МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Інформаційне та програмне забезпечення захищеного веб-сайту»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Ободяк В.К.**

Студента групи КБ-71-9 **Ященко Б.В.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2021 р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи КБ-71-9 спеціальності "Кібербезпека" денної форми навчання Ященко Богдана Володимировича.

Тема: " Інформаційне та програмне забезпечення захищеного веб-сайту"

Затверджена наказом по СумДУ

№_____ от _____2021 р.

Зміст пояснювальної записки: 1) Аналіз захисних систем веб-сайту; 2)характеристика методів та інструментарію дослідження інформаційної безпеки; 3)порівняльний аналіз систем керування вмістом.

Дата видачі завдання "_____"_____2021 р.

Керівник випускної роботи _____ Ободяк В.К.

Завдання приняв до виконання _____ Ященко Б.В

РЕФЕРАТ

Записка: 62 стор., 22 рис., 2 табл., 16 джерел.

Об’єкт дослідження — процес забезпечення захисту веб-сайту.

Мета роботи — оцінювання усі системи захисту веб-сайту так часткова реалізація системи які таким критеріям як:

- простота використання та реалізації;
- рівень безпеки інформації;
- рівень безпеки коду ресурсу.

Методи дослідження — емпіричний науковий метод.

Результати — розроблений проект з реалізованими системи захисту інформації веб-сайту

МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ, REACT, ІНФОРМАЦІЙНА БЕЗПЕКА, ТИПИ АТАК, ВЕБ-РЕСУРС, ЗАХИСТ, АНАЛІЗ, МОДЕЛЬ БЕЗПЕКИ, БЕЗПЕКА.

ЗМІСТ

ВСТУП

Internet – це глобальна комп’ютерна мережа яка охоплює увесь світ. Все більше людей проводять свій вільний час в онлайні. На сьогоднішній день кількість інтернет-користувачів збільшилася до понад чотирьох з половиною мільярдів користувачів. Кожного місяця всесвітня мережа збільшує свій розмір на 7-10%, що в свою чергу забезпечує збільшення зв’язку різних інформаційних систем.

Якщо комп’ютерні мережі 20 років тому починали свій шлях з передачі файлів та повідомлень через електронні скрики, то на сьогоднішній час за допомогою Інтернету є можливість купувати їжу та отримувати її вдома. Це допомагає розвитку всій сфері інформаційних технологій.

Мати власний бізнес в Інтернет мережі для будь-якої компаніє є значною перевагою. Веб-додатки служать ефективною сферою реклами. Використовуючи правильно такий інструмент як сторінка веб-блогу або веб-магазин, допомагає користувачам детальніше ознайомитися з продукцією та послугами. Але веб-додатки без належної системи захисту інформації можуть допомогти в розвитку компанії конкурентів. Тому велику роль в розвитку займає правильно розроблена система захисту всіх можливих проблем веб-додатку.

Отже метою дослідження є проаналізувати всі можливі популярні методи атак веб-додатків, та довідатися усі можливі методи протидії.

1. ЗАГАЛЬНА ХАРАКТЕРИСТИКА ІНТЕРНЕТУ

1.1 Історія мережі Інтернет

В кінці 60-тих років управління перспективних проектів DARPA по завданню міністерства оборони США (Сполучені Штати Америки) приступило до реалізації інноваційного проекту по створенню першої експериментальної мережі передачі даних пакетів між приладами. Мережа мала назву APRANET, призначалась для вивчення методів захисту та надійності зв'язку між комп'ютерами різних типів. Купа методів для передачі даних за допомогою та через модеми були створенні в APRANET. Під час розробки були розроблені протоколи передачі даних у мережі – TCP/IP. Структурою TCP/IP є конструкт з багатьох комунікаційних протоколів, що дозволяють різним типам комп'ютерів спілкуватися між собою. Це дозволяє легше використання комп'ютерних мереж бо процеси передачі, пакетування, оброблення, маршрутизації та прийняття робить лише один протокол.

Експериментальний проект був успішним, тому багато організацій мали намір використовувати технологію передачі даних APRANET. Тому в 1975 році технологія передачі даних перетворилася в робочу мережу замість експериментальної. Подальший розвиток протоколу TCP/IP було створення стандартів, що дозволяють працювати по правилам.

Першим стандартом для протоколів TCP/IP був MIL STD (Military Standarts), тобто воєнний, та усіх користувачів мережі. Для полегшення упровадження протоколу була використана фірма Berkley Software Design. З подальшою модернізацією та адаптацією TCP/IP був змінений в загальнодоступний стандарт, і термін Інтернет увійшов в загальний ужиток.

В 1991 році компанія APRANET припинила своє існування. В свою чергу мережа Інтернет своє існування не припинила, вона почала об'єднувати велику кількість мереж усього світу для передачі інформації не тільки в межах держави, а й усього світу. Мережа почала розвиватися з початкової мережі в чотири комп'ютера, Інтернет почав нарощувати свою

міць до понад чотирьох з половиною мільярдів користувачів, що активно користуються нею.

Рисунок 1.1 – Діаграма кількості користувачів Інтернету.

З кожним роком кількість користувачів в глобальній мережі Інтернет зростає більш ніж на 9% кожного року.

1.2 Протоколи мережі Інтернет

Основною рисою мережі Інтернет, що відрізняє від інших мереж є протоколи TCP/IP. TCP/IP охоплює багато різних протоколів, прикладних програм та безліч мереж. Це означає, все що використовує протоколи взаємодії між комп'ютерами є Інтернетом.

TCP/IP отримав свою назву від комунікаційних протоколів, що з'єднували мережі. Першим таким протоколом є TCP (Transmission Control Protocol). Він відповідає за управління передачею даних в мережі. Другим протоколом, що дозволяє об'єднувати окремі комп'ютерні мережі в всесвітню є IP (Internet Protocol). Мережею Інтернет використовується ще велика кількість інших протоколів, але лише два безумовно є головними та найважливішими протоколами мережі Інтернет.

В мережі Інтернет, як і в інших мережах, існує сім рівнів взаємодії комп'ютерів між собою. Цими рівнями є :

1) Фізичний рівень – основною метою рівня є представити нуль або одиницю в якості сигналів.
2) Канальний рівень – метою є передача повідомлення по каналу зв’язку.
3) Мережевий рівень – метою є в створенні мереж побудованих на основі мережевих технологій різного рівня.
4) Транспортний рівень – метод відправки даних через хостів , забезпечуючи адресацію потрібного пакету даних.
5) Сеансовий рівень – метод який вирішую яка буде передача інформації між двома процесами.
6) Представлення рівень – метою рівня є представити данні в необхідній формі.
7) Прикладний рівень - метою цього рівня є забезпечення доступу до мережевих служб.

Усі ці рівні собою створюють абстракту мережеву модель OSI, що зображена на рис 1.2.

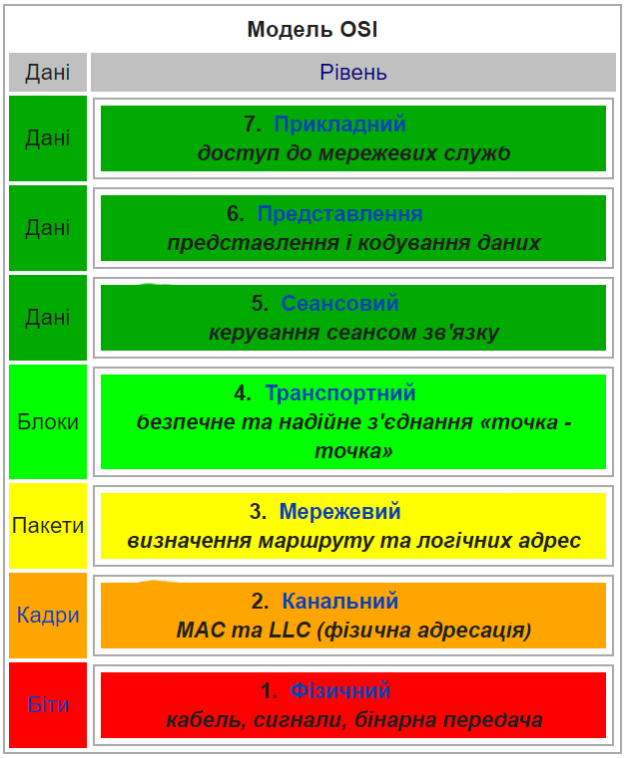

Рисунок 1.2 – Мережева модель OSI.

Вона дозволяє представити складну систему в більш простому та подрібненому на частини вигляді. Кожен рівень відповідає та обслуговує свою частину процесів.

Кожному рівню відповідають свій набір протоколів, що забезпечують його роботу.

Фізичного рівня протокол відповідає за зв'язок між комп'ютерами в мережі. Це означає, що вид зв'язку та характеристики мережі вирішується на цьому рівні. Інтернет використовує всі види зв'язку, мобільну мережу, стандартну просту дротову а також бездротові системи зв'язку.

Для зв'язку по кабелю використовують протоколи канального рівня такі як PPP (Point to Point Protocol) або HDLC (High-Level Data Link Control). Вони забезпечують доставку кадру до потрібного приладу, та підключають їх до одного мережевого сегменту.

Протоколи мережевого рівня забезпечують доставку та передачу даних між пристроями, що знаходять не в одній мережі, а декількох різних. Протоколи цього рівня такі як IP (Internet Protocol), ARP (Address Resolution Protocol) або ICMP (Internet Control Message Protocol) відповідають за маршрутизацію пакетів та їх контролювання в мережі.

Транспортний рівень є четвертим рівнем, що призначений для доставок кадрів даних або блоків даних без помилок, дублювання та авжеж без втрат. Протоколу TCP (Transmission Control Protocol) немає значення тип даних, самим призначенням рівня є передача.

Сеансовий рівень та його протоколи SOCKS (SOCKet Secure), L2F (Layer 2 Forwarding Protocol) забезпечують підтримку сеансу зв'язку будь яким чином, використовуючи безпечне підключення через проксі або тощо , що в свою чергу дозволяє безперервне використання мережі.

Протоколи рівня представлення відповідають за трансформацію протоколів. Також на цьому рівні мережевої моделі відбувається кодування та декодування даних. Усі запити програм, що отримуються від прикладного рівня , перетворюються для подальшої передачі по мережі. В свою чергу

данні, що були отримані з мережі теж перетворюються до формату, що буде зрозумілий програмі.

Прикладний рівень зі своїми протоколами забезпечую взаємодію мережі та користувача. Це дозволяю користувачеві забезпечувати користування веб-додатками в мережі, такі як електрона пошта, віддалений доступ до файлів, передачу інформації в мережі.

Протоколи прикладного рівня та рівня представлення є дуже розмитими по твердженню, адже такий протокол як HTTPS може використовуватися для перегляду інформації в мережі Інтернет через браузер, можна вважати цей протокол за протокол прикладного рівня. В той же час якщо використовувати HTTPS протокол для передачі інформації, тоді протокол буде рівня представлення.

1.3 Гіпертекстові технології Інтернету

Основним компонентом Інтернету є технологія всесвітньої павутини або WWW (World Wide Web). В свою чергу великим компонентом всесвітньої павутини є технологія гіпертексту, що дозволяю переміщуватися між текстами через посилання до інших документів, текстів, картинок.

Технологію WWW є можна вважати фундаментальною, оскільки сама суть її роботи дозволяє підтримувати практично всі типи документів існуючих в даний час, а також організовувати глобальну мережу, як систему посилань з одних ресурсів на інші. Також варто зауважити, що через стрімке зростання інтересу до даної технології раніше та через її зручну структуру, на даний момент сфера діяльності цієї технології охоплює практично всі напрямки роботи людства, що в свою чергу говорить про її актуальність та необхідність.

В 1989 році технологія гіпертексту була новою та мала великий спектр реалізації. Ідею яку описав Тім Бернерс-Лі полягала в тому, що гіпертекстова модель була застосована до інформаційної системи і ресурсам. Ця ідея була основоположною для створення систем розробки та взаємодії інформації.

Були створенні такі системи як:

- HTML – мова гіпертекстової розмітки документів,
- URL – універсальний вміст адресації ресурсів у мережі,
- HTTP - протокол обміну гіпертекстовою інформацією.

Розглянемо HTML, ця мова дозволяє створювати свої ресурси в мережі інтернет, використовуючи як свою власну інформацію так і інформацію з мережі. Основною користю гіпертексту є його зручність, за допомогою нього ви можете переглядати документ, знайти в ньому пункт, що вас цікавить, навести на нього курсором та перейти на ресурс з більш детальним висвітленням даного пункту, що дозволяє більш детально вивчити саме те, що вас цікавить. В загальному HTML це проста мова розмітки, яка дозволяє позначати елементи тексту та вставляти на них посилання, як було сказано вище, а також виділяти окремі елементи, форматувати текст та виконувати багато різних функцій, основною метою яких є перетворення маси інформації в читабельний ресурс за допомогою розміток. Також існує таке поняття, як база даних HTML документів, це частина файлової системи, в якій зберігаються файли в форматі HTML та пов’язана з ними документація, а також ресурси. Базою для створення HTML став SGML (Standard Generalised Markup Language), дана мова розмітки була стандартом до розробки HTML. Спостерігаючи за розвитком мови розмітки HTML є можливим сказати, що з часом мова розмітки перетворится в мову розробки інтерфейсів для різнобічно розвинутих систем, оскільки з кожним оновленням додаються нові елементи направлені в даному напрямку.

Іншою основною технологією, яка підтримує роботу WWW є URL, дана технологія дозволяє отримати доступ до ресурсів в мережі та робити переадресацію на них, фактично без використання цієї технології, описані вище функції по роботі з гіперпосиланнями не будуть працювати, оскільки за допомогою HTML є можливим позначити гіперпосилання, а URL в свою чергу організовує їх роботу, тому є можливим назвати HTML та URL технології основою коректної роботи технології WWW.

