ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра інформаційних технологій

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Бакалавр»

«Розробка інтернет-магазину з продажу одягу»

(тема кваліфікаційної роботи українською мовою)

«Development of an Online Clothing Store»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання

спеціальності 122 Комп'ютерні науки

(код, назва спеціальності)

Освітня програма Комп'ютерні науки

(назва)

Коханчук Максим Леонідович

(прізвище, ім'я, по-батькові здобувача)

Керівник ст. викладач Штефан Н.З. _____

(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент к.т.н., доцент Перелигін Б.В.

(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій .

№ 1 від 09 червня 2024 р.

Завідувачка кафедри

_____ КАЗАКОВА Надія .

(підпис) (прізвище,ім'я)

Захищено на засіданні ЕК № 13 .
протокол № 19 від 20 червня 2024 р.

Оцінка добре / С / 75 .

(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК

_____ КОПИЧЕНКО Іван .

(підпис) (прізвище,ім'я)

Одеса 2024

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

CSS – Cascading Style Sheets

HTML – Hypertext Markup Language

Activity diagram – діаграм діяльності

Auth Service – сервіс авторизації

SEO – пошукова оптиміхація

Databases – бази даних

Django – бекенд фреймворк

Frontend- rowser – клієнтський рівень

JavaScript – мова програмування

MySQL – СУБД

Usability – Юзабіліті

Use Cases – варіант використання

Order Service – сервіс замовлень

Performance – продуктивність

Product Service – сервіс товарів

React.js – бібліотека

Reliability – надійність

Security – безпека

Sequence diagram – діаграм послідовності

SFA – однофакторна аутентифікація

SQL Server – СУБД

ВСТУП

Інтернет вітрини одягу – це онлайн платформи, де магазини чи бренди виставляють свої товари для продажу. Вони дозволяють покупцям переглядати, вибирати та купувати одяг, не виходячи з дому. Онлайн покупки стали актуальними з багатьох причин, які стосуються як зручності, так і економічної вигоди для покупців.

Ось основні причини, чому онлайн покупки набирають популярності:

1. Зручність: інтернет-магазини працюють цілодобово, що дозволяє робити покупки в будь-який час доби, коли це зручно для покупця. Крім того, необмежене місце розташування: Можливість здійснювати покупки з будь-якого місця, де є доступ до інтернету, незалежно від географічного положення.

2. Широкий вибір: онлайн магазини зазвичай пропонують значно ширший асортимент товарів, ніж фізичні магазини. Доступ до міжнародних брендів дає можливість купувати товари, які можуть бути недоступні в локальних магазинах.

3. Порівняння та огляди: покупці можуть читати відгуки інших користувачів, що допомагає прийняти обґрунтоване рішення. Можливість швидко порівняти ціни на різних платформах і обрати найкращу пропозицію.

4. Економія часу та грошей: не потрібно витрачати час на поїздки до магазину і стояння в чергах. Онлайн магазини часто пропонують спеціальні знижки, акції та промокоди, що робить покупки вигіднішими.

5. Персоналізований досвід: багато платформ використовують алгоритми, щоб рекомендувати товари, які можуть сподобатися покупцю, на основі їхніх попередніх покупок і переглядів. Персоналізовані пропозиції та знижки на основі покупок користувача.

6. Безпечні платежі: сучасні інтернет-магазини забезпечують високий рівень безпеки платежів, використовуючи передові технології шифрування. Багато онлайн магазинів пропонують зручні умови повернення та обміну товарів, що робить покупку менш ризикованою.

7. Інновації та технології: використання доповненої реальності (AR) для віртуальних примірок одягу, що допомагає краще уявити, як товар виглядатиме на покупцеві. Постійне вдосконалення логістики та служб доставки забезпечує швидку доставку товарів, іноді навіть у день замовлення.

Таким чином, онлайн покупки пропонують покупцям безліч переваг, які роблять їх більш привабливими та актуальними в сучасному світі. Розробка інформаційної системи для продажу одягу вимагає ретельного планування та врахування багатьох аспектів, щоб забезпечити зручний і ефективний досвід для користувачів.

Метою роботи є розробка інформаційної системи для онлайн продажу одягу, яка забезпечить зручний і швидкий процес покупки для користувачів, а також ефективне управління асортиментом і замовленнями для адміністрації магазину.

Об'єкт: процес розробки інтерактивної веб-платформи для онлайн продажу одягу, яка включає в себе:

- Фронтенд частину: користувацький інтерфейс для перегляду, вибору та покупки товарів;
- Бекенд частину: серверна частина для обробки запитів, управління базами даних, обробки платежів та управління замовленнями;
- Адміністративну панель: інтерфейс для управління асортиментом товарів, замовленнями та користувачами.

Предметом розробки є конкретні компоненти та функції інформаційної системи, що забезпечують її роботу: розробка UI, впровадження кошика для покупок, системи реєстрації, а також фільтрів для пошуку товарів.

Щоб досягти мети розробки зручного та функціонального веб-сайту для продажу одягу, потрібно виконати низку завдань:

1) планування: провести аналіз вимог та конкурентний аналіз;
2) створити план проекту з визначенням етапів, термінів;
3) розробити прототипи сторінок сайту, враховуючи зручність використання;

4) розробка фронтенду і бекенду;
5) спроектувати та реалізувати базу даних для зберігання інформації про користувачів, товари, замовлення тощо;
6) провести тестування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис об’єкту розробки

Вибір онлайн вітрин для продажу одягу має ряд важливих переваг і причин, які роблять цей підхід привабливим і ефективним у сучасному бізнес-середовищі.

Основні характеристики інтернет вітрин одягу:

1. Каталог товарів: упорядкований перелік одягу, аксесуарів та інших товарів, з можливістю фільтрації за різними критеріями (розмір, колір, бренд, ціна тощо).
2. Фотографії та описи: кожен товар має якісні фотографії з різних ракурсів, а також детальні описи, що включають матеріали, розміри, інструкції з догляду тощо.
3. Система пошуку та фільтрів: покупці можуть швидко знайти потрібний товар за допомогою пошукової системи та різноманітних фільтрів.
4. Відгуки та рейтинги: можливість залишати відгуки про товари, що допомагає іншим покупцям робити обґрунтовані вибори.
5. Корзина та система замовлення: інтуїтивно зрозумілий інтерфейс для додавання товарів у корзину, оформлення замовлення та вибору способу оплати та доставки.
6. Програми лояльності та акції: пропозиції, знижки та бонуси для постійних клієнтів.
7. Система підтримки клієнтів: чат, електронна пошта чи телефонна підтримка для вирішення питань покупців.

Переваги інтернет вітрин:

- зручність: можливість робити покупки з будь-якого місця та в будь-який час;
- широкий асортимент: доступ до товарів, які можуть бути відсутніми в локальних магазинах;

- порівняння цін та продуктів: легко порівнювати різні варіанти та обирати найкращий;
- відгуки інших покупців: дозволяють дізнатися більше про якість товару та досвід інших клієнтів.

Боротьба з конкуренцією на онлайн ринку вимагає стратегічного підходу та постійного вдосконалення. Ось кілька ефективних стратегій, які можуть допомогти виділитися серед конкурентів:

1. Унікальна Торгова Пропозиція (УТП): слідм визначити, що робить продукт чи послугу унікальними. Це може бути особлива якість, ексклюзивний асортимент, екологічність або інноваційні рішення.
2. Висока якість обслуговування клієнтів, а саме підтримка клієнтів через різні канали (чат, електронна пошта, телефон) та післяпродажне обслуговування, тобто, пропонуючи допомогу з використанням товару або швидке вирішення проблем.
3. Оптимізація сайту та користувацького досвіду:
   - зручний інтерфейс зробіть сайт інтуїтивно зрозумілим і легким у навігації;
   - мобільна версія може забезпечити відмінний досвід на мобільних пристроях, оскільки багато покупок здійснюються через смартфони;
   - швидкість завантаження, щоб знизити кількість відмов через повільне завантаження.
4. SEO та контент-маркетинг.
5. Програми лояльності:
   - бонусні програми, які заохочують повторні покупки;
   - персоналізовані пропозиції: використання даних про покупців для створення персоналізованих пропозицій та знижок.
6. Аналіз конкуренції та ринку:
   - моніторинг конкурентів;

- SWOT-аналіз (сильні та слабкі сторони, можливості та загрози) для оцінки бізнесу та прийняття стратегічних рішень.

