним інструментом для швидкої і ефективної розробки користувацьких інтерфейсів інтернет-магазину, особливо якщо важливі аспекти, такі як адаптивність, консистентність та швидкість розробки, є високо пріоритетними. Його можна назвати готовою бібліотекою, яка працює на основі HTML, CSS та JS.

Material-UI - це бібліотека компонентів інтерфейсу користувача для React, яка реалізує дизайн-систему Material Design від Google. Це дизайн-мова, розроблена компанією Google для створення консистентних та привабливих інтерфейсів веб-сайтів та мобільних додатків. Вона була вперше представлена у 2014 році. Основна ідея Material Design полягає в створенні платформенно-незалежних елементів інтерфейсу, які виглядають однаково добре на різних пристроях та розмірах екранів. Material Design став широко відомим своєю чіткою, консистентною та простою дизайн-мовою, яка сприяє створенню високоякісних та естетично приємних інтерфейсів.

Material-UI надає широкий набір готових до використання компонентів, таких як кнопки, тексти, таблиці, картки, та багато інших, які дотримуються принципів Material Design. Це значно полегшує створення стилізованих та консистентних користувацьких інтерфейсів.

Material-UI часто використовується для створення інтерфейсів інтернет-магазинів, оскільки вона пропонує рішення, які допомагають створювати сучасні та зручні інтерфейси швидко і з мінімальними зусиллями. Ця бібліотека може ефективно використовуватися для створення продуктивних та атрактивних інтернет-магазинів, забезпечуючи високий рівень користувацького досвіду.

У таблиці 3.1 подано опис переваг роботи з Material-UI згідно з дослідженнями Guo Lin [10].

Таблиця 3.1. - Переваги роботи з Material-UI

Назва переваги	Опис
Спрощений дизайн	Material-UI дозволяє швидко та легко створювати інтерфейси, які дотримуються принципів Material Design. Це включає в себе чітке визначення кольорів, тіней, та анімацій, що робить інтерфейси сучасними та естетично привабливими
Гнучкі компоненти	Material-UI пропонує різноманіття готових компонентів, які можна легко налаштовувати та інтегрувати в зручний спосіб. Це дозволяє розробникам швидше будувати та модифікувати інтерфейс за допомогою готових рішень
Адаптивний дизайн	Material-UI розроблено з урахуванням адаптивності, що означає, що інтерфейси можуть оптимально виглядати на різних пристроях та розмірах екрану, що важливо для інтернет-магазинів з різноманітними аудиторіями
Розвинена спільнота та документація	Material-UI користується великою та активною спільнотою розробників. Це означає, що завдяки обміну знаннями та розширеним документаційним ресурсам, розробники можуть ефективно вирішувати завдання та швидше розбиратися в функціональності бібліотеки

3.2. Визначення компонентів та характеристик порівняння

В другому розділі було визначено елементи та функціонал сторінок. Для їх реалізації потрібні використати наступні компоненти.

На заголовку (header) кожної сторінки:

- Sticky (фіксований) - елементи, які залишаються на місці під час прокрутки сторінки;
- Side bar (бічна панель) - вертикальна панель з боку сторінки для навігації або відображення інших елементів;
- i18next - бібліотека для локалізації (перекладу) веб-сайту на різні мови;
- Конвертер валют - інструмент для перетворення вартості товарів чи послуг з однієї валюти в іншу;
- Навігація - система, яка дозволяє користувачам переходити між різними сторінками або розділами веб-сайту;
- Пошук - поле для введення запитів для пошуку інформації на веб-сайті;
- Іконка - графічний символ, який представляє собою певний об’єкт чи дію.

На нижній частині (footer) кожної сторінки:

- Email - відомості про електронну пошту для зв’язку;
- Номер телефону - відомості про контактний телефон для зв’язку;
- Текстове поле - область для введення тексту;
- Посилання - елемент, який веде до інших сторінок чи ресурсів.

На головній сторінці:

- Зображення - графічний елемент для візуального представлення інформації;
- Кнопка - елемент для виклику певної дії при натисканні;

- Фільтр - інструмент для обмеження відображення вмісту за певними критеріями;
- Список - перелік елементів чи об’єктів;
- Типографія - стиль та вигляд тексту на сторінці;
- Розгортання/згортання - можливість розширення чи скорочення вмісту;
- Load more (завантажити ще) - опція для підвантаження додаткового вмісту.

На сторінці продукту:

- Breadcrumbs (хлібні крихти) - шлях, який вказує на розташування сторінки у веб-сайті;
- Tabs (вкладки) - розділи для відображення різних аспектів продукту чи інформації;
- Перемикач - елемент для вибору між кількома станами або опціями;
- Checkbox (прапорець) - елемент для вибору одного чи декількох варіантів;
- Радіокнопка - елемент для вибору лише одного варіанту зі списку візуально видимих опцій;
- Модальне вікно - вікно, яке виходить поверх основного вмісту для відображення додаткової інформації чи виклику дії;
- Аватар - зображення користувача або представлення особи.

Для порівняння інструментів реалізації компонентів створено набір параметрів оцінювання. На рисунку 2.2 зображена схема, що містить список з одинадцяти параметрів оцінювання реалізованих компонентів.

Рисунок 2.2. - Параметри оцінювання реалізованих компонентів

Читабельність (Readability). Читабельність коду є ключовим параметром в оцінці якості компонентів в JavaScript. Чим зрозуміліший код, тим легше його супроводжувати та розвивати. Чітка структура, зрозумілі імена змінних та функцій, а також дотримання код-стайлу сприяють поліпшенню читабельності.

Масштабованість (Scalability). Масштабованість визначає здатність компонента пристосовуватися до зростання розміру проекту без втрати продуктивності та збереження функціональності. Гнучка архітектура, яка дозволяє легко додавати новий функціонал та розширювати можливості, є ключовим аспектом масштабованості.

Реюзабельність (Reusability). Реюзабельність вказує на те, наскільки легко можна використовувати компоненти в інших частинах програми чи навіть у інших проектах. AlOmar Eman Abdullah у свох дослідженням навів необхідність забезпечення реюзабельності а також навів способи рефакторингу існуючого коду при потребі перевикористати певні частини коду [11]. Інтерфейси та функції повинні бути створені таким чином, щоб було зручно їх перевикористовувати в різних контекстах.

Відповідність принципам SOLID. Відповідність принципам SOLID (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion) гарантує створення гнучких, зручних для розширення та підтримки компонентів. Arora G. Kumar у своїй праці “SOLID Principles Succinctly.” Детально розібрав всі принципи, навів приклади та покрив більшість випадків [12]. Ці принципи сприяють високій якості коду та покращують його структуру.

Cohesion та Coupling. Cohesion визначає, наскільки пов’язані між собою елементи компонента, а coupling оцінює ступінь залежності між різними компонентами. Висока cohesion і низький coupling свідчать про добре спроектований код, де компоненти розташовані логічно та мають мінімальні взаємозалежності. Candela Ivan описав способи досягнення високого cohesion і низького coupling, навів таблицю з чотирьох можливих комбінацій, пояснив причини та способи досягнення [13].

Ймовірність виникнення помилок (Fault Tolerance). Ймовірність виникнення помилок визначає, наскільки стійкий до помилок є компонент. Ефективне управління винятками та здатність обробляти непередбачувані ситуації сприяють покращенню якості компоненту.

Розширення (Extensibility). Розширюваність вказує на здатність додавати новий функціонал чи модифікувати існуючий без значних змін у вихідному коді. Гнучка архітектура, що підтримує розширення, є важливою для довгострокового розвитку проекту. Lerner Benjamin дослідив всі особливості імплементації цього параметру у світі веб-технологій [14].

Перевикористання (Adaptability). Перевикористання визначає, наскільки легко можна адаптувати компонент для використання в різних середовищах чи у різних умовах. Гнучкість та адаптивність компоненту важливі для успішної інтеграції в різноманітні сценарії використання.

Продуктивність (Performance). Продуктивність визначається ефективністю роботи компонента в умовах великого обсягу даних чи завдань. Вимірюється часом виконання операцій та оптимізацією ресурсів, таких як

пам’ять та процесор. Ефективний код забезпечує швидку відповідь та оптимальне використання ресурсів.