Та останнім розглянутим протоколом є HTTP даний протокол організовує процес обміну гіпертекстовими ресурсами, реально ж розробник працює з даним протоколом в разі роботи з зовнішніми до поточної структури ресурсами, такими як бази даних. Також важно зауважити, що існує інша форма даного протоколу HTTPS , яка реалізує все те ж саме, але з використанням захищених каналі та шифрування при необхідності, ця версія є протоколом HTTP зі значно підвищеним рівнем захисту.

Також існує технологія NCSA, основною задачою якої є розширення можливостей технології WWW за допомогою підключення зовнішніх служб,даний підхід дозволяв запустит розвиток технології за допомогою розроюки додаткового програмного забезпечення покриваючого недостатні частини функціоналу.

Технологія WWW побудована та працює на класичній моделі “Клієнт - сервер”, суть даної модулі полягає в розподіленні роботи в системі, тобто клієнтська частина відповідає за інтерфейс користувача, з нею відбуваються всі взаємодії в ході роботи системи, серверна частина працює відносно командам наданим з клієнтської, та не є видимою для користувача. Такий підхід дозволяє організувати зручну та стабільну взаємодію користувача з системою, оскільки він отримує роботу з повним функціоналом системи за допомогою зручного інтуїтивно зрозумілого інтерфейсу. Програми працюючі зі структурою клієнт-сервер є можливим поділити на два типи, це шлюзи та інші. Інші програми виконують різні задачі пов’язані з функціоналом роботи системи, в той час як програми-шлюзи займаються організацією зв’язків з зовнішніми системами, які не є частиною поточної структури.

В завершення розгляду WWW необхідно повторити, що дана технологія є абсолютно загальнодоступна та маючи доступ до мережі будь-яка людина може організувати свій сервер, а також створити веб-ресурс.

1.4 Постановка задачі

За результатми проведеного аналізу поставлені такі задачі:

1. Вибрати середовище розробки для створення веб-додатку;
2. Вибрати методи захисту веб-ресуру.
3. Вибрати метод створення веб-додатку;
4. Розробити і протестувати розроблену інформаційну систему.

2 АНАЛІЗ МОЖЛИВОСТЕЙ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

2.1 Основи інформаційної безпеки

Захист інформації та забезпечення інформаційної безпеки є безпосередньо дотримання трьох основних властивостей інформації, таких як:

- Цілісність – це інформація данні якої не були змінені при виконанні будь якої операції, захист оригінального виду інформації, захист інформації від несанкціонованих змін.
- Конфіденційністю – є захист інформації та доступу для обмеженої кількості осіб, захист інформації від можливості несанкціонованого доступу до неї.
- Доступність – стан інформації, що дозволяє суб'єкту з правами доступу, забезпечити отримання доступ до інформації в момент необхідності.

Поняття що так інформаційні властивості, передбачає в свою чергу дотримання правил для будь–якого інформаційного простору, але розробники шкідливого ПО і хакери можуть використовувати набагато більше речей ніж звичайний користувач. Вони завжди прагнуть отримати при зломі або зараженні веб-ресурсу доступ до важливої інформації або використовувати ресурси хостингу такі як:

- Паролі - Часткова або повна втрата контролю над ресурсом, фінансами і базою даних.
- Клієнтська база (email користувачів і інші дані) - Розсилка користувачам спаму для отримання матеріальних вигод.
- Платіжні дані користувачів - Заманювання клієнтів на підставні сторінки для крадіжки облікових записів від акаунтів банків, платіжних систем і сервісів для отримання доступів, паролів,

конфіденційних даних, які потрібні для успішного проведення фінансових операцій. Це збитки і втрата довіри клієнтів.

- Можливість використовувати сайт для реклами - На сторінках веб-ресурсу буде з'являтися реклама, що приносить дохід зловмисникам, а вам незручності і обурення користувачів.
- Можливість проведення атак на інші сайти - Ресурс використовується в якості проксі-сервера, через нього здійснюють атаки зараженні інших сайтів і отримання зберігається там інформації.

Крім втрати фінансів, баз даних, контролю над управлінням ресурсу, підміни контенту і поява клонів, власники зламаного сайту стикаються з такими наслідками як відчутне скорочення трафіку через втрату довіри і погіршення репутації які принесуть великі втрати і матеріальні. Усе це не дуже гарно впливає на бізнес. Для того щоб протидіяти усім можливим проблемам потрібно розуміти методи атак веб-ресурсів.

2.2 Аналіз атак на веб-ресурси

Інформаційна безпека стоїть на першому місці у всьому світі, тому що ніхто не хоче щоб його особиста інформація була в когось стороннього. Тому аналізуючи існуючі данні можна зрозуміти що є найпоширеніші методи атак.

Таблиця 2.1 – Види атак, їх розповсюдження та короткий опис протидії.

№	Вид атаки	Вразливість веб-ресурсу	Протидія
1	«Insufficient transport layer protection» - отримання даних під час передавання.	70%	Використання протоколу HTTPS
2	«Information leakage»- витік інформації	56%	Тестування програмної частини , перевірка повідомлень на стороні сервера

Продовження таблиці 2.1

3	«Cross-site scripting» - міжсайтове використання сценаріїв	45%	Очищення та валідація даних
4	«Brute force» - підбір паролів	28%	Створення складних паролей, аналіз вхідних данних
5	«Content spoofing» - підміна даних	26%	Відмова від фреймів ,неможливість передачі шлязів до файлів
6	«Cross-site request forgery» - атаки на відвідувачів веб-ресурсів	23%	Перевірка вхідних даних
7	«URL redirector abuse» - перенаправлення на інші сайти через підміну посилання	16%	Валідація даних
8	«Predictable resource location» - знаходження прихованого функціоналу та даних	15%	Контролювання доступами до файлів серверу
9	SQL Injection – впровадження вільних SQL команд	9%	Фільтрування запитів, контроль даних, шифрування даних

Найпопулярнішою є атака «Insufficient transport layer protection». Атака базується на перехопленні даних під час передачі. Вона використовуються в 70% атак на ресурси. Для того щоб захиститися від подібних атак достатньо використовувати стандартний протокол HTTPS.

Витік інформації («Information leakage»), це ситуація коли система, створена так, щоб бути закритою для перехоплення, але розкриває деяку інформацію в результаті неправильної роботи програмного забезпечення. Інколи також виникає в разі рушення логіки програми. Дану атаку

використовують на 56% веб-ресурсів. Для уникнення таких проблем потрібно ретельно проаналізувати систему на помилки, ретельно тестувати веб-ресурс.

Міжсайтове використання сценаріїв («Cross-site scripting»), це атака яка дозволяє передавати JavaScript на виконання не на сервері, а на браузері користувача. Атаки такого плану ще мають назву як HTML-інєкції. Механізм схожий с SQL-інєкціями, але впровадження коду виконується не на сервер, базу даних, а в браузері користувача. Для того щоб захистити веб-ресурс від такої ін’єкції потрібно проводити ретельну валідацію вхідних даних та проводити очищення.

Підбір паролів або генерація запитів («Brute force») є одним з так названих грубих варіантів. Злодій генерує та перевіряє усі можливі паролі та парольні фрази поки не знайде правильні дані. Для того щоб захиститися від такого способу злому потрібно забезпечити користувачів складними паролями та правильно налаштувати сервер на вхідні данні.

Атака «Content spoofing», це атака підміни даних через заміну інформації та даних на сторінці. Зловмисник використовуючи цей спосіб змушує користувача повірити, що сторінка реальна та згенерована самим веб-ресурсом, що не є правдою. Для того щоб захистися від такого виду атак потрібно від фреймів, а також ніколи не давати доступу або параметри до локальних файлів.

Наступним видом атак є атаки відвідувачів веб-ресурсів «Cross-site request forgery». Якщо жертва переходить по посиланню на підозрілий веб-ресурс, створений зловмисником, тоді від її особи таємно відправляється запит на інший сервер з даними користувача, після чого відбувається швидка операція зняття грошей на рахунок зловмисника. Результатом є грабіж інформації або грошей звичайного користувача зловмисником використовуючи підроблений веб-ресурс. Для того щоб захиститися від цього необхідно проводити перевірку вхідних даних з форми вводу інформації веб-ресурсу, наприклад додавши унікального додатку або капчі.

Ще одним схожим типом атаки э перенаправлення на інші сайти через підміну посилання («URL redirector abuse»). Алгоритм роботи цього алгоритму в тому що як і багато інших він є помилкою перевірки вхідних даних. Для вирішення цієї атаки потрібно лише розробити валідацію вхідних даних.

Ще одним типом популярних атак є знаходження прихованого функціоналу та даних («Predictable resource location»). Вирішенням такої проблеми ж контроль доступу до файлів сервера.

SQL–ін'єкції це атака на базу даних, що працює за принципом вбудовування вільних SQL-команд, в результаті якого змінюється логіка запиту. Це призводить до заволодіння усією потрібною інформації для зловмисника. Прикладом ін'єкціє є такий вразливий сценарій рис 2.1:

```
<?php
…
$data=mysqli_query($link,"SELECT * FROM profiles WHERE id=".$_GET["id"]);
…
?> [1]
```

Рисунок 2.1 – Приклад вразливого сценарію.

У даному прикладі можна побачити , що поля id будуть порівнюватися зі значенням параметру id, що був переданий через URL. Якщо вказати значення наприклад 3, то звичайний користувач побачить свій профіль, але якщо зловмисник буде вже вписувати в якості параметру id рядок 3 OR UserName= "Administrator", тоді запит до SQL буде мати вигляд вже стандартного запиту на пошук інформації адміністратора, SELECT * FROM profiles WHERE id=3 OR UserName= "Administrator". В результаті хакер отримає інформацію про акаунт Administrator.

2.3 Методи захисту веб-ресурсу

Статистика свідчить, що методи які орієнтовані на захист від певних типів атак не дуже ефективний метод. Максимальну школу зловмисник завдає веб-ресурсу, як правило використовуючи та комбінуючи декілька типів атак. Тому головною задачею системи захисту веб-ресурсу є розробка максимально ефективної стратегії протидії будь яким атакам зловмисників.

Розробка захисту веб-ресурсу є нетривіальною задачею над якою потрібно багато працювати для майбутнього правильного розвитку компанії. Створення удосконаленого методу захисту веб-ресурсу дозволить посприяти розвитку захисту системи компанії, яка відрізняється від аналогів своєю здатністю здійснювати системний та статичний аналіз вразливостей. Це дозволяю здійснити якісний захист від декількох типів атак зловмисників. Структура такого методу зображена на (рис 2.2).

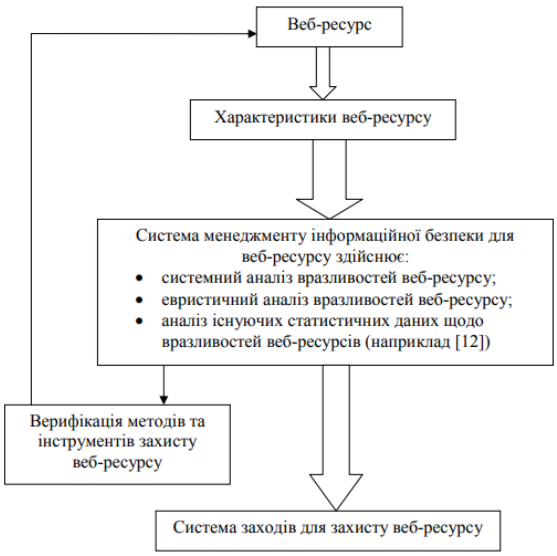

Рисунок 2.2 – Структура методу формування системи захисту веб-ресурсу.

В свою чергу уся структура захисту ділиться ще на три великі групи такі як захист CMS, захист сервера та захист комп’ютера адміністратора (рис 2.3):

Рисунок 2.3 – Методи захисту сайту.

Ці три групи забезпечують захист на різних рівнях моделі OSI, та запобігають впливу на безпеку зовнішніх факторів.

2.2.1 Захист CMS

CMS - система управління сайтом і всією інформацією на ньому. Це серце, а тому захист починається саме з нього. Використовуйте наступні методи і максимально забезпечте ресурс. Протокол SSL забезпечує надійний захист даних в інтернеті за рахунок зашифрованої передачі інформації. Для того щоб такий рівень безпеки був доступний, важливо мати SSL-сертифікат - електронний цифровий підпис вашого сайту, що містить:

- доменне ім'я;
- відомості про юридичну особу, на яку оформляється сертифікат;
- реальне місцезнаходження власника;
- термін дії;
- основні дані про постачальника сертифіката.

За допомогою цього підпису ви підтверджуєте, що домен належить реальної компанії, а власник використовує секретний ключ законно.

Одне їх важливих умов безпечного функціонування ресурсу - регулярне оновлення CMS. Кожна нова версія «движка» усуває помилки і закриває незахищені місця, через які легко зламати сайт.

Оновлення допоможе уникнути атак через уразливість в скриптах плагінів і CMS. Це не гарантує 100-відсотковий захист, але мінімізує ризик виникнення такого форсінга, як:

- SQL-ін'єкції;
- шелли;
- XSS.

Якщо створюється самостійний проект без участі конструкторів тоді використовують скрипти та плагіни що вже написані, тому їх скачують і додають до свого веб-ресурсу. Але часто в плагінах і програмах знаходяться трояни у вигляді вірусу, бекдора або шелла.

Захистити сайт від злому можливо тільки ретельна перевірка джерел і використання ліцензованих компонентів допоможуть уникнути «падіння» ресурсу. Якщо завантажуєте програми і скрипти, вибирайте тільки офіційні джерела розробників і постачальників.

Додатковий захист ресурсу забезпечують спеціальні плагіни. Їх дуже багато, кожен пропонує комплексне рішення для оборони від тих чи інших атак, сканування на наявність вірусів і спроб злому, створення копій ресурсу та інших функцій. Їх швидко встановлювати, вони прості в настройках і ефективно працюють.

2.2.2 Захист серверів

Сервер є серцем усього веб-ресурсу. Будь які помилки в скриптах, права доступу до сервера у всій компанії, а не в окремій керуючій групі. Усе це може призвести до появи сторонніх скриптів що будуть навантажувати ресурси серверу з подальшими проблемами для звичайних користувачів. Також втрата або крадіжка даних користувачів, що в свою чергу може вплинути на імідж компанії.