7. Інновації та адаптація:
   - нові технології, які можуть поліпшити користувацький досвід або ефективність бізнесу;
   - гнучкість, щоб бути готовими адаптуватися до змін ринку та нових тенденцій.

1.2 Порівнянний аналіз існуючих аналогів

Розробка інформаційної системи у вигляді веб-сайту для продажу одягу вимагає ретельного планування та врахування багатьох аспектів, щоб забезпечити зручний і ефективний досвід для користувачів. Перший крок, який допоможе у цьому процесі – це аналіз існуючих рішень, знання сильних та слабких сторін конкурентів, що може стати основою для постановки задач до об’єкту розробки.

Першим ,аналогом було обрано відомий магазин «Karmaloop» (рис. 1.1).

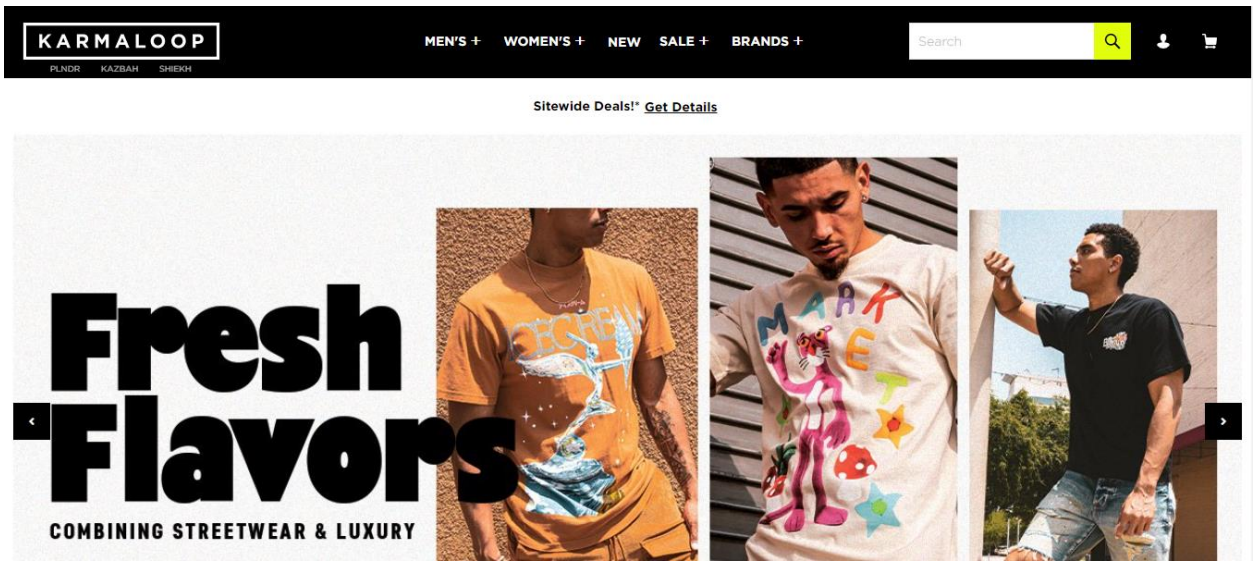

Рисунок 1.1 – Головна сторінка магазину «Karmaloop»

«Karmaloop», відомий своїм широким асортиментом вуличного одягу, пропонує такі бренди, як 10 Deep, Adidas, Billionaire Boys Club тощо, що робить його улюбленим місцем для любителів міської моди. Є кілька причин, які сприяють його видатному статусу на ринку вуличного одягу та міської моди:

1. Великий вибір брендів.

«Karmaloop» пропонує широкий вибір брендів вуличного одягу, зокрема таких популярних імен, як «Adidas», «Nike», «Billionaire Boys Club» і «10 Deep». Цей широкий асортимент гарантує, що клієнти зможуть знайти різноманітні стилі та унікальні речі, які задовольнять різні смаки в культурі вуличного одягу.

2. Новаторська роздрібна мережа вуличного одягу.

Як один із перших онлайн-магазинів, який зосередився виключно на вуличному одязі, «Karmaloop» зарекомендував себе як піонера на ринку. Цей ранній вихід дозволив їм створити потужну, лояльну клієнтську базу та закріпитися в галузі до появи багатьох конкурентів.

3. Молодіжний маркетинг.

Компанія «Karmaloop» ефективно націлилася на молодіжний ринок за допомогою своїх маркетингових стратегій, використовуючи соціальні медіа, співпрацюючи з впливовими особами та цікавий контент. Їхній підхід резонує з молодшою аудиторією, яка цінує модну міську моду та культуру.

4. Унікальні колаборації та ексклюзиви.

«Karmaloop» часто співпрацює з брендами вуличного одягу, щоб пропонувати ексклюзивні речі та випуски обмеженої кількості. Ця співпраця викликає галас і приваблює клієнтів, які шукають унікальні речі, які важко знайти.

5. Інтеграція громади та культури.

«Karmaloop» побудував спільноту навколо культури вуличного одягу, інтегруючи елементи музики, мистецтва та міського стилю життя у свій бренд.

Цей культурний зв’язок зміцнює лояльність клієнтів і позиціонує «Karmaloop» як не просто магазин роздрібної торгівлі, а бренд стилю життя.

6. Інноваційна онлайн-присутність.

Їх веб-сайт і платформа електронної комерції розроблені, щоб запропонувати безперебійний досвід покупок із детальним описом продукту, високоякісними зображеннями та простим у навігації інтерфейсом. Цей фокус на досвіді користувачів допомагає залучати та утримувати клієнтів.

7. Часті розпродажі та знижки.

У «Karmaloop» часто проводяться розпродажі, знижки та рекламні заходи, що робить його привабливим варіантом для покупців, які розуміються на бюджеті та хочуть отримати доступ до високоякісного вуличного одягу.

Ці фактори разом роблять «Karmaloop» видатним ім’ям у індустрії роздрібної торгівлі вуличним одягом, звертаючись до широкої аудиторії та зберігаючи свою репутацію провідного місця для ентузіастів міської моди.

Наступний аналог «END.» (рис. 1.2). «END» славиться одним із найкращих виборів брендів в Інтернеті. У них не тільки представлені провідні бренди вуличного одягу, такі як «FUCT», «Maharishi» та натхненник «JAPAN».

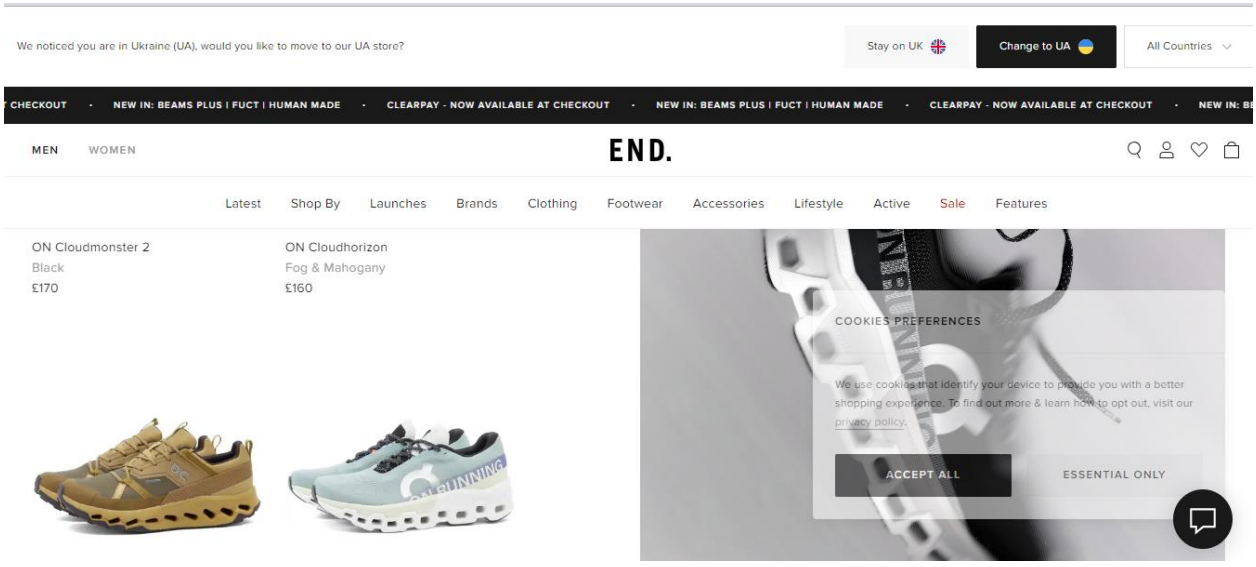

Рисунок 1.2 – Головна сторінка сайту «END.»