Враження користувача та розробника (User/Developer Experience). Якість компоненту визначається не лише його технічними аспектами, але й тим, наскільки комфортно користувачі можуть взаємодіяти з продуктом, а розробники можуть розробляти та супроводжувати код. Інтуїтивний інтерфейс для користувачів та чистий код для розробників впливають на загальне враження від продукту.

Консистенція (Consistency). Консистенція визначає стабільність та однорідність у дизайні та функціональності компонента. Єдність інтерфейсу та стандартизація процесів сприяють легкості використання та розумінню компоненту, що є важливим для забезпечення однакового досвіду для користувачів та розробників усередині системи. Anjorin Anthony вивчив це питання зі сторони розробника, а не дизайнера, тобто визначив ключові аспекти, які потрібно знати розробнику при реалізації користувацьких інтерфейсів з підтримкою консистенції, а також наголосив на важливості прийняття участі в розробці дизайну розробників [15].

На наступному буде реалізована частина зазначених вище компоненті з використанням технологій, визначених у першому підрозділі цього розділу. Під час реалізації буде аналізуватись ефективність кожної з цих технологій, визначено їх переваги та недоліки в контексті наших завдань. Окрім того, ми проведемо порівняльний аналіз між використанням різних технологій для забезпечення оптимального вибору та оптимізації нашого коду. Цей підхід дозволить нам створити ефективний та добре оптимізований набір компонентів для нашого проекту.

3.3. Порівняльний аналіз реалізації компонентів

3.3.1. Компонент кнопка

Розглянемо компонент кнопка або Button. Цей компонент дуже часто зустрічається, забезпечує взаємодію користувача зі сторінкою, має обробники подій та реагує певним чином на цю подію.

Компонент користувацького інтерфейсу кнопки повинен забезпечувати ряд важливих характеристик для зручного та ефективного використання. Основні параметри наведені у таблиці 3.2.

Таблиця 3.2. - Параметри компонента кнопка

Параметр	Опис
Типи та Розміри	Важливо мати можливість вибрати різні типи та розміри кнопок залежно від потреб дизайну та використання на конкретній сторінці чи в додатку. Наприклад, можливість вибрати велику чи маленьку кнопку
Кольори та Стани	Забезпечення можливості налаштування різних кольорових схем для звичайного стану, стану при наведенні (hover), та стану при натисканні (click). Це дозволяє створювати візуально привабливі та інтуїтивні ефекти для користувачів
Обробка Подій	Кнопка повинна підтримувати обробку різних подій, таких як клік чи інші дії, щоб розробник міг визначити необхідну логіку відповіді на події користувача
Включення/ Виключення	Можливість встановлення стану «включено/ виключено» для кнопки, щоб контролювати доступність та можливість взаємодії з нею
Іконки та Текст	Підтримка вставки тексту чи іконок в кнопку, щоб надати додаткову інформацію чи контекст

HTML містить два варіанти створення кнопки: <button> та використання атрибутів type=”button”, role=”button”.

Елемент <button> є стандартним HTML-тегом, який використовується для створення кнопок. Цей тег призначений для інтерактивного взаємодії з користувачем. Його використовують як частину форми для відправлення даних, або для запуску JavaScript-функції при кліку. Елемент <button> може мати вкладені елементи або текст, і його зовнішній вигляд легко налаштовується за допомогою CSS [16].

Альтернативний варіант - використання атрибутів type=”button” та role=”button”.. Елемент з такими атрибутами симулює поведінку кнопки. Використання з атрибутом type=”button” може бути доцільним у випадках, коли розробник має особливі вимоги щодо стилізації та має потребу створювати власний вигляд кнопок, що не підпадає під стандартні стилі кнопок. Однак важливо розуміти, що такий підхід може вводити в оману інших розробників, які будуть читати або редагувати код, оскільки природно не пов’язаний із створенням кнопок. Розробники повинні обережно використовувати цей метод, ретельно документувати свій код та враховувати можливі непорозуміння, які можуть виникнути в результаті використання для симуляції кнопок. Зазвичай для створення кнопок рекомендується використовувати стандартний тег <button>, оскільки він забезпечує чітку семантику та вбудовані функціональні можливості [17].

Для реалізації кнопки можна визначити два варіанта. У першому ми просто вставляємо тег <button> в HTML, додаємо йому клас та через JavaScript прикріплюємо обробник подій. Цей метод працює добре для одноразового використання, але при необхідності створення багатьох подібних кнопок виникає проблема з реюзабельністю. Копіювання та вставлення коду може призвести до дублювання логіки та складнощів у підтримці.

Другий варіант передбачає використання патерну модуль для ізоляції логіки кнопки. Створення окремого JS файлу, який експортує функцію для створення кнопки на основі вхідних параметрів, дозволяє забезпечити

реюзабельність. Кожен екземпляр кнопки створюється незалежно, а логіка кнопки залишається прихованою всередині модулю. Це полегшує тестування, підтримку та розширення функціоналу.

Підхід з патерном модуль дозволяє вдосконалити структуру коду, забезпечуючи чистоту та легкість розуміння для інших розробників. Розширення можливостей кнопки за допомогою додаткових параметрів чи методів стає більш простим, і це підвищує гнучкість компонента. Можливість написання юніт-тестів дозволяє підтримувати високу якість коду та зменшити ймовірність виникнення помилок.

Тому ми обираємо другий варіант. Код реалізації кнопки наведений у додатку А. В ньому є три файли.

У файлі buttonComponent.js ми реалізуємо функцію addButton, яка створює елемент кнопки та налаштовує його відповідно до переданих параметрів. Функція отримує тип кнопки (type), розмір (size), обробник події при кліці (onClick), властивість «включено/виключено» (isEnabled), вміст (текст чи іконку) (children), та батьківський елемент, до якого додається кнопка (parent). За допомогою функцій DOM ми створюємо кнопку, додаємо до неї класи стилів відповідно до типу та розміру, встановлюємо вміст та обробник події.

У файлі buttonStyles.css ми визначаємо стилі для різних типів та розмірів кнопок. Наприклад, для типу «primary» ми встановлюємо фоновий колір та колір тексту, а для розміру «large» встановлюємо більший розмір шрифту. Ці класи стилів використовуються в компоненті кнопки для стилізації.

У файлі main.js ми імпортуємо функцію addButton з файлу buttonComponent.js та використовуємо її для створення кнопки. Ми вказуємо параметри, такі як тип «primary», розмір «large», обробник події (() => alert(‘Button Clicked’)), властивість «включено» (true), текст кнопки («Click me!»), та батьківський елемент, до якого додається кнопка (parentElement). Таким чином, ми викликаємо функцію та створюємо кнопку з вказаними параметрами.

Компонент button не передбачає великої кількості роботи з DOM. Але це може стати проблемою при великих обсягах роботи з DOM. Для вирішення цієї проблеми можна використати jQuery. Ця бібліотека надає короткі та зручні методи для вибору, маніпулювання та анімації елементів на сторінці. Замість написання декількох рядків коду на чистому JavaScript для виконання операцій з DOM, вам може знадобитися лише один короткий вираз у jQuery [18].

Код наведений у додатку А розділі Button in jQuery.

У цьому коді використана бібліотека jQuery для полегшення маніпуляцій з DOM та створення елементів. Використовуючи $(‘<button></button>’), ми створюємо новий елемент кнопки. Методи addClass, html та on додають класи стилів, встановлюють текст чи іконку, та додають обробник події, відповідно. Метод prop встановлює властивість «включено/виключено», і append додає кнопку до батьківського елемента за допомогою jQuery.

Таким чином ми відійшли від чистого JS для пришвидшення написання коду, читабельності, покращення developer’s experience. Але збільшиться розмір бандла, адже ця бібліотека містить багато інших інструментів, які ми не використовуємо, але вони реалізовані і доступні у файлі. До того ж сучасні версії JavaScript (ES6+) включають багато функцій, які раніше були унікальні для jQuery (наприклад, вирази стрілкової функції, проміси, деструктуризація). З випуском цих функцій чистий JavaScript стає більш потужним і менш потребує використання додаткових бібліотек.

Bootstrap містить готові стилі для компонента кнопка. Розглянемо можливості кнопки в Bootstrap. На рисунку 3.1. зображено готові типи кнопок, зокрема primary, secondary, success, danger, warning та info. Кожен тип містить свій колір заливки та тексту.