Для того щоб сервер не був доступний для зловмисників, є елементарні такі правила технічної безпеки як:

- Обмежуйте користувачів в правах на доступ до бази даних. Не давайте неперевіреними людям прав доступу і паролі для входу в адміністративну панель, можливість додавати HTML-код.
- Відстежуйте, що вводить користувач на сайті. Це допоможе захистити сайт від спаму і XSS-атак, які часто проводять через форми зворотного зв'язку, замовлення, підписки і так далі.
- Постійно міняйте паролі.
- При настройці доступу до серверу використовуйте аутентифікацію по ключам SSH замість паролів. Це дозволить не піклуватися часто за пароль, оскільки SSH ключі складні, інколи їх неможливо зламати за допомогою підбору.
- Встановить файрвол, він допомагає серверу захищатися від DDOS атак.

2.2.3 Захист ПК

Комп'ютери того, хто користується акантом адміністратор і має доступ до важливої інформації на сайті, а також тих, у кого є вихід до сервера, потрібно перевіряти на наявність вірусів. Чи впораються з цим антивіруси. Постійний захист веб-сайтів гарантує збереження даних та інформації. Важливо оновлювати програми до актуальної версії. Також використовувати брандмауери для контролю доступу програм до глобальної мережі.

2.4 Проектування веб-додатку

Використання технології розробки веб-ресурсу забезпечують доступ до створення та розвитку будь-яких інформаційних ресурсів. Гіпертекстові документи перетворюються в стандарті HTML. Ці документи можуть зберігатися в статичному вигляді на диску, або в динамічному. Для створення правильної функціональної системи веб-проекту, потребується розробка структурно-функціональних моделей. Ця модель забезпечує грамотний розвиток організації бізнес-процесів.

2.4.1 Структурно-функціональне моделювання

Для моделювання функціонування системи в даній роботі була використана методологія IDEF0, ця методологія дозволяє розглянути та дослідити функціонал системи не змішуючи його з програмною реалізацією. Таким чином за використанням даної методології є можливим розглянути цілісну картину системи відкинувши нюанси її реалізації.

Для побудови діаграми використовувались такі дані:

- Вхідні данні: запит на замовлення товару.
- Керівництво: асортимент товару та керівництво по оформленню замовлення.
- Використані механізми: вебресурс, апаратне забезпечення та технічне забезпечення.
- Вихідні данні: оформлене замовлення обраного користувачем товару

Функціональна діаграма приведена нижче (рис 2.4)

Рисунок 2.4 - Функціональна діаграма.

Після побудови функціональної діаграми необхідно більш детально розібрати процес. Тому проведемо декомпозицію процесу замовлення товару.

Процес замовлення товару поділяється на підпроцеси, які і необхідно розглянути, цими підпроцесами є :

- вибір товару;
- вибір типу оплати;
- вибір способу доставки.

Таблиця 2.2 – Вхідні та вихідні дані для діаграми оформлення замовлення

Стрілка/ Підпроцес	**Вхідні дані**	**Управління**	**Механізми**	**Вихідні дані**
Вибір товару	ID товару або його назва	Керівництво по оформленню замовлення	Вебресурс, апаратне забезпечення та технічне забезпечення	Підтвердження обраного товару
Вибір типу оплати	Можливі методи оплати товару			Вибір методу оплати
Вибір способу доставки	Можливі методи доставки товару			Вибір способу доставки та підтвердження замовлення

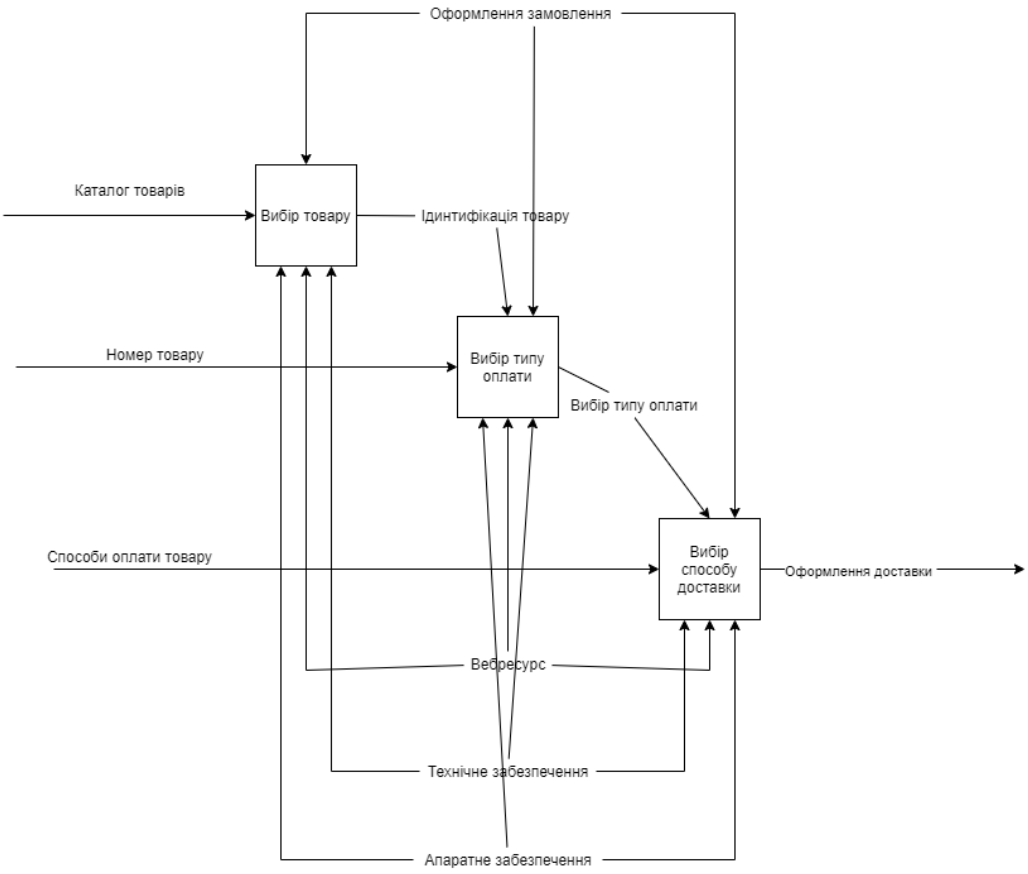

Рисунок 2.5 – Діаграма оформлення замовлення в веб-ресурсі

На діаграмі зображений процес оформлення заказу користувачем на веб-додатку.

3 РОЗРОБКА ВЕБ-ДОДАТКУ

3.1 Архітектура веб-додатку

Веб-додаток це продукт, комп'ютерна програма, яка створена для виконання певної функції, в якості відображення результату використовує веб-браузер. Додаток може бути будь яким, елементарно простим блогом, або ж багатокористувацький мобільний додаток. Але усі вони починають з проектування архітектури, що є описом компонентів, які були використані при розробці того чи іншого додатку.

Архітектура веб-додатку компанії «Двері Білорусії» має в собі:

- база данних, що містить усю інформацію:
- дизайн, візуальне відображення:
- елементи захисту коду.

Все це зображене на діаграмі HLD (High Level Design) [12]. Призначення такої діаграми в тому щоб описати усі компоненти використані при розробці.

3.2 Розробка дизайну веб-додатку

Розробка дизайну веб-додатку є процесом створення макету, шаблону. Перш ніж почати розробляти макет додатку, потрібно розробити майбутню структуру веб-сторнінки, що відповідає за розміщення блоків. Відповідно до потреб від продукту створюється макет (рис.3.1).

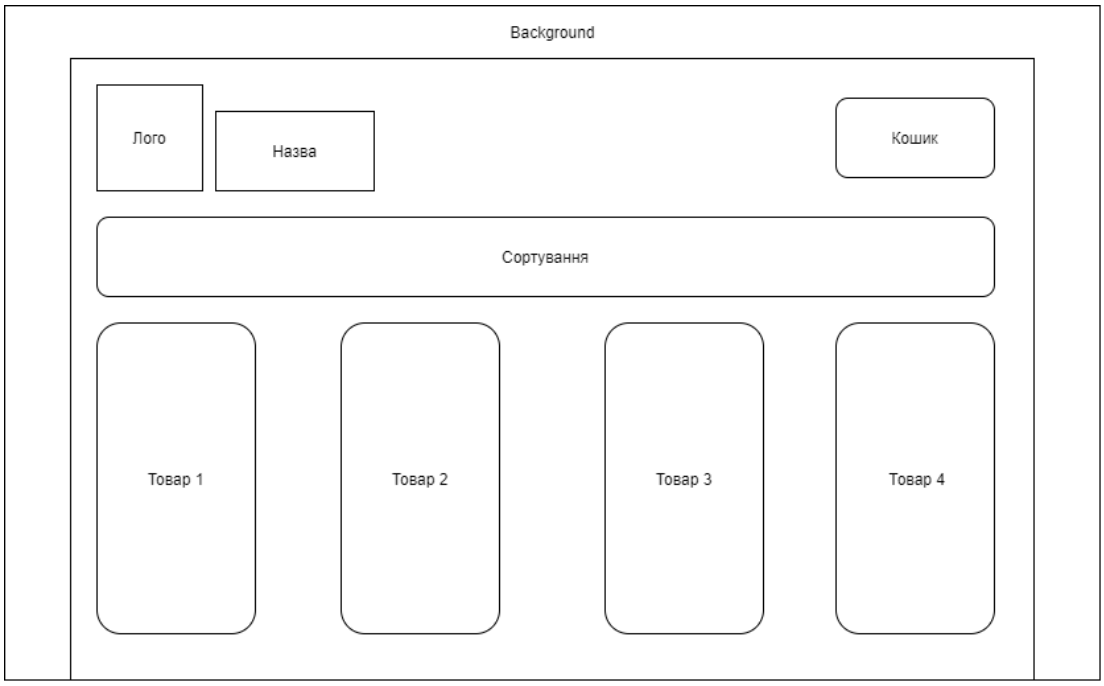

Рисунок 3.1 – Створений макет веб-додатку

В результаті по макету бив розроблений декілька шаблонів дизайну сайту(рис.3.2-3.3).

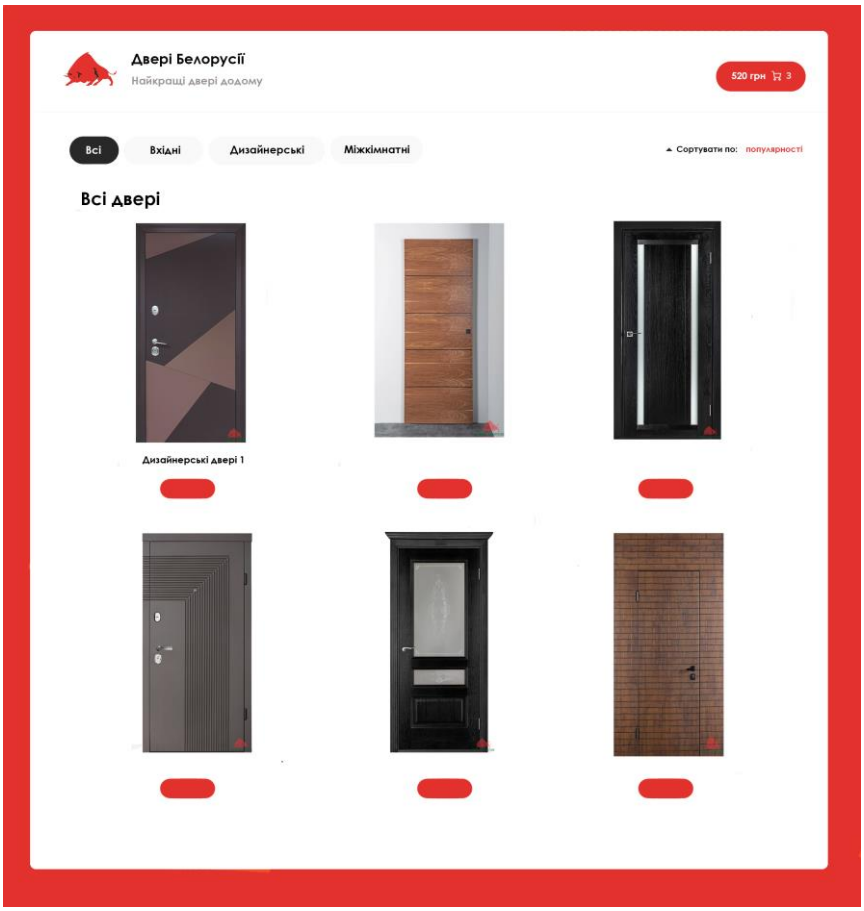

Рисунок 3.2 – Перший варіант веб-додатку

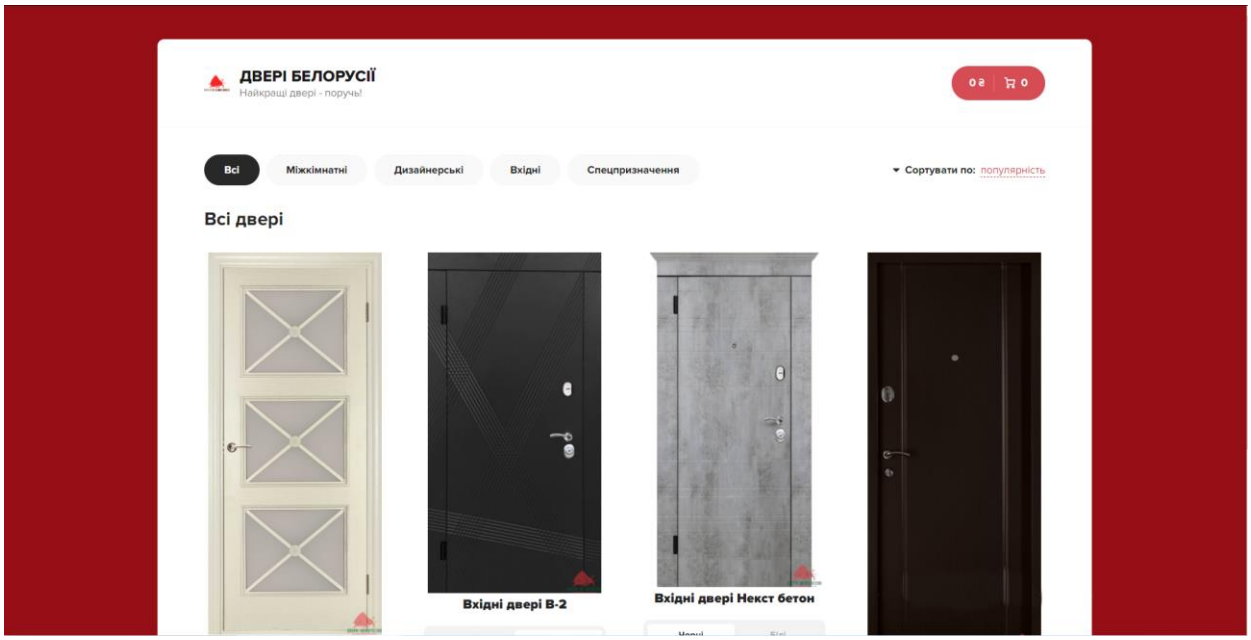

Рисунок 3.3 – Другий варіант веб-додатку

Для забезпечення успіху компанії та щоб вона запам’ятовувалась був використаний в основній своїй гаммі червоний колір який притягує увагу, білий що концентрує її на контенті та асфальтовий. Налаштування спеціальних шрифтів (Forts) виконувалося за допомогою програмно забезпечення Figma та Google.