Тут також представлені найкращі дизайнери розкоші, зокрема «Thom Browne», «Dries Van Noten» і «Givenchy». END також встановлює високі стандарти, створюючи лукбуки для своїх майбутніх випусків. Магазин спеціалізується на вуличному одягу високого класу та сучасній моді, має кілька переваг, які відрізняють його від конкурентів:

1. Підібраний вибір преміальних брендів.

«END.» пропонує ретельно відібраний вибір брендів преміум-класу, включаючи розкішні та нішеві дизайнери, як-от Stone Island, Off-White, Balenciaga та Common Projects. Цей продуманий підхід гарантує, що клієнти мають доступ до ексклюзивних високоякісних продуктів, які часто важко знайти деінде.

2. Ексклюзивні релізи та співпраця.

«END.». часто співпрацює з провідними брендами, щоб випускати ексклюзивні товари та співпрацювати з обмеженими тиражами. Ці ексклюзивні пропозиції приваблюють ентузіастів моди та колекціонерів, які шукають унікальні речі, які не можна знайти в інших магазинах.

3. Високоякісний досвід електронної комерції.

Кінець. веб-сайт розроблений таким чином, щоб забезпечити преміальний досвід покупок із зрозумілим сучасним інтерфейсом, зображеннями високої роздільної здатності, детальним описом продукту та зручною системою навігації. Така увага до взаємодії з користувачем робить покупки на їхньому сайті приємними та ефективними.

4. Глобальне охоплення з місцевим обслуговуванням.

«END.» має сильну міжнародну присутність із глобальними варіантами доставки. Вони також пропонують місцеві послуги через свої фізичні магазини у Великій Британії, які надають клієнтам можливість відчути бренд особисто. Таке поєднання глобального охоплення та місцевого обслуговування покращує підтримку клієнтів і доступність.

5. Комплексне обслуговування клієнтів.

6. Різноманітний асортимент продукції.

Крім одягу, «END.» також пропонує широкий асортимент аксесуарів, взуття та товарів для стилю життя. Цей різноманітний асортимент продукції дозволяє клієнтам купувати різноманітні модні та життєві потреби в одному місці, що робить його зручним місцем для комплексного шопінгу моди.

7. Контент і редакція

«END.» створює високоякісний редакційний контент, зокрема лукбуки, елементи бренду та посібники зі стилю. Цей вміст не лише надихає моду, але й навчає клієнтів про останні тенденції та історії брендів, сприяючи глибшому зв’язку з аудиторією.

8. Програми лояльності та стимули.

Зосередившись на цих ключових сферах, «END.» виділяється на конкурентному онлайн-ринку моди, пропонуючи чудовий досвід покупок і ексклюзивні продукти, які приваблюють вимогливих ентузіастів моди.

Отже, крім особливостей аналогів, слід відмітити окремо дизайн он-лайн вітрини, сучасні рішення, кольорова гамма та навігація – все робить топових представників магазинів популярними.

1.3 Функціональні вимоги до інформаційної системи

Функціональні вимоги до інтернет-магазину описують конкретні функції та можливості, які система повинна мати для задоволення потреб користувачів і бізнесу. Вони включають детальний опис процесів і дій, які повинна виконувати система. Функціональні вимоги можна сприймати як функції, які користувач може спостерігати і використовувати. Вони відрізняються від нефункціональних вимог, що визначають, як система повинна працювати внутрішньо (наприклад, продуктивність, безпека тощо).

Функціональні вимоги складаються з двох частин: функцій та поведінки. Функція описує, що саме виконує система (наприклад, «розрахувати податок з продажу»). Поведінка вказує, як система реалізує цю функцію (наприклад,

«Система розраховує податок із продажу, помноживши ціну покупки на ставку податку»). При формуванні функціональних вимог важливо, щоб вони були конкретними, вимірними, досяжними, релевантними та обмеженими у часі [1].

Варіанти використання (Use Cases) є інструментом для опису функціональних вимог до програмного продукту. Вони показують, як користувачі взаємодіють з системою та як система повинна реагувати на їх дії. Варіанти використання описують взаємодію між основним актором, системою та вторинними акторами, необхідними для досягнення мети основного актора.

Зазвичай варіанти використання ініціюються первинним актором, але в деяких випадках їх може запускати інша система, зовнішня подія або таймер. Варіанти використання пишуться з точки зору актора і уникають опису внутрішньої роботи системи [2]. На рисунку 1.3 представлено діаграму варіантів використання для актора-Користувача. Опис основних варіантів використання (ВВ) представлено нижче.

ВВ №1 «Реєстрація користувача».

Основний актор: Новий користувач.

Передумови: Користувач не зареєстрований у системі.

Основний сценарій:

1. Користувач переходить на сторінку реєстрації.
2. Користувач вводить необхідні дані (ім'я, прізвище, електронну пошту, пароль).
3. Система надсилає підтвердження на електронну пошту.
4. Користувач підтверджує реєстрацію через посилання в електронному листі.
5. Система активує обліковий запис користувача.

Альтернативні сценарії: Користувач вводить некоректні дані (наприклад, неправильний формат електронної пошти). Система відображає відповідне повідомлення про помилку.