HTML

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
```

Рисунок 3.1. - Готові типи кнопок

На рисунку 3.2. зображено види обведення кнопок, зокрема primary, secondary, success, danger, warning та info. Кожен тип визначає колір тексту та меж.

HTML

```
<button type="button" class="btn btn-outline-primary">Primary</button>
<button type="button" class="btn btn-outline-secondary">Secondary</button>
<button type="button" class="btn btn-outline-success">Success</button>
<button type="button" class="btn btn-outline-danger">Danger</button>
```

Рисунок 3.2. - Обведення кнопок

На рисунку 3.3. зображено два розміри: small і large.

HTML

```
<button type="button" class="btn btn-primary btn-lg">Large button</button>
<button type="button" class="btn btn-secondary btn-sm">Small button</button>
```

Рисунок 3.3. - Розміри кнопок

Також bootstrap вбудовані іконки. Для цього достатньо додати тег з назвою “i” і прокинути в атрибут class ="bi bi-star-fill", де bi - це префікс для іконок Bootstrap. bi-star - це клас самої іконки "зірка".

Стан isEnabled можна встановити html атрибутом “disabled”. Потрібно тільки додати onClick handler для кнопки.

Таким чином bootstrap покриває більшість потреб компонента кнопка, які можна налаштовувати одразу в html розмітці [19].

React - це бібліотека для створення інтерфейсів користувача веб-додатків. Однією з ключових концепцій React є розбиття інтерфейсу на компоненти. Тому створимо компонент Button. Код наведений у додатку А розділі Button in React.

Розглянемо файл Button.jsx.

На початку файла знаходиться імпорт бібліотек:

- React: Імпорт базової бібліотеки React, необхідної для створення React-компонентів. З версії React 17.0.0 і вище необов'язково додавати імпорт React у файлах компонентів. З появою версії 17, React може визначати React автоматично, навіть якщо ви не імпортуєте його у кожному файлі окремо;
- PropTypes: Імпорт бібліотеки для визначення типів властивостей (props) компонента. PropTypes можна замінити на TypeScript, який надає статичну типізацію для властивостей компонентів. TypeScript дозволяє визначати типи для пропсів, які автоматично перевіряються на етапі компіляції, забезпечуючи більшу безпеку та допомагаючи у виявленні помилок ще до виконання програми [21];
- cx: Імпорт бібліотеки для зручного комбінування класів CSS з урахуванням умов;
- styles: Імпорт об'єкта стилів, який, містить CSS-класи для стилізації кнопки.

Потім оголошується функціональний компонент Button, який приймає ряд властивостей (props) для налаштування вигляду та поведінки кнопки.

Властивості, які можна передавати:

- type: Тип кнопки;
- size: Розмір кнопки;
- onClick: Обробник події кліку на кнопку;
- disabled: Флаг, який вказує, чи має бути кнопка вимкненою;
- className: Додаткові CSS-класи для кнопки;
- children: Вміст кнопки (текст, іконка, інший React-елемент).

Після цього створюється набір класів CSS для кнопки. Для цього використовується cx для зручного об'єднання CSS-класів залежно від значень type, size, та className.

Функція повертає JSX з кнопкою <button>, до якої застосовані стилі залежно від значень btnClassName, disabled, та onClick.

Вміст кнопки визначається через children.

Вкінці знаходяться PropTypes та defaultProps:

Визначається структура та типи властивостей (propTypes), які може приймати компонент. Задаються значення за замовчуванням для властивостей, які можуть не бути передані (defaultProps).

Компонент Button в MUI фактично є готовим компонентом, реалізованим на React з підтримкою Material UI Theme.

На рисунку 3.4. зображені типи, зокрема text, contained і outlined. Вони визначають колір тексту, фону і меж.

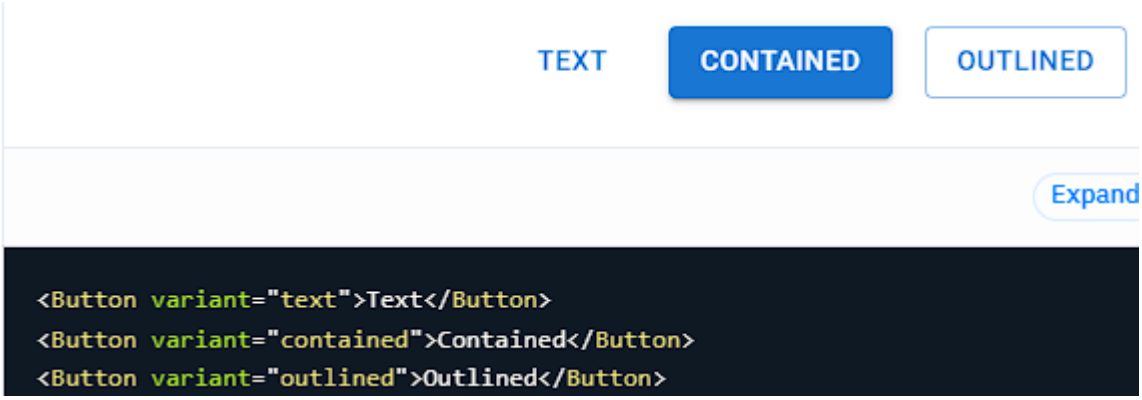


Рисунок 3.4. - Типи кнопки

На рисунку 3.5. зображені розміри кнопки, що задаються через проп size.

Рисунок 3.5. - Розміри кнопки

На рисунку 3.6. зображена можливість додати іконки через проп startIcon або sendIcon.

Рисунок 3.6. - Іконки

Також цей компонент підтримує onClick хендлер та disabled стан.

За потреби через пропс sx можна додати стилі вручну.

Важливою перевагою роботи з MUI є її підтримка тем. Ми можемо обгорнути наш продукт ThemeProvider, в який прокинути власну тему. І після цього вона буде доступна у всіх компонентах. Для прикладу ми можемо стилізувати нашу кнопку використовуючи функцію styled.

Щоб використати власні стилі та тему, потрібно використати функцію styled разом із компонентом.

Код наведений у додатку А розділі Styled Button with MUI Theme.

В цьому файлі StyledButton - це компонент, який використовує styled для визначення власних стилів. У функції, переданій styled, ми отримуємо доступ

до теми через параметр { theme }, і можемо використовувати його для стилізації компонента.

Потім MyStyledButton використовує цей стилізований StyledButton та передає всі передані пропси, включаючи children.

3.3.2. Інші компоненти користувацького інтерфейсу

Важливо зазначити, що Jquery, Bootstrap, React, Material UI базуються на фундаментальних веб-технологіях: HTML, CSS та JavaScript. І хоча використання Jquery, Bootstrap, React або Material UI робить розробку швидше і зручніше завдяки готовим рішенням та абстракціям, всі їхні можливості можна реалізувати вручну з використанням базових веб-технологій.

Таким чином всі визначені компоненти можна реалізувати на HTML, CSS та JS по аналогії з реалізацією компонента Button.

Bootstrap, не є тільки CSS надбудовою, що надає стилізацію. Він також використовує JavaScript для додавання додаткової функціональності та взаємодії на сторінках. JavaScript використовується для реалізації різноманітних компонентів та інтерактивності, а також для взаємодії з DOM (Document Object Model).

Один із прикладів, де JavaScript використовується в Bootstrap, - це модальні вікна. Вони можуть бути викликані та управлятися за допомогою JavaScript, щоб створювати плавні ефекти відкриття/закриття, обробляти події та взаємодіяти з користувачем.

Також Boostrap має такі компоненти, як DropDown, який обробляє логіку відкриття/закриття списку, Carousel, в якому JavaScript використовується для створення слайд-шоу та його управління.

Таким чином Bootstrap дозволяє створити значну частину визначених компонентів. Для реалізації інших потрібна робота з JS.

В React, робота з DOM-елементами може бути виконана за допомогою хуків, таких як useRef. useRef дозволяє створювати посилання на DOM-

елемент та отримувати доступ до його властивостей і методів. За допомогою useRef, ми можемо відійти від стандартного підходу React і працювати з DOM-елементами, як в чистому JavaScript.

Зокрема, useRef може бути використаний для отримання доступу до DOM-елементу після того, як він був відмальований на сторінці. Наприклад, ми можемо використовувати useRef для отримання посилання на елемент форми, поле вводу чи будь-який інший елемент. Це дозволяє звертатися до цих елементів напряму, маніпулювати їхніми властивостями та слухати події безпосередньо через звичайні DOM-методи та події.

Щоб використати useRef, потрібно спочатку створити посилання за допомогою const myRef = useRef(null);, а потім призначити її до ref атрибута в JSX. За допомогою створеної змінної, ми можемо отримати доступ до DOM-елементу та виконувати з ним різні операції.

Наприклад при реалізації компонента Tooltip важливо відображати його в межах видимої області екрану користувача.

Код файлу Tooltip.jsx наведений у додатку А розділі Styled Button with MUI Theme.

Цей компонент Tooltip є простим React-компонентом, який реалізує відображення підказки при наведенні на деякий елемент. Цей компонент дозволяє прокинути elementToShowOnHover, який буде відображатися у вигляді підказки, коли користувач наводить курсор на обгортку, де цей компонент використовується. Основні параметри компонента наведені у таблиці 3.3.