3.3 Програмна реалізація веб-додатку

Під час етапу програмної реалізації додатку було підготовлено середовище React.js. Для його встановлення потрібно встановити node.js (рис. 3.4).

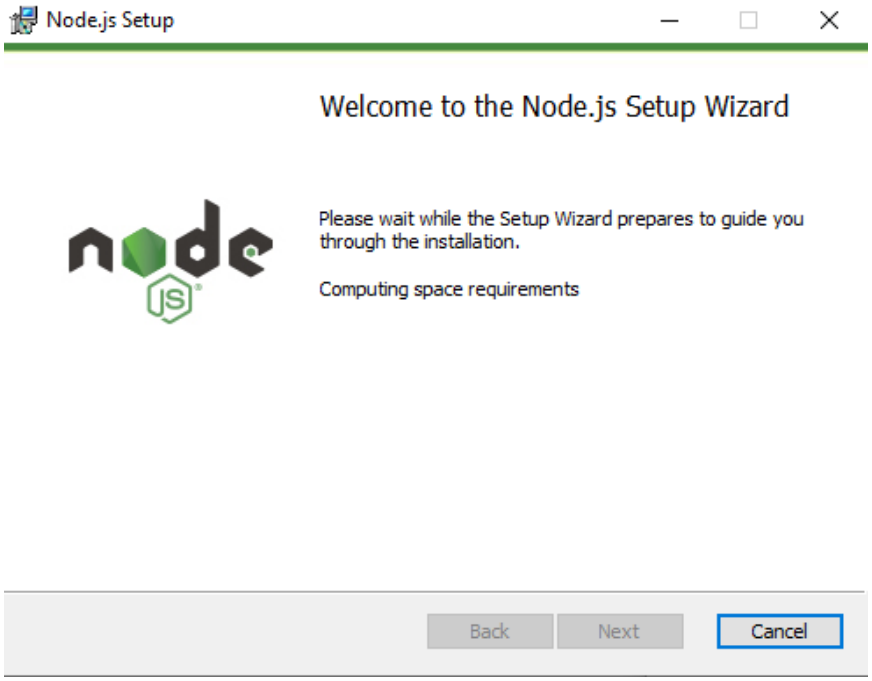

Рисунок 3.4 – Меню встановлення Node.js

Після встановлення програмного забезпечення комп'ютер сприймає команди на мові програмування Javascript.

Наступним етапом є створення нового проекту в Visual Studio Code, в якому ми створюємо дві папки. Одна папка client, вона відповідає клієнтську частину додатку. Друга папка це server, в ній буде знаходитися увесь backend.

Далі ми створюємо в папці server мы проініціалізуємо проект за допомогою команди "npm init –y". Далі ми встановлюємо бібліотеки використовуючи команду "npm install".

Далі ми створюємо шаблон веб-додатку для подальшого заповненя. Також створюємо усі блоки роздільно для оптимізації коду. Першим створюємо шапку або header (рис. 3.5).

```jsx
// Header.jsx
// src > components > Header.jsx > Header
import React from 'react';
import { Link } from 'react-router-dom';
import { useSelector } from 'react-redux';

import logoSvg from '../assets/img/logo.png';
import Button from './Button';

function Header() {
  const { totalPrice, totalCount } = useSelector(({ cart }) => cart);

  return (
    <div className="header">
      <div className="container">
        <Link to="/">
          <div className="header__logo">
            <img width="38" src={logoSvg} alt="logo" />
            <div>
              <h1>Двері Белорусії</h1>
              <p>Найкращі двері - поручь!</p>
            </div>
          </div>
        </Link>

        <div className="header__cart">
          <Link to="/cart">
            <Button className="button--cart">
              <span>{totalPrice} ₴</span>
              <div className="button__delimiter"></div>
              <svg
                width="18"
                height="18"
                viewBox="0 0 18 18"
                fill="none"
                xmlns="http://www.w3.org/2000/svg">
                <path
                  d="M6.33333 16.3333C7.06971 16.3333 7.66667 15.7364 7.66667 15C7.66667 14.2636 7.06
                  stroke="white"
                  strokeWidth="1.8"
                  strokeLinecap="round"
                  strokeLinejoin="round"
                />
                <path
                  d="M14.3333 16.3333C15.0697 16.3333 15.6667 15.7364 15.6667 15C15.6667 14.2636 15.0
                  stroke="white"
                  strokeWidth="1.8"
                  strokeLinecap="round"
```

Рисунок 3.5 – Створення Header

Наступним етапом є створення блоку кнопок (button), також створюємо блок категорій товару (categories), кошика (cartitem) (рис.3.6-3.8).

Button.jsx

src > components > Button.jsx > ...

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

const Button = ({ onClick, className, outline, children }) => {
  return (
    <button
      onClick={onClick}
      className={classNames('button', className, {
        'button--outline': outline,
      })}>
      {children}
    </button>
  );
};

Button.propTypes = {
  onClick: PropTypes.func,
};

export default Button;
```

Рисунок 3.6 – Створення Button

Categories.jsx

src > components > Categories.jsx > Categories > Categories

```jsx
import React from 'react';
import PropTypes from 'prop-types';

const Categories = React.memo(function Categories({ activeCategory, items, onClickCate
  return (
    <div className="categories">
      <ul>
        <li
          className={activeCategory === null ? 'active' : ''}
          onClick={() => onClickCategory(null)}>
          Bci
        </li>
        {items &&
          items.map((name, index) => (
            <li
              className={activeCategory === index ? 'active' : ''}
              onClick={() => onClickCategory(index)}
              key={`${name}_${index}`}>
              {name}
            </li>
          ))}
      </ul>
    </div>
  );
});

Categories.propTypes = {
  // activeCategory: PropTypes.oneOf([PropTypes.number, null]),
  items: PropTypes.arrayOf(PropTypes.string).isRequired,
  onClickCategory: PropTypes.func.isRequired,
};

Categories.defaultProps = { activeCategory: null, items: [] };

export default Categories;
```

Рисунок 3.7 – Створення Categories

CartItem.jsx

src > components > CartItem.jsx > CartItem

```jsx
import React from 'react';
import Button from './Button';

const CartItem = ({ id, name, type, size, totalPrice, totalCount, onRemove, onMinus, onPlus }) => {
  const handleRemoveClick = () => {
    onRemove(id);
  };

  const handlePlusItem = () => {
    onPlus(id);
  };

  const handleMinusItem = () => {
    onMinus(id);
  };