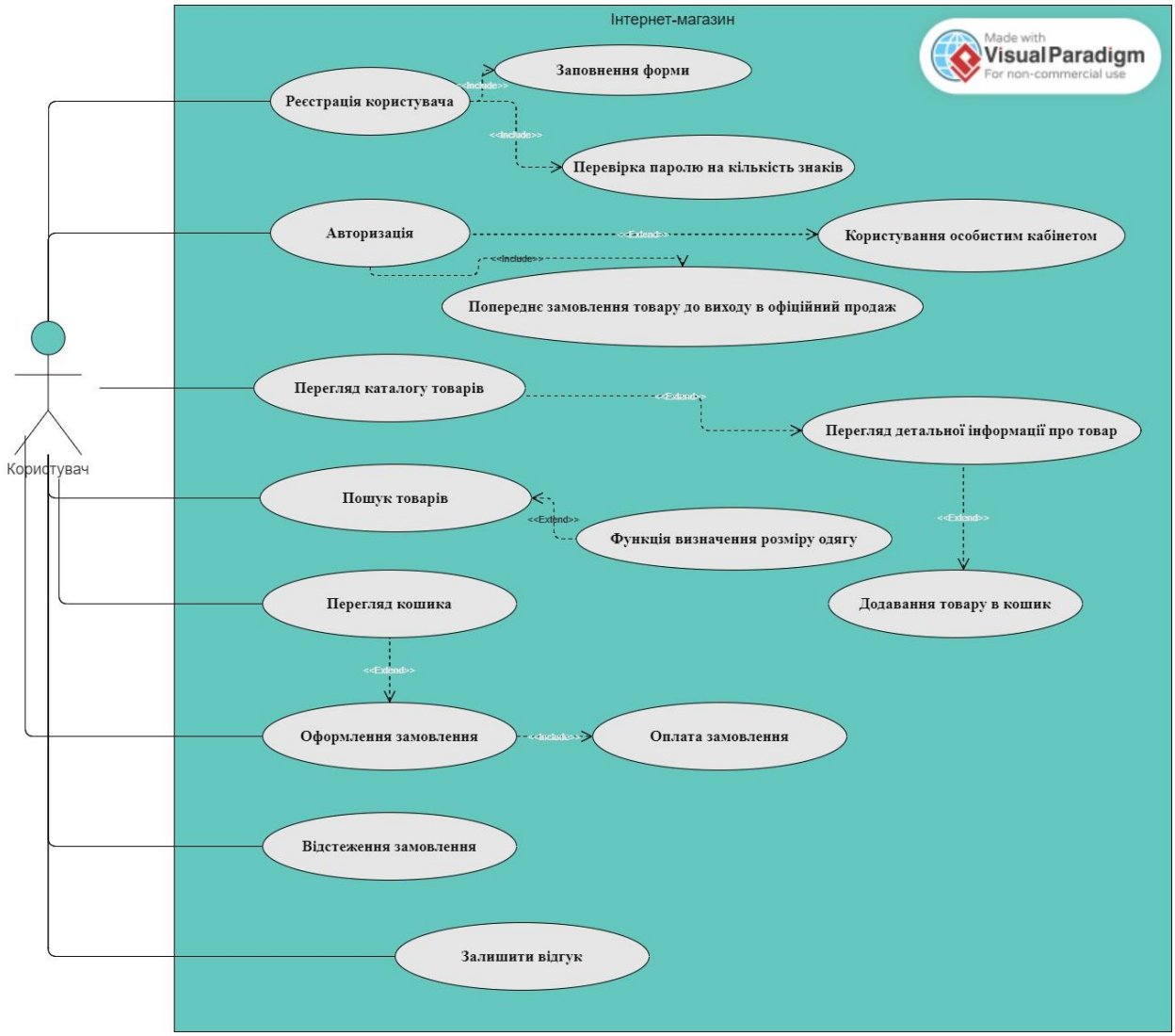

Рисунок 1.3 – Діаграма варіантів використання

ВВ №2 «Авторизація користувача»

Основний актор: Зареєстрований користувач.

Передумови: Користувач має активний обліковий запис.

Основний сценарій:

1. Користувач переходить на сторінку авторизації.
2. Користувач вводить електронну пошту та пароль.
3. Система перевіряє дані користувача.
4. Система дозволяє користувачеві увійти в систему.

Альтернативні сценарії: Користувач вводить неправильний пароль. Система відображає повідомлення про помилку і пропонує відновити пароль.

ВВ №3 «Перегляд каталогу товарів».

Основний актор: Користувач

Передумови: Товари додані в каталог.

Основний сценарій:

1. Користувач переходить до каталогу товарів.
2. Користувач фільтрує товари за категоріями, ціною, брендом тощо.
3. Користувач сортує товари за популярністю, новизною, ціною.
4. Користувач переглядає список товарів і вибирає конкретний товар для детальнішого перегляду.

Альтернативні сценарії: Система повідомляє про помилку щодо роботи функції відкриття сторінки.

ВВ №4 «Пошук товарів».

Основний актор: Користувач

Передумови: Товари додані в каталог.

Основний сценарій:

1. Користувач вводить ключове слово у поле пошуку.
2. Система відображає результати пошуку відповідно до введених ключових слів.
3. Користувач переглядає результати та вибирає потрібний товар.

Альтернативні сценарії: Система повідомляє Користувача щодо помилки під час пошуку необхідного товару та пропонує ще раз запустити цю функцію.

ВВ №5 «Перегляд детальної інформації про товар».

Основний актор: Користувач.

Передумови: Товари додані в каталог.

Основний сценарій:

1. Користувач вибирає товар зі списку.

2. Система відображає сторінку з детальною інформацією про товар (опис, ціна, фотографії, відгуки).
3. Користувач переглядає детальну інформацію.

Альтернативні сценарії: Система повідомляє Користувача щодо помилки під час завантаження обраної сторінки.

ВВ №6 «Додавання товару в кошик».

Основний актор: Користувач.

Передумови: Користувач переглядає товар.

Основний сценарій:

1. Користувач вибирає товар і його характеристики (наприклад, розмір, колір).
2. Користувач додає товар до кошика.
3. Система оновлює кошик та відображає повідомлення про успішне додавання товару.

Альтернативні сценарії: Система повідомляє Користувача щодо помилки додавання товару. Користувач повторює спробу.

ВВ №7 «Перегляд кошика»

Основний актор: Користувач.

Передумови: Товари додані в кошик.

Основний сценарій:

1. Користувач переходить до кошика.
2. Система відображає список товарів у кошику з можливістю змінювати кількість або видаляти товари.
3. Користувач переглядає та редагує вміст кошика.

Альтернативні сценарії: Система відображає порожній кошик покупця та пропонує повернутись до покупок.

ВВ №8 «Оформлення замовлення».

Основний актор: Користувач.

Передумови: Товари додані в кошик.

Основний сценарій:

1. Користувач переходить до оформлення замовлення.
2. Користувач вводить дані для доставки та вибирає спосіб оплати.
3. Система перевіряє введені дані.
4. Користувач підтверджує замовлення.
5. Система обробляє замовлення та надсилає підтвердження на електронну пошту користувача.

Альтернативні сценарії: Користувач відкликає процес оформлення замовлення та повертається до покупок.

ВВ №9 «Оплата замовлення».

Основний актор: Користувач.

Передумови: Користувач оформлює замовлення.

Основний сценарій:

1. Користувач вибирає спосіб оплати.
2. Користувач вводить платіжні дані.
3. Система перевіряє та обробляє платіж.
4. Система підтверджує успішну оплату та надсилає повідомлення користувачу.

Альтернативні сценарії: Система повідомляє щодо неможливості проведення фінансової операції.

ВВ №10 «Відстеження замовлення».

Основний актор: Користувач.

Передумови: Замовлення опрацьоване та відправлене.

Основний сценарій:

1. Користувач переходить до історії замовлень у своєму профілі.
2. Користувач вибирає замовлення для відстеження.
3. Система відображає статус доставки та трекінг-номер.

Альтернативні сценарії: Система повідомляє щодо неможливості відображення інформації.

ВВ №11 «Попереднє замовлення товару до виходу в офіційний продаж»

Основний актор: Зареєстрований користувач.

Передумови:

- Користувач повинен бути зареєстрованим та авторизованим у системі.
- Товар повинен мати статус "Доступний для попереднього замовлення".

Основний сценарій:

1. Користувач переглядає каталог товарів.
2. Користувач обирає товар, доступний для попереднього замовлення.
3. Користувач додає товар до кошика.
4. Користувач переходить до оформлення замовлення.
5. Система пропонує обрати спосіб оплати та доставки.
6. Користувач вводить платіжні та контактні дані.
7. Система зберігає замовлення та надсилає підтвердження на електронну пошту користувача.
8. Після офіційного виходу товару в продаж, система автоматично обробляє замовлення та відправляє товар користувачеві.

Альтернативні сценарії:

- товар стає недоступним до моменту завершення попереднього замовлення. Система повідомляє користувача про зміни і пропонує альтернативні варіанти.
- Користувач змінює рішення та видаляє товар з кошика перед завершенням замовлення.

Варіанти використання допомагають зрозуміти, як користувачі взаємодіють з системою та забезпечують, що система відповідає їхнім потребам. Вони описують послідовність дій, необхідних для виконання певних завдань, що дозволяє розробникам і дизайнерам створювати інтерфейси та функції, які покращують користувацький досвід.

1.4 Нефункціональні вимоги до проекту

Нефункціональні вимоги – це технічні параметри веб-сайту, які впливають на параметри якості та продуктивності, які є основою взаємодії з користувачем. Наприклад, це стосується швидкості завантаження запитуваної сторінки.