Компонент використовує хук useState для внутрішнього стану, контролюючи, чи потрібно відображати підказку в даний момент. Функція toggleShow викликається при події onMouseOver та onMouseOut, яка змінює стан show з true на false та навпаки.

Таблиця 3.3. - Параметри компонента Tooltip

Параметр	Опис
elementToShowOnHover	Вміст, який буде відображено у вигляді підказки
children	Вміст, який буде оточувати компонент і викликати підказку при наведенні на нього курсору
wrapperStyles	Додаткові стилі, які можуть бути передані для налаштування зовнішнього вигляду компонента
overlapParent	Параметр, який визначає, чи має відображатися підказка над батьківським елементом, якщо вона виходить за його межі

Структура компонента також включає в себе додатковий внутрішній елемент textCard, який представляє собою контейнер для вмісту, який буде відображатися у підказці. Цей контейнер також може використовувати хук useTranslateIfOverflow, що служить для обробки випадку, коли підказка виходить за межі батьківського елемента і використовує трансформацію для вирішення цього. Логіка цього хука доступна у файлі useTranslateIfOverflow.js.

Основний механізм хука полягає у визначенні правильних значень зсуву (offset) для трансформації елементу, який містить підказку. Основні концепції:

findParent: Ця функція шукає найближчого видимого батька для визначення правильних значень offset. Використовуючи цикл while, вона переходить вгору по ієрархії DOM до тих пір, поки не знайде батька з встановленими розмірами (clientWidth та clientHeight не рівні 0).

defineOffset: Ця функція визначає необхідний зсув (offset) для трансформації елементу в залежності від його розташування відносно вікна та батьківського елемента.

Для top і bottom перевіряється, чи не виходить підказка за верхній або нижній край вікна.

Для left і right перевіряється, чи не виходить підказка за лівий або правий край вікна.

Якщо виявляється, що підказка виходить за межі, обчислюються нові значення зсуву, які потім застосовуються за допомогою CSS-властивостей transform: translate().

refElement: Це значення ref, яке буде призначено DOM-елементу, до якого застосовується хук. Коли елемент змінюється або вперше має значення (не є null), викликається функція defineOffset, яка встановлює трансформацію, якщо це потрібно.

Використовуючи цей хук, можна забезпечити, що підказка буде відображатися в правильному місці в залежності від свого контексту та об'єму екрану.

3.3.3. Інші компоненти з використанням зовнішніх бібліотек

Rich Text Editor (RTE) - це інтерактивний текстовий редактор, який дозволяє користувачам форматувати текст за допомогою різноманітних стилів, розміщувати зображення, вставляти посилання та використовувати інші елементи оформлення. Він часто використовується на веб-сайтах, включаючи інтернет-магазини, для того, щоб користувачі могли легко вводити та форматувати текст.

Інтернет-магазини можуть використовувати RTE для опису товарів, створення блогів, редагування вмісту на сторінках та інших завдань, пов'язаних з редагуванням тексту. RTE забезпечує зручний інтерфейс для роботи з текстом, подібний до текстових редакторів, що дозволяє користувачам легко вносити зміни та додавати вміст.

Реалізація Rich Text Editor (RTE) може бути викликом через його складність та необхідність обробки як візуальної, так і логічної частини. Для

створення такого компонента вам необхідно враховувати кілька аспектів, які ускладнюють процес розробки.

По-перше, візуальна складова RTE включає в себе багато різноманітних елементів: від текстових полів та кнопок до інтерфейсу для форматування, вставки медіа-файлів та створення посилань. Стилізація цих елементів може бути витратною з точки зору часу через потребу в детальному контролі над виглядом кожного компонента.

По-друге, логіка RTE повинна бути здатна обробляти введені дані, перетворюючи їх у внутрішній стан RTE, і на виході забезпечувати HTML-вміст. Це вимагає досить складної системи обробки подій, валідації та перетворення даних.

На рівні кодування, використання бібліотек, таких як React або Material UI або Bootstrap, може полегшити розробку шляхом надання готових компонентів та функціоналу для створення інтерфейсу. Проте, розробка складного компонента RTE залишається завданням, яке вимагає великої уваги до деталей та тестування для забезпечення правильної роботи.

Тому варто використати зовнішні бібліотеки. Quill та Draft.js - це дві популярні бібліотеки для реалізації редакторів розширеного тексту (RTE) в веб-додатках. Кожна з них має свої переваги та особливості, які можуть вплинути на вибір для конкретного проекту.

Quill - це легкий та простий у використанні RTE, який надає гнучкий інтерфейс для користувачів. Він підтримує відмінне форматування тексту та має чистий дизайн. Одна з ключових переваг Quill - це висока ступінь налаштованості, дозволяючи зручно вбудовувати його в різні проекти. Quill також підтримує вставку зображень та відео, що робить його дуже функціональним [23].

Draft.js - це більш низькорівнева бібліотека, розроблена Facebook. Вона пропонує широкі можливості для контролю над кожним аспектом RTE. Draft.js використовує неімутабельний підхід до керування станом, що сприяє уникненню багатьох проблем, пов'язаних із змінами стану. Ця бібліотека

забезпечує розширені можливості для обробки блоків тексту, навігації та редагування [24].

Обираючи між Quill та Draft.js, важливо враховувати рівень гнучкості, який вам потрібен. Quill може бути кращим вибором для швидкого впровадження з більшим акцентом на простоту використання та зовнішній вигляд. З іншого боку, Draft.js надає великий рівень контролю для проектів, де необхідно здійснити глибокі налаштування та розширення.

Графіки (або "charts") – це візуальні інструменти, які використовуються для графічного відображення даних або інформації у вигляді діаграм, графіків чи інших графічних елементів. У контексті інтернет-магазинів графіки часто використовуються для наглядного представлення статистичних даних, трендів продажів, запасів товарів або інших ключових метрик, які допомагають клієнтам приймати обґрунтовані рішення.

Графіки також використовуються для відображення змін цін на товари в залежності від часу, що дозволяє клієнтам відстежувати та аналізувати динаміку цін.

Реалізовувати компоненти, що відображають нескладні графіки можна на чистому JS та Material UI.

У HTML та JavaScript використовують елемент <canvas>, який дозволяє реалізувати растрову графіку на веб-сторінці та маніпулювати зображеннями за допомогою JavaScript.

Спочатку потрібно створити елемент <canvas> у HTML-коді, визначити його розміри та надати ідентифікатор для звертання через JavaScript.

Далі, скрипт на JavaScript буде використовувати контекст <canvas> для малювання. Дістаємо посилання на елемент <canvas> та його контекст за допомогою методу getContext. Це може бути 2D-контекст для простих операцій малювання або 3D-контекст для складніших візуалізацій.

Після отримання контексту можна використовувати різні методи для малювання ліній, прямокутників, кругів та інших фігур. Також можна використовувати методи для зміни кольорів, товщин ліній, заповнення фігур,

тощо. Важливо враховувати, що операції малювання відбуваються в порядку їх виклику. Це означає, що, наприклад, якщо спочатку малюємо фон, а потім на цьому фоні інші елементи, то фон буде відображений знизу [25].

Також, для анімацій або інтерактивності, можна використовувати цикли (наприклад, requestAnimationFrame для анімацій) та обробники подій для відстеження взаємодії користувача з графікою. В таблиці 3.4. подано список готових компонентів графіків з Material UI [26].

Таблиця 3.4. - Готові компоненти графіки з Material UI

Графік	Опис