  return (
    <div className="cart__item">
      <div className="cart__item-img">
        <img
          className="pizza-block__image"
          src="https://mir-s3-cdn-cf.behance.net/project_modules/max_1200/e5ba9735176905.56ec4501b7d
          alt="Двері"
        />
      </div>
      <div className="cart__item-info">
        <h3>{name}</h3>
        <p>
          {type} колір, {size} см.
        </p>
      </div>
      <div className="cart__item-count">
        <div
          onClick={handleMinusItem}
          className="button button--outline button--circle cart__item-count-minus">
          <svg
            width="10"
            height="10"
            viewBox="0 0 10 10"
            fill="none"
            xmlns="http://www.w3.org/2000/svg">
            <path
              d="M5.92001 3.84V5.76V8.64C5.92001 9.17016 5.49017 9.6 4.96001 9.6C4.42985 9.6 4.00001
              fill="#EB5A1E"
            />
            <path
              d="M5.75998 5.92001L3.83998 5.92001L0.959977 5.92001C0.429817 5.92001 -2.29533e-05 5.4
              fill="#EB5A1E"
```

Рисунок 3.8 – Створення Cartitem

Далі створюємо потрібні сторінки на яких будуть розміщений контент, або створений кошик. Ці дві сторінки будуть взаємозв’язані за допомогою скрипту який переадресовує з однієї сторінки на іншу.

3.4 Захист веб-додатку

Використовуючи самий простий варіант для забезпечення коду захисту було використано обфускацію. Обфускація це перетворення звичайного коду в заплутаний зі збереженням його функціональності. Цей елементарний прийом відбиває охоту деякого відсотка користувачів мати змогу розібрати

його, для подальшого використання. Прикладом обфускації є головна сторінка веб-додатку(рис.3.9).

eval(function(p,a,c,k,e,r){e=function(c){return(c<a?'':e(parseInt(c/a)))+((c=c%a)>35?String.fromCharCode(c+29):c.toString(36))};if(!''.replace(/^/,String)){while(c--)r[e(c)]=k[c]||e(c);k=[function(e){return r[e]}];e=function(){return'\\w+'};c=1};while(c--)if(k[c])p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c]);return p}('w 8 x\'P\';w a x\'Q-R\';1 c=8.S(T c({3,f,g}){1[h,i]=8.U(n);1 o=8.V();1 y=3.W((2)=>2.p===f).z;1 A=()=>{i(!h)};1 B=(j)=>{1 k=j.k||(j.C&&j.C());D(!k.X(o.Y)){i(n)}};1 E=(l)=>{D(g){g(l)}i(n)};8.Z(()=>{11.12.13(\'14\',B)},[]);15(<7 16={o}e="17"><7 e="18"><q e={h?\'19\':\'\'}1a="10"1b="6"1c="0 0 10 6"F="1d"1e="1f://1g.1h.1i/1j/q"><k d="1k 1l 5.r 9.G 5.s 9.H 5.1m.1n 5.I 9.1o 5.J 9.1p 5.1q.1r.1s 5.J 0.1t 5.I 0.K 5.1u.m 5.s 0 5.r 0 1v 4.t 0.m 4.u 0.K 4.1w.1x 0.1y.u 0.m 4.t 0 5 1z.r 0 5.s 0.m 5.1A 0.1B.H 4.1C.G 4.u 10 4.t 10 1D"F="#1E"/></q>Сортуватипо:<L M={A}>{y}</L></7>{h&&(<7 e="1F"><N>{3&&3.1G((2,I)=>(<O M={()=>E(2)}e={f===2.p?\'1H\':\'\'}1I={`${2.p}1J${I}`}>{2.z}</O>))}</N></7>)}</7>}));c.1K={f:a.1L.v,3:a.1M(a.1N).v,g:a.1O.v,};c.1P={3:[],};1Q 1R c;',62,116,'|const|obj|items||||div|React||PropTypes||SortPopup||className|activeSortType|onClickSortType|visiblePopup|setVisiblePopup|event|path|index|061849|false|sortRef|type|svg|16927|31576|83073|68424|isRequired|import|from|activeLabel|name|toggleVisiblePopup|handleOutsideClick|composedPath|if|onSelectItem|fill|93815|81445|56315|625|185547|span|onClick|ul|li|react|prop|types|memo|function|useState|useRef|find|includes|current|useEffect||document|body|addEventListener|click|return|ref|sort|sort__label|rotated|width|height|viewBox|none|xmlns|http|www|w3|org|2000|M10|5C10|43945C9|69075|54427|375|625H0|625C0|455729|309245|43945C0|5C0|56055L4|56055|185547C4|0C5|43945|185547L9|56055C9|5Z|2C2C2C|sort__popup|map|active|key|_|propTypes|string|arrayOf|object|func|defaultProps|export|default'.split('|'),0,{}))

Рисунок 3.9 – Обфускований код веб-додатку

Це є простий, але діючий трюк що дає змогу на перший час бути спокійним за проект.

Для того щоб не було плагіату контенту с веб-додатку, на всі зображення були нанесені водяні знаки(рис.3.10). Це дозволяє захистити зображення від крадіжки.

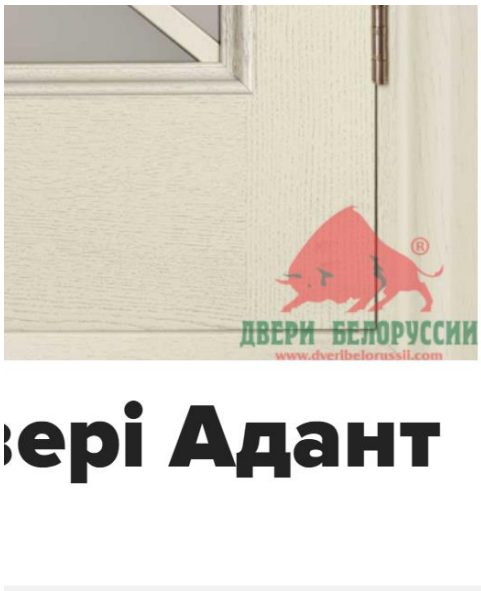

Рисунок 3.10 – Водяний знак

Для забезпечення тексту захисту, можна використати підписи. Усі новини, статі або товари на сайті в описі можна залишати об’єкт авторського права. Також можна додати посилання на справжнє джерело інформації.

3.5 Демонстрація роботи веб-додатку

Головна сторінка веб-додатку складається з шести областей що зв’язані між собою. Одразу як ми зайдемо на сайт ми можемо побачити товар та усі варіанти сортування його. Також з правої сторони є наступна сторінка і це кошик для швидкого оформлення заказу(рис. 3.11-3.12).

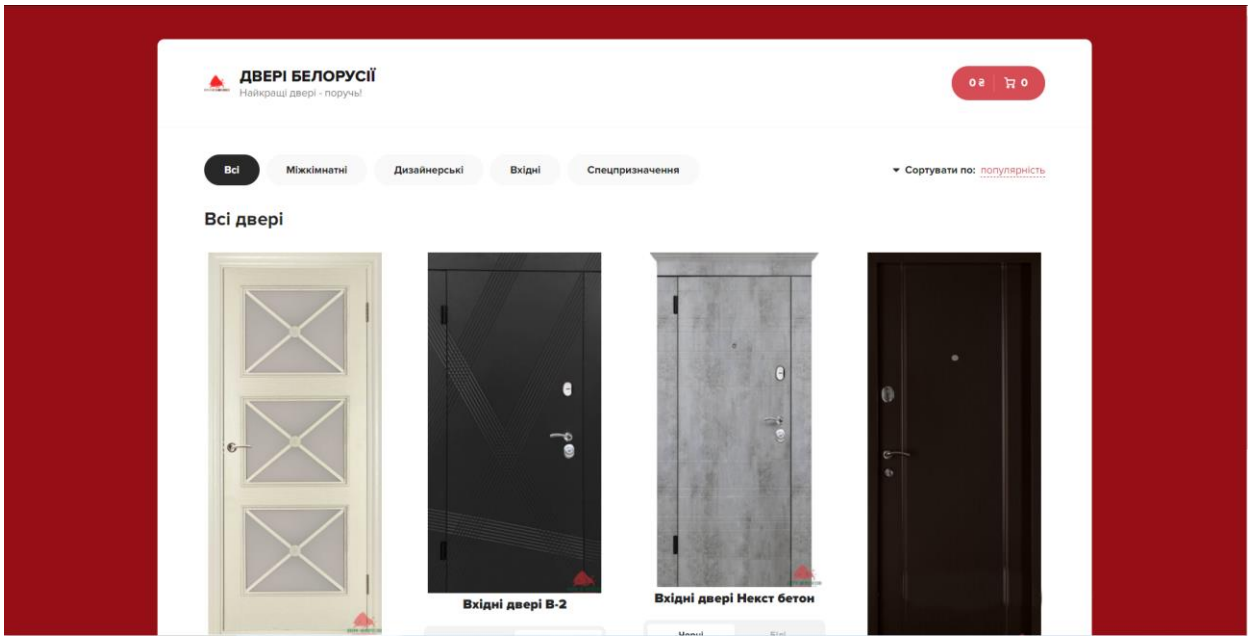

Рисунок 3.11 – Головна сторінка

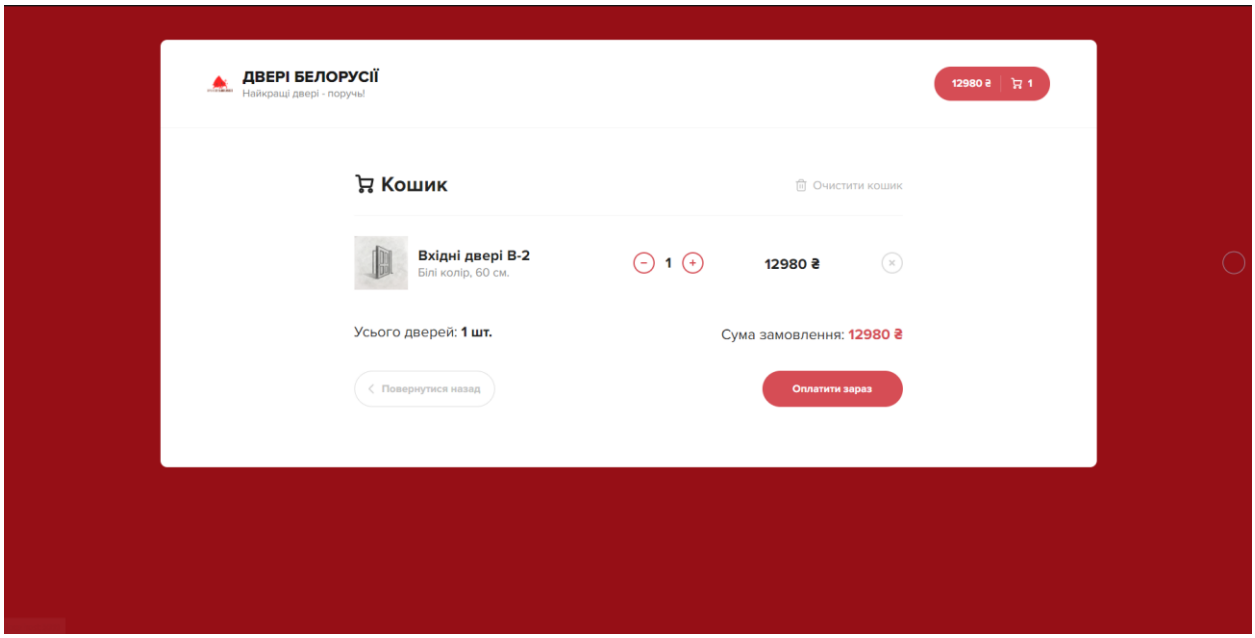

Рисунок 3.12 – Сторінка кошику

Після огляду всіх дверей що знаходяться на веб-сторінці, користувач може обрати будь яке полотно що потрібно, вибрати колір з тих що є в наявності та обрати розмір дверного полотна (рис.3.13-3.14).

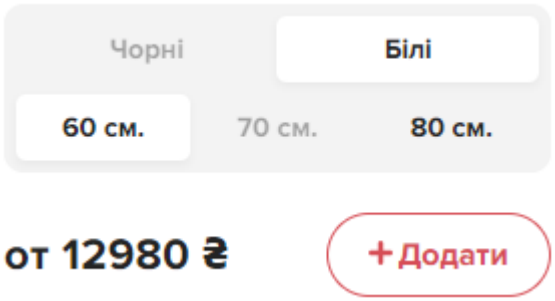

Рисунок 3.13 – Варіант з неповними характеристиками полотна

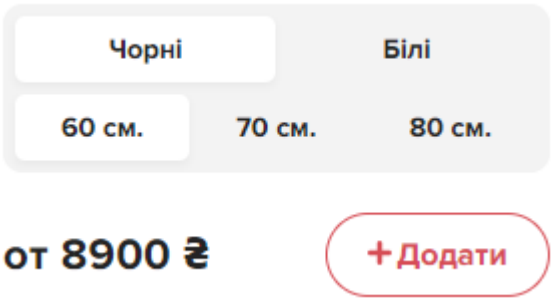

Рисунок 3.14 – Варіант з усіма можливими характеристиками

Використовуючи функцію сортування по типу дверей можна швидко підібрати потрібне полотно та відстежувати новинки. Функція дозволяє відображати всі двері, що є в наявності, тільки міжкімнатні двері, розроблені компанією дизайнерські унікальні двері та двері спецпризначення (рис. 3.15-3.19).

Такі типи дверей створені по противопожежній системі, влагостійкі та індивідуального призначення які можуть комбінувати характеристики.

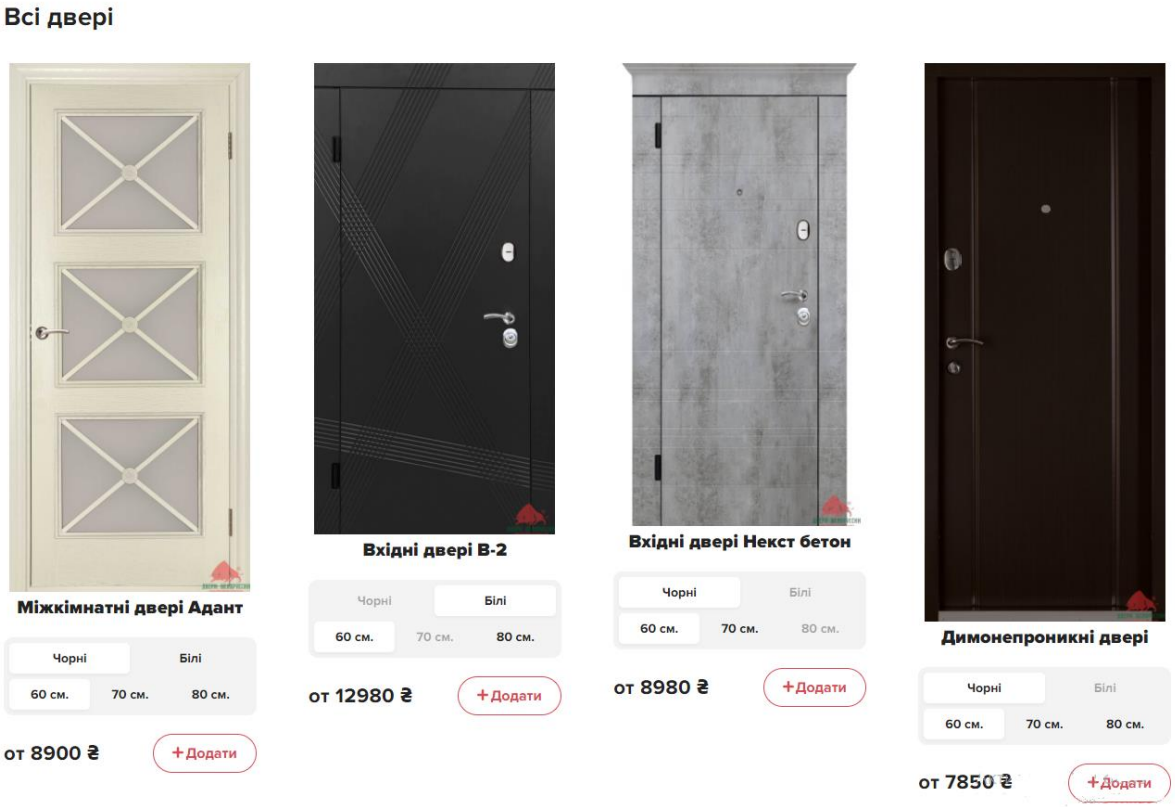

Рисунок 3.15 – Розділ з усіма дверима

Цей спеціальний розділ комбінує в собі усі розділи, та відображає увесь товар, що є на сайті. Це допомагає користувачу легко ознайомитися з усім асортиментом що має компанія для подальшого створення замовлення або консультації з консультантом.

Рисунок 3.16 – Розділ з міжкімнатини дверима

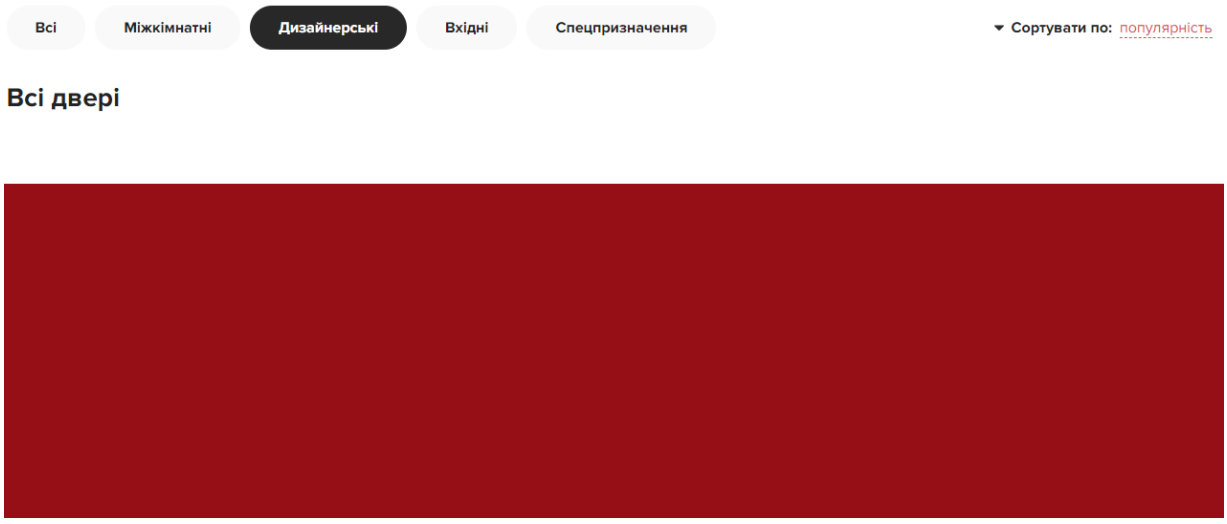

Рисунок 3.17 – Розділ в якому немає ніяких дверей

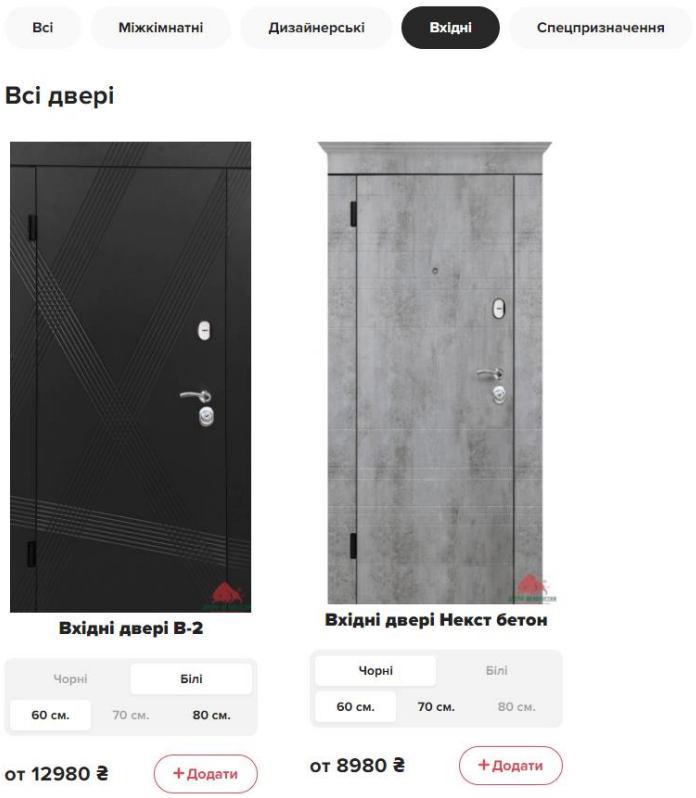

Рисунок 3.18 – Розділ вхідних дверей

Рисунок 3.19 – Розділ з дверми спецпризначення

Коли користувач визначається з вибором товару, перед додаванням до кошику потрібно вибрати характеристики полотна що представленні. Далі додати до кошику і оформити замовлення. Якщо потребується декілька однакових дверей , є можливість додати їх в кошику, а не шукати знову потрібну модель або ж видалити непотрібні моделі дверей(рис 3.20-3.22).

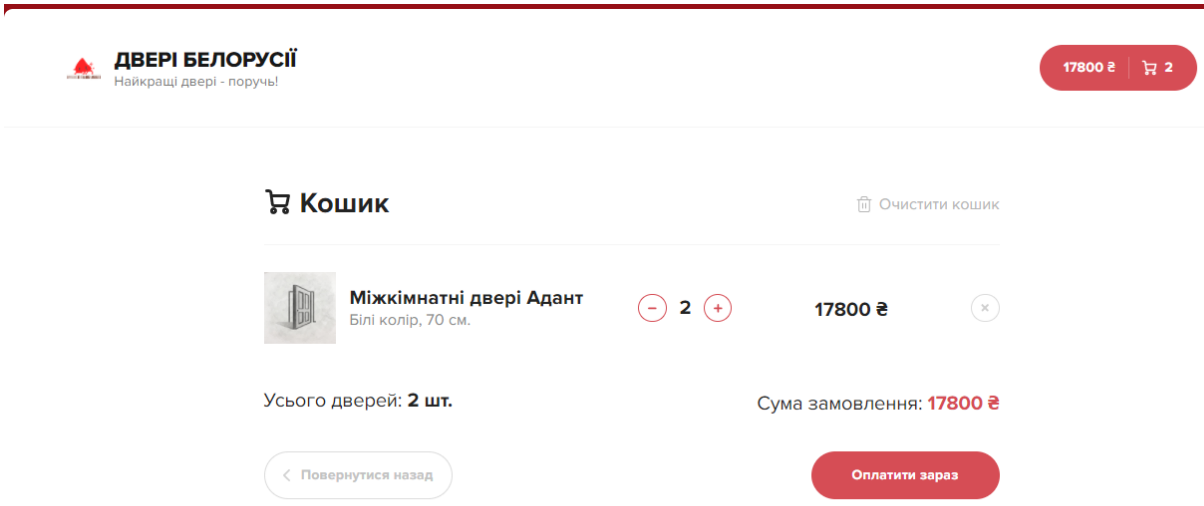

Рисунок 3.20 – Товар в кошику

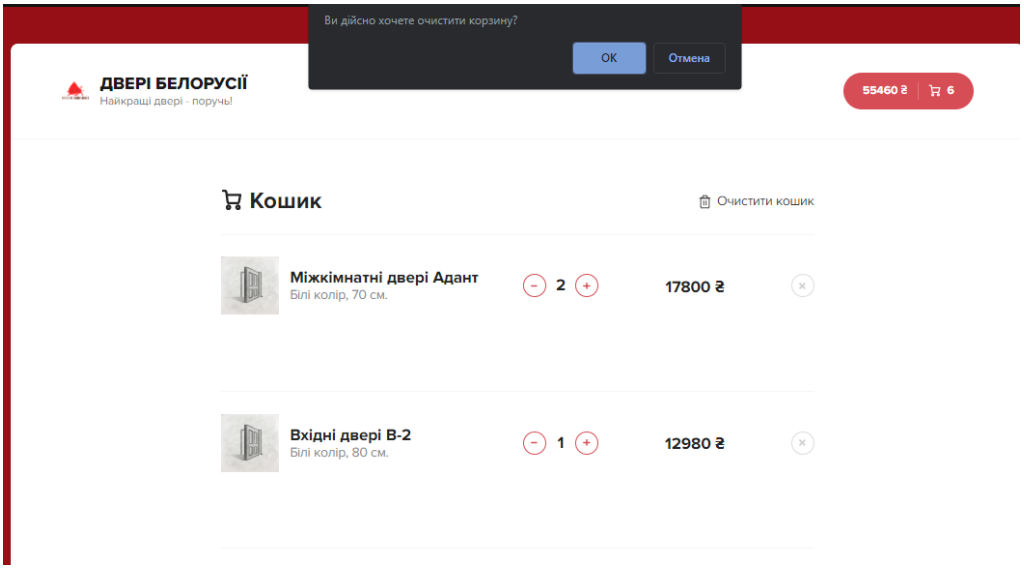

Рисунок 3.21 – Очищення кошику

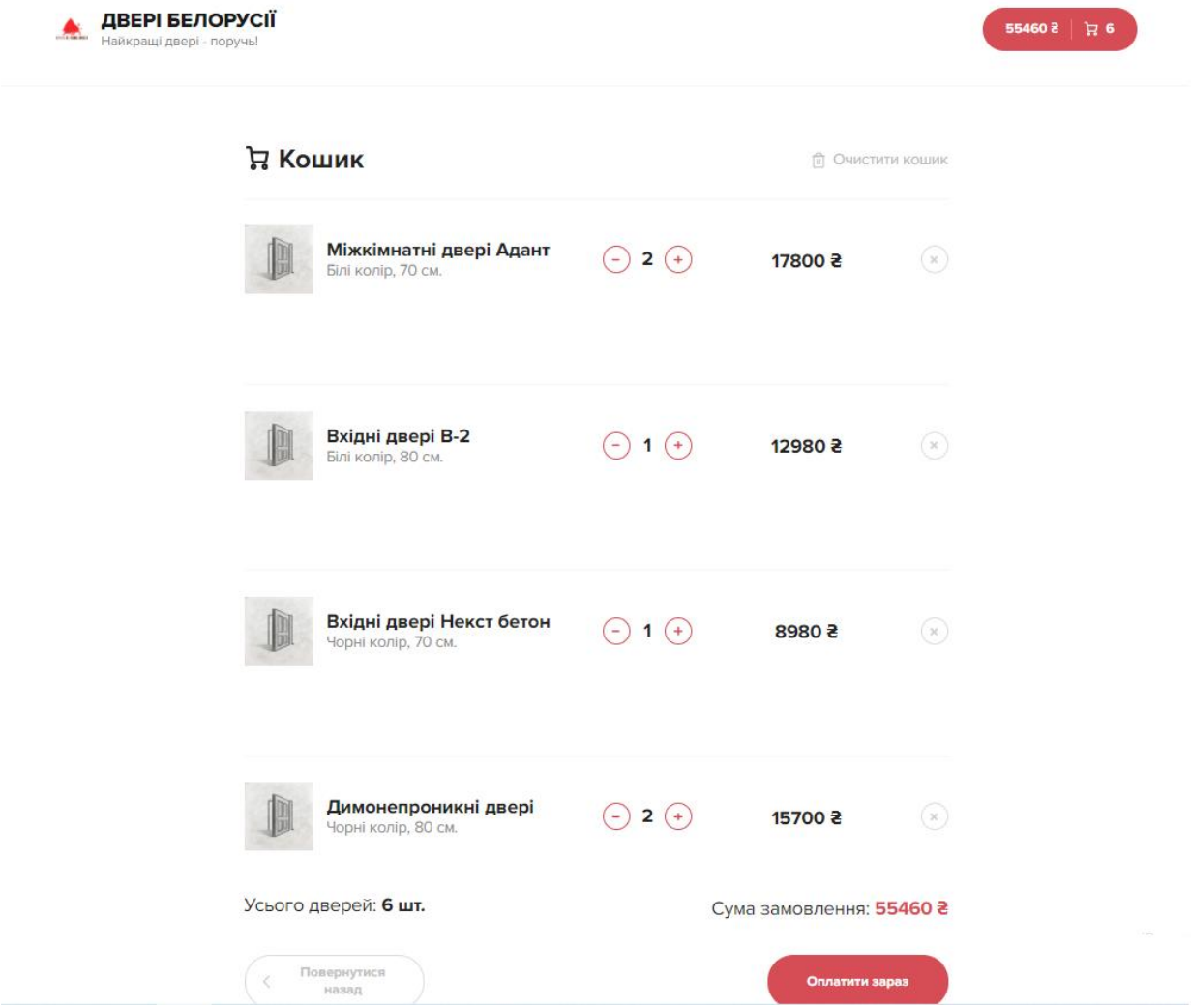

Рисунок 3.22 – Додавання декілька товарів

Замовлення повністю формується від вибору клієнта за короткий час з одразу відображеною ціною.

ВИСНОВОК

В результаті виконаної роботи був створений тестовий веб-додаток з використанням системи React.js.

Під час реалізації проекту було розглянуто і проаналізовано значну кількість варіацій, і варіантів атак кібер-злочинців. В результаті чого було визначено актуальні проблеми з якими потрібно працювати та боротися. Також були наведені приклади протидії атакам злочинців на веб-додаток. В результаті було вибрано методи технічної реалізації проекту у вигляді веб-додатка.

Також проаналізувавши всі доступні методи захисту веб-додатку був використаний метод обфускації коду. Це дозволило захистити програмну реалізацію веб-додатку від можливого плагіату, а також деякі типи атак.

У проектній частини було виповнено моделювання структурно-функціональної частини, а також була побудована діаграма користування веб-додатка.

Під час практичної частини було реалізовано ескіз дизайну додатка. Розроблений дизайн був реалізований в програмному вигляді, а також була проведена демонстрація роботи веб-додатка.

СПИСОК ЛІТЕРАТУРИ

1. Смоляк В.А. Структурно-функціональне моделювання процесу : [Електронний ресурс] – Режим доступу до ресурсу: http://www.rusnauka.com/18_EN_2009/Economics/48755.doc.htm.

2. Безпечність зв’язку : [Електронний ресурс] – Режим доступу до ресурсу: https://www.kp.ru/guide/bezopasnost-saita.html.

3. Prototiping : [Електронний ресурс] – Режим доступу до ресурсу: https://www.interactiondesign.org/... Prototyping.

4. Тодд Варфел. Прототипування : [Електронний ресурс] – Режим доступу до ресурсу: www.litmir.me/br/?b=429624&p=1.

5. Прототипування сайту та безпеки : [Електронний ресурс] – Режим доступу до ресурсу: https://in-scale.ru/blog/prototipirovanie-sajta-sozdanie-instrumenty-i-programmy.

6. Особливості проектування та побудови захищених web-систем :[Електронний ресурс] – Режим доступу до ресурсу: https://igorosa.com/osobennosti-proektirovaniya-i-postroeniya-zashhishhennyx-web-sistem.

7. Що таке HTTPS: [Електронний ресурс] – Режим доступу до ресурсу: https://hostiq.ua/wiki/http-https.

8. Як захистити сайт від вірусів : [Електронний ресурс] – Режим доступу до ресурсу: https://livepage.pro/knowledge-base/protect-website-from-viruses.

9. Як захистити свяй сайт? : [Електронний ресурс] – Режим доступу до ресурсу: https://sendpulse.com/ru/blog/how-to-prevent-website-copying.

10. Безпека і SSL : [Електронний ресурс] – Режим доступу до ресурсу: https://hostiq.ua/wiki/faq/ssl-and-security.

11. Web Application Firewall (WAF) : [Електронний ресурс] – Режим доступу до ресурсу: https://smartnet.ua/ru/services/zashhita-veb-prilozhenij-i-sajtov.

12. Шевчук М. ЧТО ТАКОЕ HLD, КОМУ И ДЛЯ ЧЕГО МОЖЕТ ПОНАДОБИТЬСЯ? : [Електронний ресурс] – Режим доступу до ресурсу: https://itsource.com.ua/blog/chto-takoe-hld-i-dlya-chego-moget-ponadobitsya/.

13. IDEF0 : [Електронний ресурс] – Режим доступу до ресурсу: https://studme.org/87184/ekonomika/metodologiya_idef0.

14. Про неможливість забруднення програм : [Електронний ресурс] – Режим доступу до ресурсу: https://dl.acm.org/doi/10.1145/2160158.2160159.

15. Що таке обфускація : [Електронний ресурс] – Режим доступу до ресурсу: https://www.kv.by/archive/index2008071108.htm.

16. Безпека React : [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/ruvds/blog/467255/.

ДОДАТОК

ЛІСТИНГ ПРОГРАМНОГО КОДУ ОСНОВНИХ МОДУЛІВ ВЕБ-ДОДАТКУ