Нефункціональні вимоги для інтернет-магазину мають декілька складових:

1. Продуктивність:
   - час відгуку: система повинна реагувати на запити користувачів протягом 2 секунд для більшості операцій;
   - час завантаження сторінки: сторінки повинні завантажуватися протягом 3 секунд або менше;
   - обробка транзакцій: час обробки платіжної транзакції не повинен перевищувати 5 секунд.
2. Масштабованість.
3. Надійність.
4. Юзабіліті:
   - інтерфейс повинен бути інтуїтивно зрозумілим і доступним для користувачів різного віку та навичок;
   - всі основні функції повинні бути доступні не більше ніж у три кліки.
5. Сумісність: система повинна коректно працювати у всіх основних браузерах та повинна бути підтримка для мобільних пристроїв (адаптивний дизайн для різних розмірів екранів).
6. Підтримка та обслуговування

Нефункціональні вимоги є критично важливими для забезпечення стабільної, безпечної та ефективної роботи інтернет-магазину. Вони гарантують, що система буде відповідати очікуванням користувачів і бізнесу щодо якості обслуговування та технічних характеристик [3].

2 ОПИС ЕТАПУ ПРОЕКТУВАННЯ

Люди часто плутають вимоги до проекту з цілями та завданнями. Різниця між цими термінами досить проста: цілі та завдання описують те, що саме хочеться досягти, тоді як вимоги пов’язані з тим, як розробник збирається досягти цих цілей. Іншими словами, вимога описує обов’язкові функції та завдання, щоб переконатися, що ваша мета чи мета досягнута, а ваш проект буде успішним.

Якщо немає задокументованого процесу чи методу того, як щось робиться, буде важче відтворити зусилля розробника чи визначити ризики, перш ніж вони стануть проблемами [4].

2.1 Розробка плану проекту

Розробка інтернет-магазину одягу є комплексним проектом, що включає кілька етапів від планування до впровадження та підтримки. Нижче наведено детальний план розробки такого проекту.

1. Підготовчий етап.

1.1. Аналіз вимог: визначити цільову аудиторію, зібрати та проаналізувати бізнес-вимоги, визначити функціональні та нефункціональні вимоги, а також провести конкурентний.

1.2. Проектування концепції:

- розробити бачення та мету проекту;
- сформулювати цілі та завдання проекту.

2. Проектування.

2.1. Архітектура системи:

- визначити архітектурний стиль (монолітна, мікросервісна архітектура);
- проектувати базу даних.

2.2. Прототипування та UX/UI дизайн:

- створити прототипи та вайрфрейми;
- розробити дизайн інтерфейсу користувача;
- провести тестування з користувачами для збирання зворотного зв’язку.

3. Розробка.

3.1. Розробка фронтенду:

- вибір фреймворків та технологій (наприклад, react, angular);
- розробка інтерфейсів користувача;
- реалізація адаптивного дизайну для мобільних пристроїв.

3.2. Розробка бекенду:

- вибір фреймворків та технологій;
- реалізація логіки бізнес-процесів;
- інтеграція з базою даних;
- реалізація системи обробки платежів.

3.3. Інтеграція сторонніх сервісів:

- інтеграція з платіжними системами;
- інтеграція системи визначення розміру одягу;
- інтеграція системи знижок та бонусів.

4. Тестування.

5. Впровадження, підтримка та обслуговування.

Розробка інтернет-магазину одягу включає кілька ключових етапів, кожен з яких вимагає ретельного планування та виконання. Від початкового аналізу та проектування до впровадження та підтримки – всі етапи повинні бути скоординовані для досягнення успіху проекту.

Діаграма Ганта – це ефективний інструмент для візуалізації розкладу завдань у проекті та контролю за їх виконанням у часовому аспекті. У такому проекті вона може допомогти з багатьма аспектами:

1. Візуалізація завдань: діаграма Ганта надає чітке уявлення про всі завдання, що входять до проекту, та їх часові рамки. Це допомагає

учасникам проекту краще розуміти, які кроки необхідно виконати для досягнення мети.

2. Планування та графік: дозволяє розподілити завдання між учасниками проекту, встановити терміни виконання кожного етапу та оцінити загальний час, необхідний для завершення проекту.
3. Контроль за термінами: за допомогою діаграми Ганта можна легко відслідковувати прогрес виконання завдань та порівнювати його з планом. Якщо виникають затримки або перебої, їх можна виявити та вирішити швидко, щоб уникнути відставання від графіку.
4. Діаграма Ганта може служити інструментом комунікації між учасниками проекту, клієнтами та зацікавленими сторонами. Вона надає зручний спосіб відображення прогресу та статусу проекту.
5. Діаграма Ганта може бути використана для розподілу ресурсів, таких як людські ресурси, матеріали та бюджет, між різними завданнями проекту [5].

Загалом, діаграма Ганта допомагає керувати проектом, забезпечуючи структуроване планування, ефективний моніторинг та контроль за часом та ресурсами. Отже, для побудови діаграми Ганта спершу слід розробити структуру послідовності робіт (табл. 2.1).

Таблиця 2.1 – Структура декомпозиції робіт

№	Назва задачі	Початок	Завершення
1	Специфікація вимог	01.02.2024	13.02.2024
1.1	Опис предметної області	01.02.2024	06.02.2024
1.2	Функціональні вимоги	07.02.2024	12.02.2024
1.3	Нефункціональні вимоги	13.02.2024	15.02.2024
2	Проектування ІС	14.02.2024	26.03.2024
2.1	Визначення архітектури	14.02.2024	19.02.2024
2.2	Розробка UML-діаграми	20.02.2024	26.03.2024

Продовження таблиці 2.1

3	Розробка фронтенду	05.03.2024	20.03.2024
4	Розробка бекенду	21.03.2024	06.05.2024
4.1	Створення архітектури	21.03.2024	26.03.2024
4.2	Реалізація бізнес-логіки	27.03.2024	11.04.2024
4.3	Інтеграція з БД	12.04.2024	25.04.2024
4.4	Тестування окремих модулів та компонентів	26.04.2024	06.05.2024
5	Тестування	07.05.2024	24.05.2024
5.1	Складання тесткейсів	07.05.2024	10.05.2024
5.2	Виправлення помилок	13.05.2024	24.05.2024
6	Впровадження та підтримка (оформлення пояснювальної записки)	27.05.2024	10.06.2024

Діаграму Ганта для ПЗ представлено на рис.2.1.

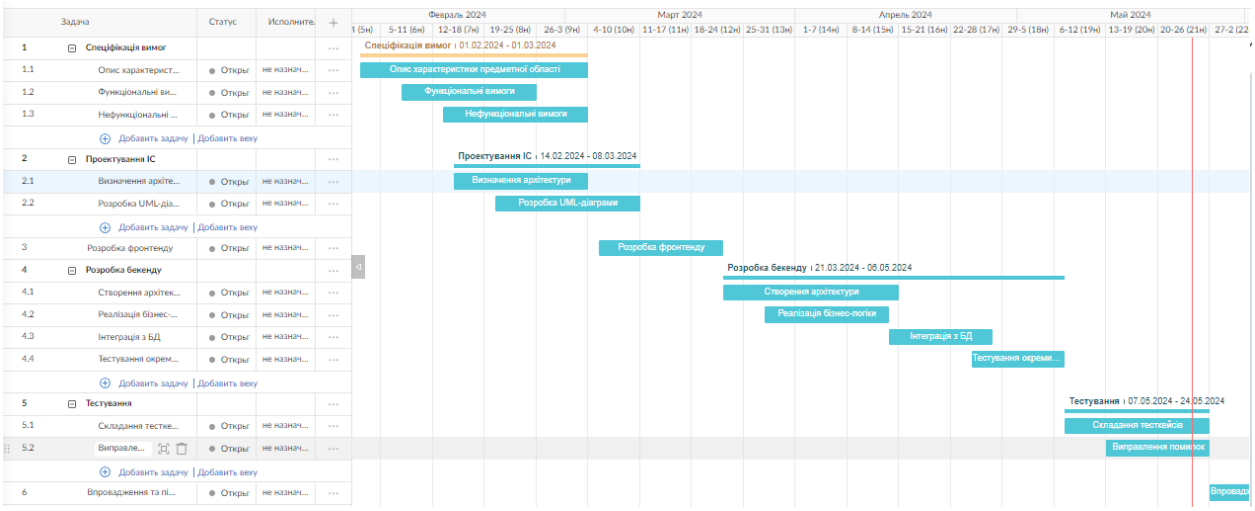

Рисунок 2.1 – Діаграма Ганта

2.2 Вибір архітектури

Для розробки онлайн магазину одягу варто розглянути архітектуру, яка буде забезпечувати масштабованість, безпеку, ефективність та зручність для користувачів. Ось деякі архітектурні варіанти, які можна розглянути:

1. Мікросервісна архітектура (рис. 2.2): полягає в розбитті функціональності магазину на невеликі сервіси, які працюють незалежно один від одного. Кожен сервіс може відповідати за конкретний аспект, наприклад, каталог товарів, обробку замовлень, оплату, аутентифікацію тощо. Це забезпечить гнучкість в розробці та підтримці, а також швидке масштабування окремих компонентів [6].