Bars	Стовпчасті діаграми виражають кількість через довжину стовпчика, використовуючи загальну базову лінію
Lines	Лінійні діаграми можуть виражати такі властивості даних, як ієрархія, виділення та порівняння
Pie	Кругові діаграми відображають частини цілого за допомогою дуг або кутів у межах кола
Scatter	Точкові діаграми виражають зв’язок між двома змінними за допомогою точок на поверхні
Sparkline	Спарлайн-діаграми можуть надати огляд тенденцій даних. Вони представляють загальну форму варіації у спрощений спосіб
Custom	Бібліотек надає можливість модифікувати існуючі компоненти, поєднувати їх та створювати інші

Таким чином ми можемо закрити потребу використання певних графіків. Але при роботі з canvas потрібно витрачати багато часу на розробку нового графіку. З цього боку Material UI значно спрощую процес розробки. Проте Material UI містить обмежений набір графіків. Тому якщо на етапі аналізу та дизайну визначається значна потреба в графіках, варто використати готові бібліотеки.

Chart.js і Highcharts.js є двома популярними бібліотеками для створення інтерактивних графіків та діаграм на веб-сайтах. Обидві бібліотеки пропонують широкий спектр можливостей, але вони мають кілька ключових відмінностей.

Chart.js - це легка та проста у використанні бібліотека, яка розроблена для швидкого створення простих графіків. Вона базується на HTML5-канвасі і підтримує кілька базових типів графіків, таких як лінійні, кільцеві, стовпчасті тощо. Це може бути ідеальним вибором для невеликих проєктів або випадків, коли простота важлива [27].

Highcharts.js надає більше функціоналу та можливостей. Вона є потужнішою та багатофункціональною, підтримуючи широкий спектр типів графіків, включаючи тривимірні графіки та графіки великого обсягу даних. Highcharts.js також має розширену підтримку для взаємодії з графіками та налаштувань [28].

Проте якщо продукт є графіко орієнтований варто розглянути потужніші інструменти, такі як D3. D3.js (Data-Driven Documents) - це JavaScript-бібліотека для маніпулювання даними та створення інтерактивних візуалізацій на веб-сторінках. Основна ідея D3 полягає в тому, щоб пов'язати дані з DOM-елементами та використовувати HTML, SVG та CSS для представлення цих даних у вигляді графіків, діаграм, карт і т. д. D3 надає широкі можливості для створення складних та креативних візуалізацій, але водночас вимагає від розробника глибокого розуміння того, як працюють веб-технології [29].

C3.js - це обгортка (wrapper) навколо D3.js, яка спрощує створення графіків та діаграм. C3 використовує D3 внутрішньо, але надає більш

високорівневий API, що полегшує роботу з графіками. C3 включає в себе різні типи графіків, такі як лінійні, стовпчасті, кругові та інші, і дозволяє швидко налаштовувати зовнішній вигляд графіків за допомогою конфігураційних параметрів [30].

Також існують інші обгортки навколо D3, такі як NVD3 та Chartist, які надають високорівневий API для візуалізацій та діаграм, використовуючи потужні можливості D3 без необхідності в глибокому розумінні його внутрішнього API.

Обгортки навколо D3 можуть бути корисні для розробників, які хочуть швидко створювати привабливі та інтерактивні візуалізації без необхідності вивчення всіх деталей D3.js. Однак, при потребі у складних та специфічних візуалізаціях може варто прямо використовувати D3 для більшої гнучкості та контролю.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Обґрунтування можливих чинників травмонебезпечних ситуацій

Процеси виконанням різноманітних завдань, що пов'язані з програмуванням, тестуванням та іншими комп'ютерними роботами можуть супроводжуватися травмонебезпечними ситуаціями, які виникають внаслідок недбалості, непередбачуваних обставин або несправностей обладнання, наприклад:

- Перегрів комп'ютера: Довготривале використання може спричинити перегрів комп'ютерної техніки, що може призвести до випадків виходу з ладу апарату;
- Втрата даних: Аварійні випадки та програмні помилки можуть призвести до втрати важливих даних;
- Хакерські атаки: Комп'ютерні системи можуть бути піддані хакерським атакам, що може призвести до втрати конфіденційної інформації або порушення безпеки даних;
- Коротке замикання: Неправильне використання електроживлення може спричинити коротке замикання та можливість виникнення пожеж.

Ці травмонебезпечні ситуації вимагають детального аналізу та розробки ефективних заходів профілактики для забезпечення безпеки та здоров'я користувачів комп'ютерної техніки.

Перш за все, важливо визначити структуру робочого місця, включаючи правильне розташування монітора, клавіатури та інших пристроїв, щоб уникнути перенапруження м'язів та неправильного положення тіла.

На функціональному рівні необхідно забезпечити належну організацію робочого часу та регулярні перерви для відпочинку, а також розробити ефективні процедури обробки інформації та управління документами. До цього входить і використання спеціального програмного забезпечення для запобігання втомленості та стресових ситуацій.

З точки зору безпеки необхідно вдосконалити аспекти ергономіки робочого простору, забезпечити адекватне освітлення та вентиляцію, а також впровадити заходи для профілактики синдрому карпального каналу та інших захворювань, пов'язаних із роботою за комп'ютером. Крім того, слід надавати працівникам необхідну інформацію та навчати їх правилам безпеки при роботі з електронною технікою.

Перш за все, важливо організувати раціональне розташування робочого місця, забезпечуючи правильне освітлення та вентиляцію. Рекомендується уникати відблисків на моніторі, регулювати яскравість та контрастність екрану для зменшення втоми очей.

Додатково, слід встановити ергономічні меблі та обладнання, забезпечуючи правильну позу під час роботи. Регулювання висоти та кута налаштувань стільця та монітора може сприяти уникненню напруги в шиях і спині. Важливо навчати працівників правильним прийомам роботи з клавіатурою та мишею, а також проводити регулярні паузи для розслаблення м'язів і очей.

Технічні заходи включають в себе використання спеціального антивірусного програмного забезпечення для захисту від можливих загроз безпеки в Інтернеті. Регулярне оновлення програм та операційних систем також є ключовим елементом забезпечення кібербезпеки. Комп'ютери та периферійні пристрої повинні бути утримувані в належному технічному стані, а також забезпечувати можливість резервного копіювання важливих даних для запобігання можливим втратам інформації. Загальна свідомість працівників про безпекові правила та процедури також грає важливу роль у забезпеченні ефективної охорони праці при використанні ПК.

4.2. Умови і обставини виникнення небезпечних ситуацій та їх наслідки

Для запобігання виникненню травмонебезпечних ситуацій при розробці користувацьких інтерфейсів інтернет-магазину необхідно дотримуватися наступних вимог:

Організація робочого місця:

- робоче місце повинно бути обладнане відповідно до вимог безпеки праці;
- робоче місце повинно бути добре освітленим;
- робоче місце повинно бути вільним від сторонніх предметів.

Забезпечення безпеки при роботі з обладнанням:

- обладнання повинно бути справним і відповідати вимогам безпеки праці;
- перед початком роботи з обладнанням необхідно ознайомитися з інструкцією з експлуатації;
- під час роботи з обладнанням необхідно дотримуватися правил безпеки праці.

Забезпечення безпеки при роботі з електричним обладнанням:

- електричне обладнання повинно бути заземлено;
- електричне обладнання повинно відповідати вимогам безпеки праці;
- під час роботи з електричним обладнанням необхідно дотримуватися правил безпеки праці.

Забезпечення пожежної безпеки:

- на робочому місці повинні бути встановлені засоби пожежогасіння;
- горючі матеріали повинні зберігатися в спеціально відведених місцях.