```
{
  "name": "Doors",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@babel/core": "7.9.0",
    "@svgr/webpack": "4.3.3",
    "@testing-library/jest-dom": "^4.2.4",
    "@testing-library/react": "^9.3.2",
    "@testing-library/user-event": "^7.1.2",
    "@typescript-eslint/eslint-plugin": "^2.10.0",
    "@typescript-eslint/parser": "^2.10.0",
    "axios": "^0.19.2",
    "babel-eslint": "10.1.0",
    "babel-jest": "^24.9.0",
    "babel-loader": "8.1.0",
    "babel-plugin-named-asset-import": "^0.3.6",
    "babel-preset-react-app": "^9.1.2",
    "camelcase": "^5.3.1",
    "case-sensitive-paths-webpack-plugin": "2.3.0",
    "classnames": "^2.2.6",
    "css-loader": "3.4.2",
    "dotenv": "8.2.0",
    "dotenv-expand": "5.1.0",
    "eslint": "^6.6.0",
    "eslint-config-react-app": "^5.2.1",
    "eslint-loader": "3.0.3",
    "eslint-plugin-flowtype": "4.6.0",
    "eslint-plugin-import": "2.20.1",
    "eslint-plugin-jsx-a11y": "6.2.3",
    "eslint-plugin-react": "7.19.0",
    "eslint-plugin-react-hooks": "^1.6.1",
    "file-loader": "4.3.0",
    "fs-extra": "^8.1.0",
    "html-webpack-plugin": "4.0.0-beta.11",
    "identity-obj-proxy": "3.0.0",
    "jest": "24.9.0",
    "jest-environment-jsdom-fourteen": "1.0.1",
    "jest-resolve": "24.9.0",
    "jest-watch-typeahead": "0.4.2",
    "mini-css-extract-plugin": "0.9.0",
    "node-sass": "^4.14.1",
    "optimize-css-assets-webpack-plugin": "5.0.3",
```