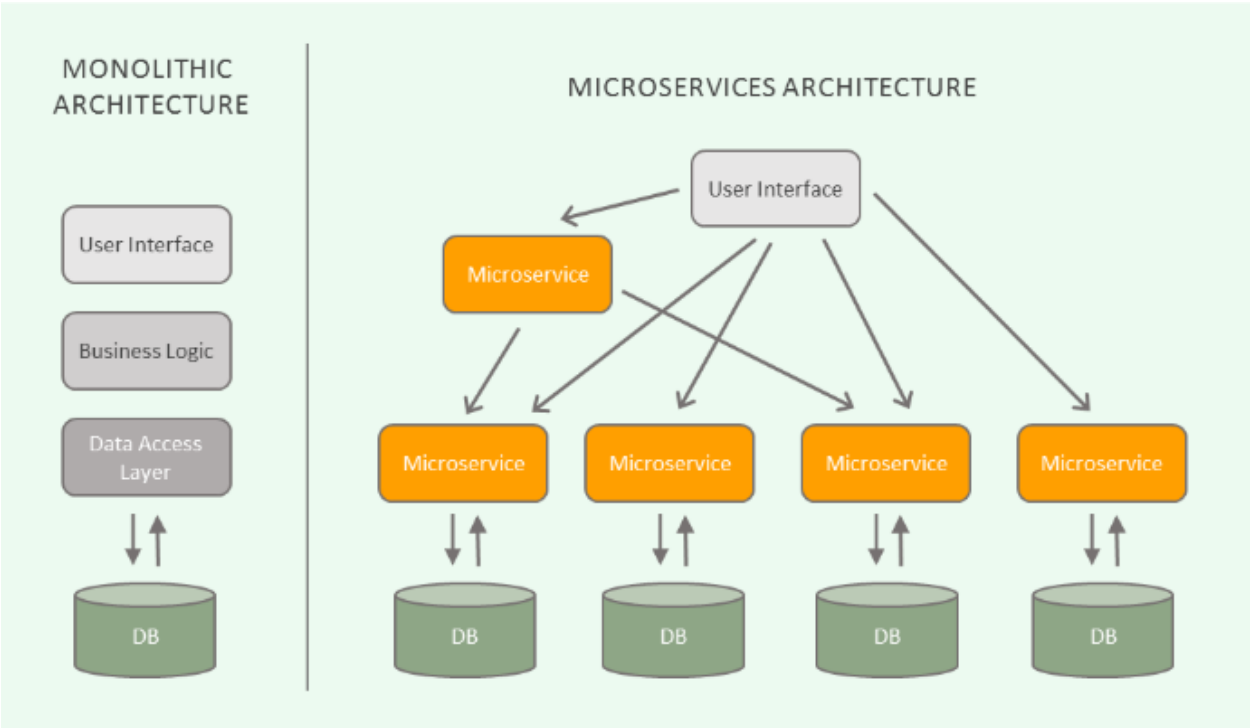

Рисунок 2.2 –Порівняння монолітної та мікросервісної архітектури

2. Орієнтована на події (Event-Driven) архітектура: у такій архітектурі система відповідає на події, які відбуваються в магазині, такі як замовлення, оплати, зміни статусів товарів тощо. Це дозволяє реагувати на

події в реальному часі та підтримувати асинхронний обмін даними між компонентами системи.

3. Хмарна архітектура: використання хмарних сервісів, таких як AWS, Google Cloud або Azure, може забезпечити високу доступність, масштабованість та безпеку магазину. Можна використовувати такі сервіси, як хмарні бази даних, обчислювальні ресурси, CDN, щоб оптимізувати продуктивність вашого магазину та зменшити витрати на інфраструктуру.

Для дипломної роботи обрану мікросервісну архітектуру. Мікросервісна архітектура має кілька значних переваг, особливо у контексті розробки онлайн магазину одягу:

1. Гнучкість та швидкість в розробці та впровадженні: мікросервіси дозволяють розбити великі, складні системи на менші, більш керовані компоненти. Це дозволяє командам розробників працювати над окремими сервісами паралельно, що сприяє швидкішому впровадженню та постійному вдосконаленню.
2. Масштабованість: завдяки мікросервісній архітектурі можна масштабувати тільки ті компоненти системи, які потребують більшої потужності або ресурсів. Наприклад, якщо магазин виявляє збільшений попит на розділ з оплатою, можна масштабувати лише сервіс, що відповідає за цей функціонал, зберігаючи ефективне використання ресурсів.
3. Незалежність та резилієнтність: кожен мікросервіс може працювати незалежно від інших, що дозволяє знизити вплив помилок або відмов окремих сервісів на загальну систему. Якщо один сервіс зазнає проблем, це не призведе до повного відмови системи, а лише до обмеження функціональності, якою він відповідає.
4. Технологічна гетерогенність: кожен мікросервіс може бути розроблений та підтримуваний з використанням різних технологій, що дозволяє використовувати найкращі практики та інструменти для кожного конкретного завдання [6].

5. Легка масштабованість команд: команди розробників можуть бути розподілені на невеликі групи, кожна з яких відповідає за певний сервіс. Це сприяє зменшенню завантаження та полегшує комунікацію між учасниками команди.

Загалом, мікросервісна архітектура дозволяє створювати гнучкі, масштабовані та надійні системи, які відповідають потребам сучасного онлайн бізнесу, такого як магазини одягу.

2.3 Проектування моделі даних

Проектування моделі даних для інформаційної системи з продажу товарів включає створення логічної структури бази даних, яка відображає ключові сутності та їх взаємозв'язки. Ось основні сутності та можливі зв'язки між ними:

1 Таблиця «Користувачів» (Users):

- ідентифікатор користувача;
- ім'я;
- прізвище;
- електронна адреса;
- адреса доставки;
- інші персональні дані, такі як телефон та інше.

2 Таблиця «Товарів» (Products):

- ідентифікатор товару;
- назва товару;
- опис товару;
- категорія товару (наприклад, жіночий одяг, чоловічий одяг, аксесуари тощо);
- ціна;
- зображення товару;
- інші характеристики товару.

3 Таблиця «Замовлень» (Orders):

- ідентифікатор замовлення;
- ідентифікатор користувача (зовнішній ключ до таблиці користувачів);
- дата та час замовлення;
- статус замовлення (нове, в обробці, відправлено тощо);

4 Таблиця «Елементів замовлення» (Order Items):

- ідентифікатор елемента замовлення;
- ідентифікатор замовлення (зовнішній ключ до таблиці замовлень;)
- ідентифікатор товару (зовнішній ключ до таблиці товарів);
- кількість;
- ціна за одиницю товару.

5 Таблиця «Розмірів» (Sizes):

- ідентифікатор розміру;
- назва розміру;
- категорія розміру (жіночий одяг, чоловічий одяг, дитячий одяг тощо);
- інші характеристики розміру.

6 Таблиця «Брендів» (Brands):

- ідентифікатор бренду;
- назва бренду;
- інформація про бренд (опис, логотип тощо).

7 Таблиця «Вибору Розміру» (Size Selection):

- ідентифікатор вибору розміру;
- ідентифікатор користувача (зовнішній ключ до таблиці користувачів);
- ідентифікатор бренду (зовнішній ключ до таблиці брендів);
- ідентифікатор розміру (зовнішній ключ до таблиці розмірів).