- під час роботи слід дотримуватися правил пожежної безпеки;

Психологічний комфорт:

- необхідно забезпечити розробникам умови для відпочинку та відновлення сил;

необхідно створити в колективі атмосферу взаєморозуміння та підтримки.

У таблиці 4.1. розглянуто процес формування травмонебезпечних ситуацій при роботі з електрообладнанням.

Таблиця 4.1. - Процес формування травмонебезпечних ситуацій

Вид роботи, виробничий підрозділ	Виробнича небезпека			Можливі наслідки	Заходи запобігання
	Небезпечна умова НУ	Небезпечна дія НД	Небезпечна ситуація НС		
Робота з електро-обладнанням, використання розеток Блок-схема	$НУ_1$ – обладнання зламалось; $НУ_2$ – несправна електро-мережа $НУ_1 \rightarrow$ $НУ_2 \rightarrow$	НД – перебування у небезпечній зоні працівника НД $\searrow$ $\rightarrow$	$НС_1$ – пошкодження обладнання; $НС_2$ – дотикання до корпуса; $НС_3$ – ураження струмом $НС_1 \rightarrow НС_2 \rightarrow НС_3$	Т – Травма $\rightarrow$ Т	Обладнати засобами захисту, стабілізатор струму та відсікач. Перевіряти справність заземлення і електро-обладнання

4.3. Безпека в надзвичайних ситуаціях

Актуальність проблеми безпеки в інформаційних технологіях постійно зростає, особливо в контексті збільшення кількості кібератак та інших надзвичайних ситуацій цифрового характеру. Сучасний світ стає все більш залежним від інформаційних технологій, а це вносить свої виклики та загрози, які необхідно аналізувати та протидіяти.

Безпека в надзвичайних ситуаціях в сучасному світі інформаційних технологій стала ключовим аспектом забезпечення стабільності та

функціонування різноманітних сфер життя. Однією з основних загроз є кібератаки, що стають все більш вибагливими та складними. Зловмисники активно використовують різні методи, включаючи вразливості програмного забезпечення, соціальний інжиніринг та атаки на мережеві інфраструктури.

Кібератаки можуть призвести до серйозних наслідків, таких як втрата конфіденційності даних, порушення цілісності інформації та навіть припинення роботи критичних систем. Важливо розглядати ці загрози як надзвичайні ситуації, що вимагають виваженої стратегії та швидкого реагування.

Додатково, інші надзвичайні ситуації в сфері інформаційних технологій можуть включати природні катастрофи, такі як землетруси або повені, що можуть призвести до втрати доступу до інфраструктури та даних. Також важливо враховувати людський фактор, який може призвести до витоку конфіденційної інформації через помилки персоналу або необережність.

У контексті безпеки в надзвичайних ситуаціях в інформаційних технологіях, розробка та впровадження ефективних стратегій захисту, регулярне навчання персоналу та постійне вдосконалення систем безпеки є критичними елементами для забезпечення сталості та реакції на широкий спектр надзвичайних ситуацій в світі інформаційних технологій.

РОЗДІЛ 5
ВИЗНАЧЕННЯ ЕФЕКТИВНОСТІ

5.1. Значення та типи ефективності

Ефективність є ключовим показником успіху в будь-якій сфері діяльності та визначає, наскільки ефективно використовуються ресурси для досягнення поставлених цілей. В контексті розробки інтернет-магазину, ефективність означає досягнення максимального результату з мінімальними зусиллями та ресурсами.

Експерти визначають наступні два типи ефективності:

- Технічна ефективність, яка включає в себе швидкодію, надійність, масштабованість та безпеку програмного забезпечення. Технічна ефективність гарантує оптимальне використання технологій та інфраструктури;
- Бізнес-ефективність, яка вимірює досягнення бізнес-цілей та ефективність в управлінні ресурсами. Це включає в себе фінансовий успіх, відношення вартості до користі, стратегічне планування та здатність ініціювати зміни для вдосконалення бізнес-процесів. Бізнес-ефективність визначає, наскільки інтернет-магазин сприяє досягненню цілей компанії та його конкурентоспроможність на ринку.

5.2. Розрахунок ефективності

Використання користувацьких інтерфейсів при розробці інтернет-магазину має низку переваг, які сприяють підвищенню ефективності та якості проекту. Професійний підхід до розробки інтерфейсів означає не лише технічну експертність, але і глибоке розуміння потреб та вимог користувачів, що є ключовим для успішного, привабливого, прибуткового інтернет-магазину.

Професійний підхід до розробки користувацьких інтерфейсів передбачає активну співпрацю з замовником на етапах планування та дизайну. Це включає не лише технічну, але і стратегічну обізнаність, яка дозволяє збирати та аналізувати вимоги замовника, а також прогнозувати його потреби в майбутньому. Тривала робота в цьому напрямку допомагає створити інтерфейс, який не лише відповідає технічним вимогам, але й інтегрує в себе стратегічний вектор розвитку бізнесу замовника. Такий підхід є ключовим для побудови продукту, який ефективно взаємодіє зі своєю цільовою аудиторією та відповідає її потребам.

Пошук оптимальних рішень є важливим етапом професійної розробки інтерфейсів для інтернет-магазину. Це включає в себе аналіз конкурентного середовища, вибір оптимальних технологій та визначення ключових функціональних можливостей. Такий детальний аналіз допомагає уникнути невдач та непотрібних витрат на реалізацію функціоналу, який буде перероблятись. Поєднання стратегічного підходу та експертної технічної імплементації визначає оптимальний шлях до успішної розробки інтернет-магазину, який відповідає не лише сучасним технічним вимогам, але й вигідно вирізняється серед конкурентів на ринку.

Чим менше змін вноситься під час реалізації проекту, тим більш стабільним і прогнозованим стає процес розробки. Детальний план інтерфейсу, сформований на етапі планування та дизайну, дозволяє точно визначити обсяг робіт та розподіл завдань між командою розробників. Це полегшує створення чіткого графіка, якого можна легко дотримуватись, оскільки відомо, що основні етапи проекту вже ретельно продумані та не піддаються суттєвим змінам.

Стабільний графік реалізації сприяє ефективному управлінню ресурсами та забезпечує оптимальне використання команди. Завдяки чіткому розподілу завдань та відсутності радикальних змін, можна планувати сталу кількість працівників на проекті, що сприяє стабільності команди та забезпечує її ефективну співпрацю протягом усього проекту.

Готовий дизайн перед початком реалізації інтернет-магазину дозволяє розробникам зосередитися на технічній імплементації без витрат на вирішення дизайнерських питань. Це сприяє прискоренню термінів розробки, зменшенню ризиків затримок і полегшенню управління проектом. Готовий дизайн також дозволяє забезпечити консистентність та професійний вигляд інтерфейсу, що важливо для задоволення очікувань користувачів.

Чітко визначені вимоги та функціональні можливості на етапі планування роблять вибір технологій більш обґрунтованим та ефективним. Розробники можуть вибрати ті технології, які найкраще відповідають потребам проекту, не витрачаючи час на експерименти або зміни під час реалізації. Це сприяє покращенню продуктивності, забезпечує використання оптимальних інструментів та технологій для досягнення поставлених цілей.

Таким чином створення користувацького інтерфейсу, його ретельний аналіз та підбір відповідних інструментів реалізації дозволяє випустити якісний продукт, з мінімальними витратами на переробки та інші операції, пришвидшує терміни реалізації, відповідно забирає можливість затримок, економить гроші замовника і залишає позитивний досвід роботи.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було вивчено призначення користувацьких інтерфейсів, їх використання в інтернет-магазинах. Вивчено історію розвитку користувацьких інтерфейсів, ключових характеристик та потреб. Проведено аналіз користувацьких інтерфейсів на прикладі популярних інтернет-магазинів Allo та Rozetka. Вивчено завдання створення інтернет-магазину як продукту, що базується на користувацьких інтерфейсах.