```
    "pnp-webpack-plugin": "1.6.4",
    "postcss-flexbugs-fixes": "4.1.0",
    "postcss-loader": "3.0.0",
    "postcss-normalize": "8.0.1",
    "postcss-preset-env": "6.7.0",
    "postcss-safe-parser": "4.0.1",
    "prop-types": "^15.7.2",
    "react": "^16.13.1",
    "react-app-polyfill": "^1.0.6",
    "react-content-loader": "^5.1.0",
    "react-dev-utils": "^10.2.1",
    "react-dom": "^16.13.1",
    "react-redux": "^7.2.0",
    "react-router-dom": "^5.2.0",
    "redux": "^4.0.5",
    "redux-thunk": "^2.3.0",
    "resolve": "1.15.0",
    "resolve-url-loader": "3.1.1",
    "sass-loader": "8.0.2",
    "semver": "6.3.0",
    "style-loader": "0.23.1",
    "terser-webpack-plugin": "2.3.5",
    "ts-pnp": "1.1.6",
    "url-loader": "2.3.0",
    "webpack": "4.42.0",
    "webpack-dev-server": "3.10.3",
    "webpack-manifest-plugin": "2.2.0",
    "workbox-webpack-plugin": "4.3.1",
    "json-server": "^0.16.1"
  },
  "scripts": {
    "start": "node scripts/start.js",
    "build": "node scripts/build.js",
    "test": "node scripts/test.js",
    "server": "node server.js",
    "json-server": "json-server --watch public/db.json --port=3001"
  },
  "eslintConfig": {
    "extends": "react-app"
  },
  "proxy": "http://localhost:3001",
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
```

```
      "last 1 safari version"
    ]
  },
  "jest": {
    "roots": [
      "<rootDir>/src"
    ],
    "collectCoverageFrom": [
      "src/**/*.{js,jsx,ts,tsx}",
      "!src/**/*.d.ts"
    ],
    "setupFiles": [
      "react-app-polyfill/jsdom"
    ],
    "setupFilesAfterEnv": [],
    "testMatch": [
      "<rootDir>/src/**/__tests__/**/*.{js,jsx,ts,tsx}",
      "<rootDir>/src/**/*.{spec,test}.{js,jsx,ts,tsx}"
    ],
    "testEnvironment": "jest-environment-jsdom-fourteen",
    "transform": {
      "^.+\\.(js|jsx|ts|tsx)$": "<rootDir>/node_modules/babel-jest",
      "^.+\\.css$": "<rootDir>/config/jest/cssTransform.js",
      "^(?!.*\\.(js|jsx|ts|tsx|css|json)$)": "<rootDir>/config/jest/fileTransform
.js"
    },
    "transformIgnorePatterns": [
      "[/\\\\]node_modules[/\\\\].+\\.(js|jsx|ts|tsx)$",
      "^.+\\.module\\.(css|sass|scss)$"
    ],
    "modulePaths": [],
    "moduleNameMapper": {
      "^react-native$": "react-native-web",
      "^.+\\.module\\.(css|sass|scss)$": "identity-obj-proxy"
    },
    "moduleFileExtensions": [
      "web.js",
      "js",
      "web.ts",
      "ts",
      "web.tsx",
      "tsx",
      "json",
      "web.jsx",
      "jsx",
      "node"
    ],
    "watchPlugins": [
      "jest-watch-typeahead/filename",
      "jest-watch-typeahead/testname"
```

```
    ]
  },
  "babel": {
    "presets": [
      "react-app"
    ]
  }
}
```

```
import React from 'react';
import { useSelector, useDispatch } from 'react-redux';

import { Categories, SortPopup, DoorBlock, DoorLoadingBlock } from '../components
';

import { setCategory, setSortBy } from '../redux/actions/filters';
import { fetchDoors } from '../redux/actions/doors';
import { addDoorToCart } from '../redux/actions/cart';

const categoryNames = ['Міжкімнатні', 'Дизайнерські', 'Вхідні', 'Спецпризначення'
];
const sortIems = [
  { name: 'популярність', type: 'popular', order: 'desc' },
  { name: 'ціна', type: 'price', order: 'desc' },
  { name: 'алфавіт', type: 'name', order: 'asc' },
];

function Home() {
  const dispatch = useDispatch();
  const items = useSelector(({ doors }) => doors.items);
  const cartItems = useSelector(({ cart }) => cart.items);
  const isLoaded = useSelector(({ doors }) => doors.isLoaded);
  const { category, sortBy } = useSelector(({ filters }) => filters);

  React.useEffect(() => {
    dispatch(fetchDoors(sortBy, category));
  }, [category, sortBy]);

  const onSelectCategory = React.useCallback((index) => {
    dispatch(setCategory(index));
  }, []);

  const onSelectSortType = React.useCallback((type) => {
    dispatch(setSortBy(type));
  }, []);

  const handleAddDoorToCart = (obj) => {
    dispatch({
      type: 'ADD_DOOR_CART',
```

```
      payload: obj,
    });
  };

  return (
    <div className="container">
      <div className="content__top">
        <Categories
          activeCategory={category}
          onClickCategory={onSelectCategory}
          items={categoryNames}
        />
        <SortPopup
          activeSortType={sortBy.type}
          items={sortIems}
          onClickSortType={onSelectSortType}
        />
      </div>
      <h2 className="content__title">Всі двері</h2>
      <div className="content__items">
        {isLoaded
          ? items.map((obj) => (
              <DoorBlock
                onClickAddDoor={handleAddDoorToCart}
                key={obj.id}
                addedCount={cartItems[obj.id] && cartItems[obj.id].items.length}
                {...obj}
              />
            ))
          : Array(12)
              .fill(0)
              .map((_, index) => <DoorLoadingBlock key={index} />)}
      </div>
    </div>
  );
}

export default Home;

import React from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { Link } from 'react-router-dom';

import cartEmptyImage from '../assets/img/empty-cart.png';
import { CartItem, Button } from '../components';
import { clearCart, removeCartItem, plusCartItem, minusCartItem } from '../redux/
actions/cart';

function Cart() {
  const dispatch = useDispatch();
```

```
  const { totalPrice, totalCount, items } = useSelector(({ cart }) => cart);

  const addedDoors = Object.keys(items).map((key) => {
    return items[key].items[0];
  });

  const onClearCart = () => {
    if (window.confirm('Ви дійсно хочете очистити корзину?')) {
      dispatch(clearCart());
    }
  };

  const onRemoveItem = (id) => {
    if (window.confirm('Ви дійсно хочете видалити?')) {
      dispatch(removeCartItem(id));
    }
  };

  const onPlusItem = (id) => {
    dispatch(plusCartItem(id));
  };

  const onMinusItem = (id) => {
    dispatch(minusCartItem(id));
  };

  const onClickOrder = () => {
    console.log('ВАШЕ ЗАМОВЛЕННЯ', items);
  };

  return (
    <div className="container container--cart">
      {totalCount ? (
        <div className="cart">
          <div className="cart__top">
            <h2 className="content__title">
              <svg
                width="18"
                height="18"
                viewBox="0 0 18 18"
                fill="none"
                xmlns="http://www.w3.org/2000/svg">
                <path
                  d="M6.33333 16.3333C7.06971 16.3333 7.66667 15.7364 7.66667 15C
7.66667 14.2636 7.06971 13.6667 6.33333 13.6667C5.59695 13.6667 5 14.2636 5 15C5
15.7364 5.59695 16.3333 6.33333 16.3333Z"
                  stroke="white"
                  strokeWidth="1.8"
                  strokeLinecap="round"
                  strokeLinejoin="round"
```

```
                />
                <path
                  d="M14.3333 16.3333C15.0697 16.3333 15.6667 15.7364 15.6667 15C
15.6667 14.2636 15.0697 13.6667 14.3333 13.6667C13.597 13.6667 13 14.2636 13 15C1
3 15.7364 13.597 16.3333 14.3333 16.3333Z"
                  stroke="white"
                  strokeWidth="1.8"
                  strokeLinecap="round"
                  strokeLinejoin="round"
                />
                <path
                  d="M4.78002 4.99999H16.3334L15.2134 10.5933C15.1524 10.9003 14.
9854 11.176 14.7417 11.3722C14.4979 11.5684 14.1929 11.6727 13.88 11.6667H6.83335
C6.50781 11.6694 6.1925 11.553 5.94689 11.3393C5.70128 11.1256 5.54233 10.8295 5.
50002 10.5067L4.48669 2.82666C4.44466 2.50615 4.28764 2.21182 4.04482 1.99844C3.8
0201 1.78505 3.48994 1.66715 3.16669 1.66666H1.66669"
                  stroke="white"
                  strokeWidth="1.8"
                  strokeLinecap="round"
                  strokeLinejoin="round"
                />
              </svg>
              Кошик
            </h2>
            <div className="cart__clear">
              <svg
                width="20"
                height="20"
                viewBox="0 0 20 20"
                fill="none"
                xmlns="http://www.w3.org/2000/svg">
                <path
                  d="M2.5 5H4.16667H17.5"
                  stroke="#B6B6B6"
                  strokeWidth="1.2"
                  strokeLinecap="round"
                  strokeLinejoin="round"
                />
                <path
                  d="M6.66663 5.00001V3.33334C6.66663 2.89131 6.84222 2.46739 7.1
5478 2.15483C7.46734 1.84227 7.89127 1.66667 8.33329 1.66667H11.6666C12.1087 1.66
667 12.5326 1.84227 12.8451 2.15483C13.1577 2.46739 13.3333 2.89131 13.3333 3.333
34V5.00001M15.8333 5.00001V16.6667C15.8333 17.1087 15.6577 17.5326 15.3451 17.845
2C15.0326 18.1577 14.6087 18.3333 14.1666 18.3333H5.83329C5.39127 18.3333 4.96734
 18.1577 4.65478 17.8452C4.34222 17.5326 4.16663 17.1087 4.16663 16.6667V5.00001H
15.8333Z"
                  stroke="#B6B6B6"
                  strokeWidth="1.2"
                  strokeLinecap="round"
                  strokeLinejoin="round"
```

```
                />
                <path
                  d="M8.33337 9.16667V14.1667"
                  stroke="#B6B6B6"
                  strokeWidth="1.2"
                  strokeLinecap="round"
                  strokeLinejoin="round"
                />
                <path
                  d="M11.6666 9.16667V14.1667"
                  stroke="#B6B6B6"
                  strokeWidth="1.2"
                  strokeLinecap="round"
                  strokeLinejoin="round"
                />
              </svg>

              <span onClick={onClearCart}>Очистити кошик</span>
            </div>
          </div>
          <div className="content__items">
            {addedDoors.map((obj) => (
              <CartItem
                key={obj.id}
                id={obj.id}
                name={obj.name}
                type={obj.type}
                size={obj.size}
                totalPrice={items[obj.id].totalPrice}
                totalCount={items[obj.id].items.length}
                onRemove={onRemoveItem}
                onMinus={onMinusItem}
                onPlus={onPlusItem}
              />
            ))}
          </div>
          <div className="cart__bottom">
            <div className="cart__bottom-details">
              <span>
                Усього дверей: <b>{totalCount} шт.</b>
              </span>
              <span>
              Сума замовлення: <b>{totalPrice} ₴</b>
              </span>
            </div>
            <div className="cart__bottom-buttons">
              <a href="/" className="button button--outline button--add go-back-
btn">
                <svg
                  width="8"
```

```
                  height="14"
                  viewBox="0 0 8 14"
                  fill="none"
                  xmlns="http://www.w3.org/2000/svg">
                  <path
                    d="M7 13L1 6.93015L6.86175 1"
                    stroke="#D3D3D3"
                    strokeWidth="1.5"
                    strokeLinecap="round"
                    strokeLinejoin="round"
                  />
                </svg>
                <Link to="/">
                  <span>Повернутися назад</span>
                </Link>
              </a>
              <Button onClick={onClickOrder} className="pay-btn">
                <span>Оплатити зараз</span>
              </Button>
            </div>
          </div>
        </div>
      ) : (
        <div className="cart cart--empty">
          <h2>
            Кошик пустий <i>😕</i>
          </h2>
          <p>
            Скоріш за все, ви не вибрали двері.
            <br />
            Для того, щоб замовити двері, перейдіть на головну сторінку.
          </p>
          <img src={cartEmptyImage} alt="Empty cart" />
          <Link to="/" className="button button--black">
            <span>Повернутися назад</span>
          </Link>
        </div>
      )}
    </div>
  );
}

export default Cart;

import React from 'react';
import PropTypes from 'prop-types';
import classNames from 'classnames';

const Button = ({ onClick, className, outline, children }) => {
  return (
```

```
    <button
      onClick={onClick}
      className={classNames('button', className, {
        'button--outline': outline,
      })}>
      {children}
    </button>
  );
};

Button.propTypes = {
  onClick: PropTypes.func,
};

export default Button;

import React from 'react';
import Button from './Button';

const CartItem = ({ id, name, type, size, totalPrice, totalCount, onRemove, onMin
us, onPlus }) => {
  const handleRemoveClick = () => {
    onRemove(id);
  };

  const handlePlusItem = () => {
    onPlus(id);
  };

  const handleMinusItem = () => {
    onMinus(id);
  };

  return (
    <div className="cart__item">
      <div className="cart__item-img">
        <img
          className="door-block__image"
          src="https://mir-s3-cdn-
cf.behance.net/project_modules/max_1200/e5ba9735176905.56ec4501b7dc3.jpg"
          alt="Двері"
        />
      </div>
      <div className="cart__item-info">
        <h3>{name}</h3>
        <p>
          {type} колір, {size} см.
        </p>
      </div>
      <div className="cart__item-count">
```