8 Таблиця «Історія замовлень» (Order History):

- ідентифікатор запису історії замовлень (primary key);

- ідентифікатор замовлення (foreign key);
- дата та час події (створення, відправлення, отримання тощо);
- опис події (зміна статусу, додавання коментаря тощо).

8 Таблиця «Відгуки» (Reviews):

- ідентифікатор відгуку (primary key);
- ідентифікатор товару (foreign key);
- ідентифікатор користувача (foreign key);
- оцінка (наприклад, від 1 до 5 зірок);
- текст відгуку;
- дата та час додавання відгуку.

Зв'язки між таблицями:

- Кожен користувач може зробити багато замовлень, тому в таблиці Замовлень є зовнішній ключ до таблиці «Користувачів».
- Кожен товар може бути частиною багатьох замовлень, тому таблиця Елементів замовлення має зовнішній ключ до таблиці «Товарів».
- Товар може мати різні розміри, тому використовується таблиця зв'язку між «Товарами» та «Розмірами».
- Кожен товар належить до певної категорії, тому таблиця «Товарів «містить зовнішній ключ до таблиці «Категорій».
- Кожен товар може належати до певної марки, тому використовується зовнішній ключ до таблиці «Брендів» в таблиці «Товарів».
- Кожен замовлення може мати багато записів історії замовлень, тому в таблиці «Історії замовлень» є зовнішній ключ до таблиці «Замовлень».
- Кожен товар може мати багато відгуків, тому в таблиці «Відгуків» є зовнішній ключ до таблиці «Товарів».
- Кожен користувач може залишати багато відгуків, тому в таблиці «Відгуків» є зовнішній ключ до таблиці «Користувачів».

На рис. 2.3 представлено діаграму сутність-зв’язок.

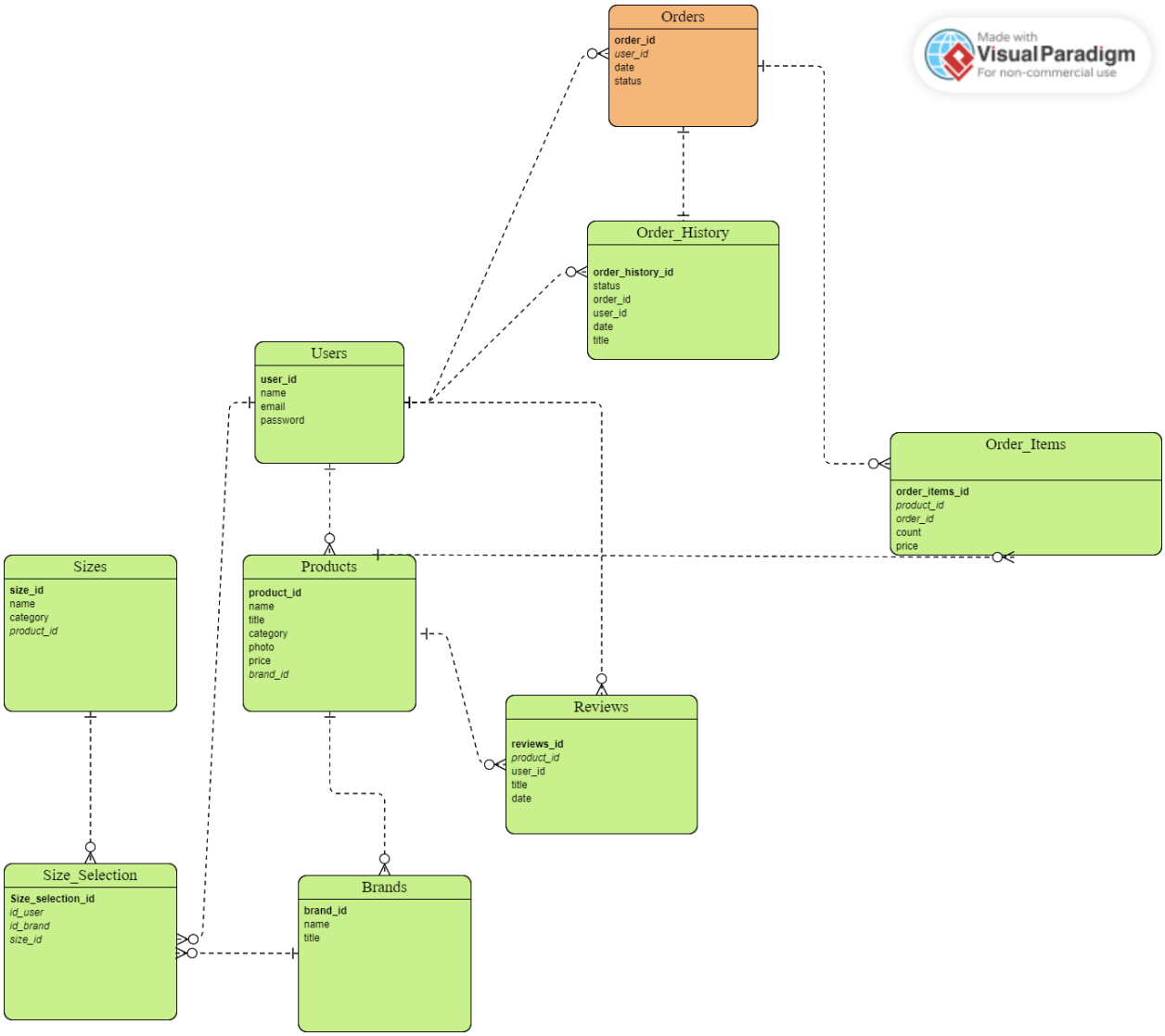

Рисунок 2.3 – Діаграма «сутність-зв’язок»

2.4 Моделювання поведінки системи

Діаграма активності для процесів реєстрації (рис. 2.4) та авторизації користувача (рис. 2.5) допомагає візуалізувати кроки та рішення, які відбуваються під час цих процесів.

Опис процесу реєстрації:

1. Початок процесу: Користувач відкриває форму реєстрації.
2. Введення даних: Користувач заповнює форму реєстрації, вказуючи необхідні дані (ім'я, прізвище, електронну адресу, пароль тощо).

3. Перевірка даних: Система перевіряє, чи всі необхідні поля заповнені правильно.
4. Перевірка унікальності: Система перевіряє, чи вже існує користувач з такою електронною адресою.
5. Збереження даних: якщо перевірка успішна, система зберігає дані користувача в базі даних.
6. Підтвердження реєстрації: Система відправляє користувачу підтвердження про успішну реєстрацію.
7. Кінець процесу: Користувач може увійти в систему.