Обрано велосипедний магазин як предметну область. Досліджено її, включаючи огляд конкурентів, їхні технології та функціонал. Проаналізовано та задокументовано вимоги, функціонал.

Змодельовано список сторінок інтернет-магазину, включаючи заголовок (Header), підвал (Footer) та бічну панель (Sidebar) як складові кожної сторінки. Серед сторінок є головна сторінка, каталог товарів, сторінка окремого товару, кошик та кроки процесу замовлення.

Створено перелік компонентів, необхідних для реалізації користувацьких інтерфейсів цих сторінок, визначено інструменти для їх реалізації, зокрема: HTML, CSS, JS; React; Bootstrap; Material UI. На прикладі компонента кнопки проаналізовано переваги та недоліки його створення з використанням вищевказаних технологій.

Проведено докладний огляд технологій, в тому числі розглянуто можливість роботи в React з чистим JS, на прикладі реалізації хука useTranslateIfOverflow для компонента Tooltip. Також обґрунтовано потребу в застосуванні інших спеціалізованих бібліотек для реалізації конкретних компонентів, таких як Rich Text Editor, Charts.

У результаті можна зазначити, що набір HTML, CSS та JS надає можливість реалізувати практично будь-який функціонал, адже інші бібліотеки виступають лише обгортками навколо цих технологій. Проте цей підхід вимагає значної кількості часу на розробку, тестування та інші процеси, які вже покриті цими обгортками. Bootstrap спрощує стилізацію та надає певну кількість готових компонентів, але все одно багато елементів доводиться

створювати вручну. Серед недоліків Bootstrap можна відзначити обмежену кастомізованість і збільшення розміру сайту, що впливає на тривалість його завантаження.

React є зручною, малою UI бібліотекою для створення компонентів, але порівняно з фреймворками такими як Angular вимагає додаткових бібліотек для повноцінної роботи. Проте всі компоненти потрібно створювати вручну, хоч це і зручніше і швидше у порівнянні з попередніми варіантами.

Material UI пропонує готові React компоненти. Це дійсно зручно і швидко. Їх можна легко стилізувати з допомогою Theme. Однак ця бібліотека вимагає використання React, що в результаті приводить до значного збільшення розміру сайту, зниження швидкості завантаження тощо. Хоч і виглядає що вона ідеальна, адже зібрала найкращі характеристики вищенаведених бібліотек, а також опрацювала більшість проблем, проте це не так, адже є компоненти які вона не має і реалізовувати їх вручну недоцільно. Наприклад для реалізації компонентів Rich Text Editor та Charts варто використати готові рішення.

Не можна однозначно сказати, який набір технологій є оптимальним для роботи з користувацькими інтерфейсами. Вибір технологій повинен відбуватися на етапі проектування архітектури та дизайну SDLC, враховуючи потреби проекту, доступні фінансові ресурси та обмеження часу. NPM дозволяє легко та швидко підключати всі необхідні бібліотеки, що полегшує управління залежностями в процесі розробки.

Також варто додати, що дизайни постійно розвиваються і для їх імплементації потрібні нові підходи. Паралельно з цим відбувається оновлення версій існуючих технологій, також поява нових. І це підкреслює потребу аналізу як потреб продукту, так і існуючих інструментів щоразу при створенні нового продукту. Загалом на даний момент використання React + Material UI має забезпечити більшість потреб.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Pulli Petri; Antoniac Peter. User interface. U.S. Patent No 6,771,294, 2004.
2. Куклінова Тетяна Вікторівна. Сучасні тенденції Інтернет-торгівлі в Україні. Вісник соціально-економічних досліджень, 2018, 1: 95-102.
3. Гончар В. М.; Римар П. В. Історія розвитку технологій для створення веб-сторінок. Прикладні аспекти сучасних міждисциплінарних досліджень, 2021, С. 60-62.
4. Грабовський Є. М.; Грабовський Е. Н. Проектування інтелектуального користувацького інтерфейсу систем підтримки електронного навчання. 2018. 11 С.
5. Супруненко О. О. Аналіз вимог до програмного забезпечення. 2012. 24С.
6. Ткаченко Олександр Андрійович; Ткаченко Ольга Іванівна; Делант, Олександр Олександрович. СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ЩОДО ВИБОРУ ВЕЛОСИПЕДУ. ITSynergy, 2023, 1: С. 60-73.
7. Ranjan Alok, Abhilasha Sinha and Ranjit Battewad. JavaScript for modern web development: building a web application using HTML, CSS, and JavaScript. BPB Publications, 2020.
8. Lazuardy Mochammad Fariz Syah, and Dyah Anggraini. “Modern front end web architectures with react. js and next. js.” Research Journal of Advanced Engineering and Science 7.1 (2022): С.132-141.
9. Gaikwad S. Shahu and Adkar. “A review paper on bootstrap framework.” IRE Journals 2.10 (2019): С. 349-351.
10. Guo Lin. "Best UI Experience: Material Design in Action", 2022. С. 519-574.
11. AlOmar, Eman Abdullah, et al. “How do developers refactor code to improve code reusability?.” Reuse in Emerging Software Engineering Practices: 19th International Conference on Software and Systems Reuse, ICSR 2020, Hammamet, Tunisia, December 2–4, 2020, Proceedings 19. Springer International Publishing, 2020.
12. Arora G. Kumar. “SOLID Principles Succinctly.” (2017).

13. Candela Ivan, et al. “Using cohesion and coupling for software remodularization: Is it enough?.” ACM Transactions on Software Engineering and Methodology (TOSEM) 25.3 (2016): C. 1-28.
14. Lerner Benjamin S. Designing for Extensibility and Planning for Conflict: Experiments in Web-Browser Design. University of Washington, 2011.
15. Anjorin Anthony, et al. “On the development of consistent user interfaces.” Companion Proceedings of the 2nd International Conference on the Art, Science, and Engineering of Programming. 2018.
16. The Button element [Електронний ресурс]. Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/button
17. ARIA: button role [Електронний ресурс]. Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Roles/button_role
18. Jquery. Write less, do more. [Електронний ресурс]. Режим доступу до ресурсу: https://jquery.com/
19. Buttons in bootstrap. [Електронний ресурс]. Режим доступу до ресурсу: https://getbootstrap.com/docs/5.2/components/buttons/
20. React. The library for web and native user interfaces. [Електронний ресурс]. Режим доступу до ресурсу: https://react.dev/
21. Typescript. [Електронний ресурс]. Режим доступу до ресурсу: https://www.typescriptlang.org/
22. Material UI. Move faster with intuitive React UI tools [Електронний ресурс]. Режим доступу до ресурсу: https://mui.com/
23. Your powerful rich text editor. [Електронний ресурс]. Режим доступу до ресурсу: https://quilljs.com/
24. Rich Text Editor Framework for React [Електронний ресурс]. Режим доступу до ресурсу: https://draftjs.org/
25. Canvas [Електронний ресурс]. Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/HTML/Element/canvas

26. Material UI charts [Електронний ресурс]. Режим доступу до ресурсу: https://mui.com/x/react-charts
27. Chart.js [Електронний ресурс]. Режим доступу до ресурсу: https://www.chartjs.org/
28. Highcharts [Електронний ресурс]. Режим доступу до ресурсу: https://www.highcharts.com/
29. D3 [Електронний ресурс]. Режим доступу до ресурсу: https://d3js.org/
30. C3 [Електронний ресурс]. Режим доступу до ресурсу: https://c3js.org/
31. Пістун І. П., Березовецький А. П., Тимочко В.О., Городецький І. М. Охорона праці (гігієна праці та виробнича санітарія): навч. посібн. / за ред. І.П. Пістуна. Ч. І. Львів: Тріада плюс, 2017. 620 с.
32. Пістун І. П., Тимочко В.О., Городецький І. М.. Березовецький А. П. Охорона праці (гігієна праці та виробнича санітарія): навч. посібн. / за ред. І.П. Пістуна. Ч. ІІ. Львів: Тріада плюс, 2011. 224 с.

ДОДАТКИ

Додаток А

Текст вихідного коду

buttonComponent.js