```
        <div
          onClick={handleMinusItem}
          className="button button--outline button--circle cart__item-count-
minus">
          <svg
            width="10"
            height="10"
            viewBox="0 0 10 10"
            fill="none"
            xmlns="http://www.w3.org/2000/svg">
            <path
              d="M5.92001 3.84V5.76V8.64C5.92001 9.17016 5.49017 9.6 4.96001 9.6C
4.42985 9.6 4.00001 9.17016 4.00001 8.64L4 5.76L4.00001 3.84V0.96C4.00001 0.42984
 4.42985 0 4.96001 0C5.49017 0 5.92001 0.42984 5.92001 0.96V3.84Z"
              fill="#EB5A1E"
            />
            <path
              d="M5.75998 5.92001L3.83998 5.92001L0.959977 5.92001C0.429817 5.920
01 -2.29533e-05 5.49017 -2.29301e-05 4.96001C-2.2907e-
05 4.42985 0.429817 4.00001 0.959977 4.00001L3.83998 4L5.75998 4.00001L8.63998 4.
00001C9.17014 4.00001 9.59998 4.42985 9.59998 4.96001C9.59998 5.49017 9.17014 5.9
2001 8.63998 5.92001L5.75998 5.92001Z"
              fill="#EB5A1E"
            />
          </svg>
        </div>
        <b>{totalCount}</b>
        <div
          onClick={handlePlusItem}
          className="button button--outline button--circle cart__item-count-
plus">
          <svg
            width="10"
            height="10"
            viewBox="0 0 10 10"
            fill="none"
            xmlns="http://www.w3.org/2000/svg">
            <path
              d="M5.92001 3.84V5.76V8.64C5.92001 9.17016 5.49017 9.6 4.96001 9.6C
4.42985 9.6 4.00001 9.17016 4.00001 8.64L4 5.76L4.00001 3.84V0.96C4.00001 0.42984
 4.42985 0 4.96001 0C5.49017 0 5.92001 0.42984 5.92001 0.96V3.84Z"
              fill="#EB5A1E"
            />
            <path
              d="M5.75998 5.92001L3.83998 5.92001L0.959977 5.92001C0.429817 5.920
01 -2.29533e-05 5.49017 -2.29301e-05 4.96001C-2.2907e-
05 4.42985 0.429817 4.00001 0.959977 4.00001L3.83998 4L5.75998 4.00001L8.63998 4.
00001C9.17014 4.00001 9.59998 4.42985 9.59998 4.96001C9.59998 5.49017 9.17014 5.9
2001 8.63998 5.92001L5.75998 5.92001Z"
              fill="#EB5A1E"
```

```
            />
          </svg>
        </div>
      </div>
      <div className="cart__item-price">
        <b>{totalPrice} ₴</b>
      </div>
      <div className="cart__item-remove">
        <Button onClick={handleRemoveClick} className="button--circle" outline>
          <svg
            width="10"
            height="10"
            viewBox="0 0 10 10"
            fill="none"
            xmlns="http://www.w3.org/2000/svg">
            <path
              d="M5.92001 3.84V5.76V8.64C5.92001 9.17016 5.49017 9.6 4.96001 9.6C
4.42985 9.6 4.00001 9.17016 4.00001 8.64L4 5.76L4.00001 3.84V0.96C4.00001 0.42984
 4.42985 0 4.96001 0C5.49017 0 5.92001 0.42984 5.92001 0.96V3.84Z"
              fill="#EB5A1E"
            />
            <path
              d="M5.75998 5.92001L3.83998 5.92001L0.959977 5.92001C0.429817 5.920
01 -2.29533e-05 5.49017 -2.29301e-05 4.96001C-2.2907e-
05 4.42985 0.429817 4.00001 0.959977 4.00001L3.83998 4L5.75998 4.00001L8.63998 4.
00001C9.17014 4.00001 9.59998 4.42985 9.59998 4.96001C9.59998 5.49017 9.17014 5.9
2001 8.63998 5.92001L5.75998 5.92001Z"
              fill="#EB5A1E"
            />
          </svg>
        </Button>
      </div>
    </div>
  );
};

export default CartItem;


import React from 'react';
import PropTypes from 'prop-types';

const Categories = React.memo(function Categories({ activeCategory, items, onClic
kCategory }) {
  return (
    <div className="categories">
      <ul>
        <li
          className={activeCategory === null ? 'active' : ''}
          onClick={() => onClickCategory(null)}>
```

```
          Bci
        </li>
        {items &&
          items.map((name, index) => (
            <li
              className={activeCategory === index ? 'active' : ''}
              onClick={() => onClickCategory(index)}
              key={`${name}_${index}`}>
              {name}
            </li>
          ))}
      </ul>
    </div>
  );
});

Categories.propTypes = {
  // activeCategory: PropTypes.oneOf([PropTypes.number, null]),
  items: PropTypes.arrayOf(PropTypes.string).isRequired,
  onClickCategory: PropTypes.func.isRequired,
};

Categories.defaultProps = { activeCategory: null, items: [] };

export default Categories;

import React from 'react';
import { Link } from 'react-router-dom';
import { useSelector } from 'react-redux';

import logoSvg from '../assets/img/logo.png';
import Button from './Button';

function Header() {
  const { totalPrice, totalCount } = useSelector(({ cart }) => cart);

  return (
    <div className="header">
      <div className="container">
        <Link to="/">
          <div className="header__logo">
            <img width="38" src={logoSvg} alt="logo" />
            <div>
              <h1>Двері Белорусії</h1>
              <p>Найкращі двері - поручь!</p>
            </div>
          </div>
        </Link>

        <div className="header__cart">
```

```
          <Link to="/cart">
            <Button className="button--cart">
              <span>{totalPrice} ₴</span>
              <div className="button__delimiter"></div>
              <svg
                width="18"
                height="18"
                viewBox="0 0 18 18"
                fill="none"
                xmlns="http://www.w3.org/2000/svg">
                <path
                  d="M6.33333 16.3333C7.06971 16.3333 7.66667 15.7364 7.66667 15C
7.66667 14.2636 7.06971 13.6667 6.33333 13.6667C5.59695 13.6667 5 14.2636 5 15C5
15.7364 5.59695 16.3333 6.33333 16.3333Z"
                  stroke="white"
                  strokeWidth="1.8"
                  strokeLinecap="round"
                  strokeLinejoin="round"
                />
                <path
                  d="M14.3333 16.3333C15.0697 16.3333 15.6667 15.7364 15.6667 15C
15.6667 14.2636 15.0697 13.6667 14.3333 13.6667C13.597 13.6667 13 14.2636 13 15C1
3 15.7364 13.597 16.3333 14.3333 16.3333Z"
                  stroke="white"
                  strokeWidth="1.8"
                  strokeLinecap="round"
                  strokeLinejoin="round"
                />
                <path
                  d="M4.78002 4.99999H16.3334L15.2134 10.5933C15.1524 10.9003 14.
9854 11.176 14.7417 11.3722C14.4979 11.5684 14.1929 11.6727 13.88 11.6667H6.83335
C6.50781 11.6694 6.1925 11.553 5.94689 11.3393C5.70128 11.1256 5.54233 10.8295 5.
50002 10.5067L4.48669 2.82666C4.44466 2.50615 4.28764 2.21182 4.04482 1.99844C3.8
0201 1.78505 3.48994 1.66715 3.16669 1.66666H1.66669"
                  stroke="white"
                  strokeWidth="1.8"
                  strokeLinecap="round"
                  strokeLinejoin="round"
                />
              </svg>
              <span>{totalCount}</span>
            </Button>
          </Link>
        </div>
      </div>
    </div>
  );
}

export default Header;
```

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

const SortPopup = React.memo(function SortPopup({ items, activeSortType, onClickS
ortType }) {
  const [visiblePopup, setVisiblePopup] = React.useState(false);
  const sortRef = React.useRef();
  const activeLabel = items.find((obj) => obj.type === activeSortType).name;

  const toggleVisiblePopup = () => {
    setVisiblePopup(!visiblePopup);
  };

  const handleOutsideClick = (event) => {
    const path = event.path || (event.composedPath && event.composedPath());
    if (!path.includes(sortRef.current)) {
      setVisiblePopup(false);
    }
  };

  const onSelectItem = (index) => {
    if (onClickSortType) {
      onClickSortType(index);
    }
    setVisiblePopup(false);
  };

  React.useEffect(() => {
    document.body.addEventListener('click', handleOutsideClick);
  }, []);

  return (
    <div ref={sortRef} className="sort">
      <div className="sort__label">
        <svg
          className={visiblePopup ? 'rotated' : ''}
          width="10"
          height="6"
          viewBox="0 0 10 6"
          fill="none"
          xmlns="http://www.w3.org/2000/svg">
          <path
            d="M10 5C10 5.16927 9.93815 5.31576 9.81445 5.43945C9.69075 5.56315 9
.54427 5.625 9.375 5.625H0.625C0.455729 5.625 0.309245 5.56315 0.185547 5.43945C0
.061849 5.31576 0 5.16927 0 5C0 4.83073 0.061849 4.68424 0.185547 4.56055L4.56055
 0.185547C4.68424 0.061849 4.83073 0 5 0C5.16927 0 5.31576 0.061849 5.43945 0.185
547L9.81445 4.56055C9.93815 4.68424 10 4.83073 10 5Z"
            fill="#2C2C2C"
          />
```

```
        </svg>
        <b>Сортувати по:</b>
        <span onClick={toggleVisiblePopup}>{activeLabel}</span>
      </div>
      {visiblePopup && (
        <div className="sort__popup">
          <ul>
            {items &&
              items.map((obj, index) => (
                <li
                  onClick={() => onSelectItem(obj)}
                  className={activeSortType === obj.type ? 'active' : ''}
                  key={`${obj.type}_${index}`}>
                  {obj.name}
                </li>
              ))}
          </ul>
        </div>
      )}
    </div>
  );
});

SortPopup.propTypes = {
  activeSortType: PropTypes.string.isRequired,
  items: PropTypes.arrayOf(PropTypes.object).isRequired,
  onClickSortType: PropTypes.func.isRequired,
};

SortPopup.defaultProps = {
  items: [],
};

export default SortPopup;


import React from 'react';
import ContentLoader from 'react-content-loader';

function LoadingBlock() {
  return (
    <ContentLoader
      className="door-block"
      speed={2}
      width={280}
      height={460}
      viewBox="0 0 280 460"
      backgroundColor="#f3f3f3"
      foregroundColor="#ecebeb">
      <circle cx="132" cy="142" r="115" />
```

```
      <rect x="0" y="273" rx="6" ry="6" width="280" height="26" />
      <rect x="0" y="310" rx="6" ry="6" width="280" height="84" />
      <rect x="0" y="418" rx="6" ry="6" width="91" height="31" />
      <rect x="137" y="408" rx="25" ry="25" width="140" height="46" />
    </ContentLoader>
  );
}

export default LoadingBlock;


import React from 'react';
import PropTypes from 'prop-types';
import classNames from 'classnames';
import Button from '../Button';

function DoorBlock({ id, name, imageUrl, price, types, sizes, onClickAddDoor, add
edCount }) {
  const availableTypes = ['Чорні', 'Білі'];
  const availableSizes = [60, 70, 80];

  const [activeType, setActiveType] = React.useState(types[0]);
  const [activeSize, setActiveSize] = React.useState(0);

  const onSelectType = (index) => {
    setActiveType(index);
  };

  const onSelectSize = (index) => {
    setActiveSize(index);
  };

  const onAddDoor = () => {
    const obj = {
      id,
      name,
      imageUrl,
      price,
      size: availableSizes[activeSize],
      type: availableTypes[activeType],
    };
    onClickAddDoor(obj);
  };

  return (
    <div className="door-block">
      <img className="door-block__image" src={imageUrl} alt="Двері" />
      <h4 className="door-block__title">{name}</h4>
      <div className="door-block__selector">
        <ul>
```

```
          {availableTypes.map((type, index) => (
            <li
              key={type}
              onClick={() => onSelectType(index)}
              className={classNames({
                active: activeType === index,
                disabled: !types.includes(index),
              })}>
              {type}
            </li>
          ))}
        </ul>
        <ul>
          {availableSizes.map((size, index) => (
            <li
              key={size}
              onClick={() => onSelectSize(index)}
              className={classNames({
                active: activeSize === index,
                disabled: !sizes.includes(size),
              })}>
              {size} см.
            </li>
          ))}
        </ul>
      </div>
      <div className="door-block__bottom">
        <div className="door-block__price">от {price} ₴</div>
        <Button onClick={onAddDoor} className="button--add" outline>
          <svg
            width="12"
            height="12"
            viewBox="0 0 12 12"
            fill="none"
            xmlns="http://www.w3.org/2000/svg">
            <path
              d="M10.8 4.8H7.2V1.2C7.2 0.5373 6.6627 0 6 0C5.3373 0 4.8 0.5373 4.8 1.2V4.8H1.2C0.5373 4.8 0 5.3373 0 6C0 6.6627 0.5373 7.2 1.2 7.2H4.8V10.8C4.8 11.4627 5.3373 12 6 12C6.6627 12 7.2 11.4627 7.2 10.8V7.2H10.8C11.4627 7.2 12 6.6627 12 6C12 5.3373 11.4627 4.8 10.8 4.8Z"
              fill="white"
            />
          </svg>
          <span>Додати</span>
          {addedCount && <i>{addedCount}</i>}
        </Button>
      </div>
    </div>
  );
}
```

```
DoorBlock.propTypes = {
  name: PropTypes.string,
  imageUrl: PropTypes.string,
  price: PropTypes.number,
  types: PropTypes.arrayOf(PropTypes.number),
  sizes: PropTypes.arrayOf(PropTypes.number),
  onClickAddDoor: PropTypes.func,
  addedCount: PropTypes.number,
};

DoorBlock.defaultProps = {
  name: '---',
  price: 0,
  types: [],
  sizes: [],
};

export default DoorBlock;
```