```
import ‘./buttonStyles.css’; // Підключення CSS стилів
const addButton = (type, size, onClick, isEnabled, children, parent) => {
const button = document.createElement(“button”);
// Додаємо класи стилів на основі переданого типу
button.classList.add(type);

// Додаємо класи стилів на основі переданого розміру
button.classList.add(size);

// Додаємо текст чи іконку до кнопки
if (children) {
button.innerHTML = children;
  }

// Додаємо обробник події onClick
if (onClick && typeof onClick === ‘function’) {
button.addEventListener(‘click’, onClick);
  }

// Встановлюємо властивість “включено” або “виключено”
button.disabled = !isEnabled;
// Додаємо кнопку до вказаного батьківського елемента
parent.appendChild(button);
};

export default addButton;
```

buttonStyles.css

```
/* Стилі для різних типів кнопок (наприклад, primary, secondary, etc.) */
.primary {
background-color: #3498db;
color: #fff;
}

.secondary {
background-color: #e74c3c;
color: #fff;
}

/* Стилі для різних розмірів кнопок (наприклад, small, large, etc.) */
.small {
font-size: 12px;
```

```
}

.large {
font-size: 18px;
}
```

main.js

```
import addButton from ‘./buttonComponent’;
const parentElement = document.getElementById(‘parentContainer’);
// Приклад використання:
addButton(‘primary’, ‘large’, () => alert(‘Button Clicked’), true, ‘Click me!’, parentElement);
```

buttonComponent.js

```
import $ from ‘jquery’; // Підключення jQuery бібліотеки
const addButton = (type, size, onClick, isEnabled, children, parent) => {
const button = $(‘<button></button>’);
// Додаємо класи стилів на основі переданого типу та розміру
button.addClass(type).addClass(size);
// Додаємо текст чи іконку до кнопки
if (children) {
button.html(children);
  }

// Додаємо обробник події onClick
if (onClick && typeof onClick === ‘function’) {
button.on(‘click’, onClick);
  }

// Встановлюємо властивість “включено” або “виключено”
button.prop(‘disabled’, !isEnabled);
// Додаємо кнопку до вказаного батьківського елемента
$(parent).append(button);
};

export default addButton;
```

main.js

```
import addButton from ‘./buttonComponent’;
import ‘buttonStyles.css’; // Підключення CSS стилів

const parentElement = $(‘#parentContainer’);
// Приклад використання:
addButton(‘primary’, ‘large’, () => alert(‘Button Clicked’), true, ‘Click me!’, parentElement);
```

Button.jsx

```
import React from 'react';
import PropTypes from 'prop-types';
import cx from 'classnames';

import styles from './Button.scss';

const Button = ({
  type,
  size,
  onClick,
  disabled,
  className,
  children
}) => {
  const btnClassName = cx(
    styles[type],
    styles[size],
    className
  );

  return (
    <button
      className={btnClassName}
      disabled={disabled}
      onClick={onClick}
    >
      {children}
    </button>
  );
};

Button.propTypes = {
  type: PropTypes.string,
  size: PropTypes.string,
  onClick: PropTypes.func,
  disabled: PropTypes.bool,
  className: PropTypes.string,
  children: PropTypes.oneOfType([PropTypes.string, PropTypes.object,
PropTypes.node])
};

Button.defaultProps = {
  type: 'defaultType',
  size: 'small',
  onClick: () => {},
```

```
  disabled: false,
  className: null,
  children: null
};

export default Button;
```

Styled Button with MUI Theme

```
import React from 'react';
import Button from '@mui/material/Button';
import { styled } from '@mui/system';

const StyledButton = styled(Button)(({ theme }) => ({
  // Ваші власні стилі тут
  color: theme.palette.primary.main, // Приклад використання кольору з теми
  fontSize: '1rem',
  '&:hover': {
    backgroundColor: theme.palette.primary.dark,
  },
}));

const MyStyledButton = ({ children, ...props }) => {
  return <StyledButton {...props}>{children}</StyledButton>;
};

export default MyStyledButton;
```

Tooltip.jsx

```
import React, { useState } from 'react';
import PropTypes from 'prop-types';
import cx from 'classnames';
import useTranslateIfOverflow from '../utils/hooks/useTranslateIfOverflow';

import styles from './Tooltip.scss';

const Tooltip = ({
  elementToShowOnHover,
  children,
  wrapperStyles,
  overlapParent
}) => {
  const [show, setShow] = useState(false);
  const toggleShow = () => {
```

```
    setShow(currentState => !currentState);
  };

  const textCard = (
    <div
      className={styles.textCardWrapper}
      ref={useTranslateIfOverflow(overlapParent)}
    >
      {elementToShowOnHover}
    </div>
  );

  return (
    <div
      className={cx(
        styles.tooltip,
        wrapperStyles,
      )}
      onMouseOver={toggleShow}
      onMouseOut={toggleShow}
    >
      {children}
      {show && textCard}
    </div>
  );
};


Tooltip.propTypes = {
  elementToShowOnHover: PropTypes.node,
  children: PropTypes.node,
  wrapperStyles: PropTypes.string,
  overlapParent: PropTypes.bool
};

Tooltip.defaultProps = {
  elementToShowOnHover: false,
  children: false,
  wrapperStyles: '',
  overlapParent: false
};


export default Tooltip;
```

useTranslateIfOverflow.js

```
import { useCallback } from 'react';

const HEADER_CONTAINER_HEIGHT = 64;
const SIDEBAR_WIDTH = 65;
const MIN_OFFSET_FROM_PARENT_TOP = 7;
const MIN_OFFSET_FROM_PARENT_LEFT = 7;
const MIN_LEFT_OFFSET = SIDEBAR_WIDTH + MIN_OFFSET_FROM_PARENT_LEFT;
const MIN_TOP_OFFSET = HEADER_CONTAINER_HEIGHT + MIN_OFFSET_FROM_PARENT_TOP;
const MIN_RIGHT_OFFSET = 18;
const MIN_BOTTOM_OFFSET = 5;

const useTranslateIfOverflow = (overlapParent) => {
  const findParent = (element) => {
    let parent = element.parentNode;
    while (parent && parent.clientWidth === 0 && parent.clientHeight === 0 && parent) {
      parent = parent.parentNode;
    }
    return parent;
  };

  const defineOffset = (element) => {
    if (element) {
      const {
        top, right, bottom, left
      } = element.getBoundingClientRect();
      const bottomViewportLine = window.innerHeight;
      const rightViewportLine = window.innerWidth;

      const parent = findParent(element);
      const posParent = !overlapParent && !!parent && parent.getBoundingClientRect();

      // checking topOffset
      let newTopOffset = 0;
      if (top < MIN_TOP_OFFSET) {
        if (overlapParent || !parent || (left > posParent.right || right < posParent.left)) {
          newTopOffset = MIN_TOP_OFFSET - top;
        } else {
          newTopOffset = posParent.bottom - top + MIN_OFFSET_FROM_PARENT_TOP;
```

```
      }
    }

    // checking bottomOffset
    let newBottomOffset = 0;
    if (newTopOffset === 0 && bottom > bottomViewportLine -
MIN_BOTTOM_OFFSET) {
      if (overlapParent || !parent || (left > posParent.right || right < posParent.left)) {
        newBottomOffset = bottomViewportLine - bottom -
MIN_BOTTOM_OFFSET;
      } else {
        newBottomOffset = posParent.top - bottom -
MIN_OFFSET_FROM_PARENT_TOP;
      }
    }

    // checking leftOffset
    let newLeftOffset = 0;
    if (left < MIN_LEFT_OFFSET) {
      if (overlapParent || !parent || (top > posParent.bottom || bottom < posParent.top))
{
        newLeftOffset = MIN_LEFT_OFFSET - left;
      } else {
        // eslint-disable-next-line no-mixed-operators
        newLeftOffset = posParent.right - left +
MIN_OFFSET_FROM_PARENT_LEFT;
      }
    }

    // checking rightOffset
    let newRightOffset = 0;
    if (newLeftOffset === 0 && right > rightViewportLine -
MIN_RIGHT_OFFSET) {
      if (overlapParent || !parent || (top > posParent.bottom || bottom < posParent.top))
{
        newRightOffset = rightViewportLine - right - MIN_RIGHT_OFFSET;
      } else {
        newRightOffset = posParent.left - right -
MIN_OFFSET_FROM_PARENT_LEFT;
      }
    }

    if (newBottomOffset !== 0 || newRightOffset !== 0 || newTopOffset !== 0 ||
newLeftOffset) {
      const xOffset = newLeftOffset === 0 ? newRightOffset : newLeftOffset;
```

```
      const yOffset = newTopOffset === 0 ? newBottomOffset : newTopOffset;
      // eslint-disable-next-line no-param-reassign
      element.style.transform = `translate(${xOffset}px, ${yOffset}px)`;
    }
  }
 };

 const refElement = useCallback((node) => {
  if (node !== null) {
   defineOffset(node);
  }
 }, []);

 return refElement;
};

export default useTranslateIfOverflow;
```