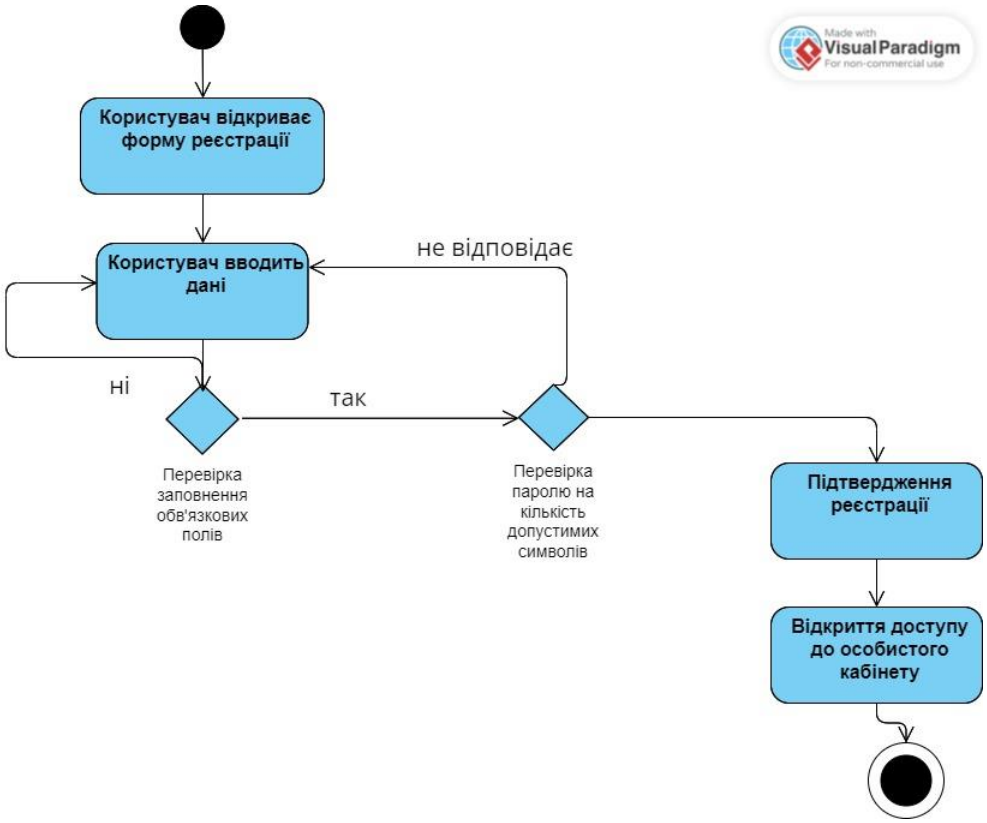

Рисунок 2.4 – Діаграма активності для процесу реєстрації користувача

Опис процесу авторизації:

1. Початок процесу: Користувач відкриває форму авторизації.
2. Введення даних: Користувач вводить електронну адресу та пароль.
3. Перевірка даних: Система перевіряє, чи всі необхідні поля заповнені.

4. Перевірка існування користувача: Система перевіряє, чи існує користувач з такою електронною адресою.

5. Перевірка пароля: Система перевіряє, чи правильно введено пароль.

6. Авторизація: якщо перевірка успішна, користувач авторизується в системі.

7. Помилка: якщо дані введено неправильно, система повертає повідомлення про помилку.

8. Кінець процесу: Користувач успішно авторизований або отримує повідомлення про помилку.

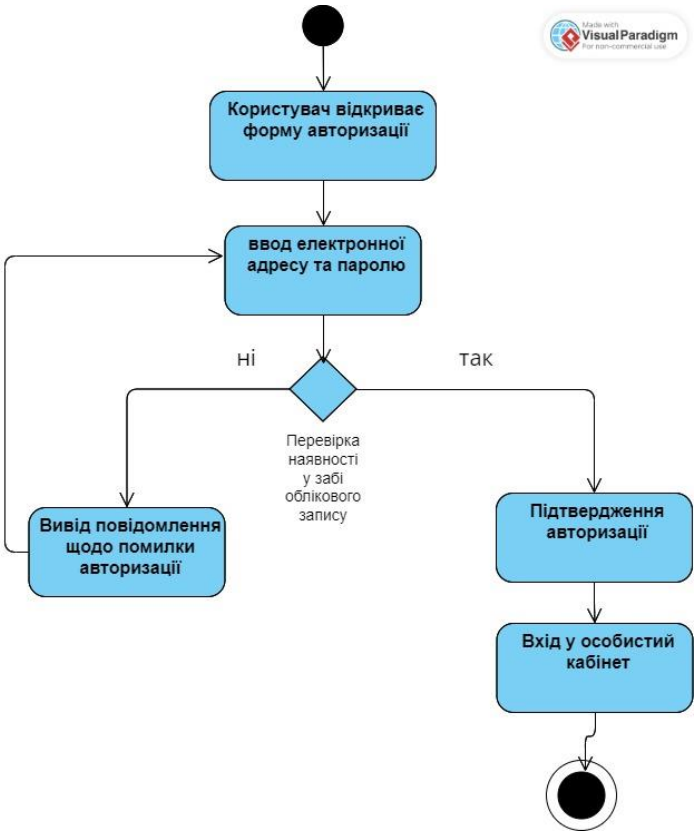

Рисунок 2.5 – Діаграма активності для процесу авторизації користувача

Діаграма послідовності відображає взаємодію між об'єктами в системі в контексті виконання певного процесу. Нижче наведено діаграму послідовності для процесу бронювання одягу в онлайн магазині:

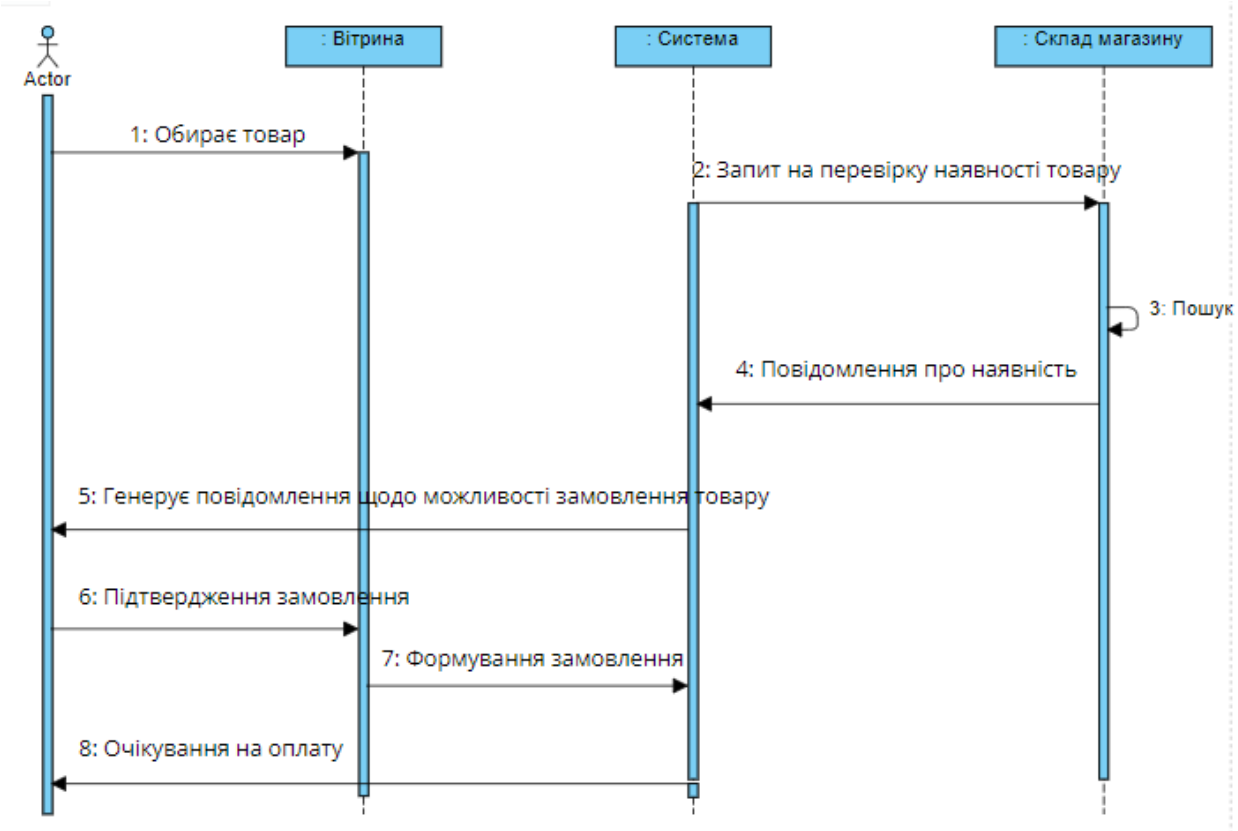

Рисунок 2.6 – Процес бронювання товару

Для відображення процесу роботи функції вибору правильного розміру обраного бренду одягу побудовано наступну діаграму послідовності дії (рис. 2.7). Опис процесу підбору розміру одягу:

1. Користувач вводить свої параметри (наприклад, обхват грудей, талії, стегон, зріст тощо).

2. Користувач обирає марку одягу.

3. Система отримує дані про параметри користувача та обрану марку.

4. Система виконує пошук у базі даних відповідних розмірів для заданої марки.

5. Система порівнює параметри користувача з параметрами розмірної сітки обраної марки.

6. Система надає рекомендацію щодо відповідного розміру.

7. Користувач отримує рекомендацію.

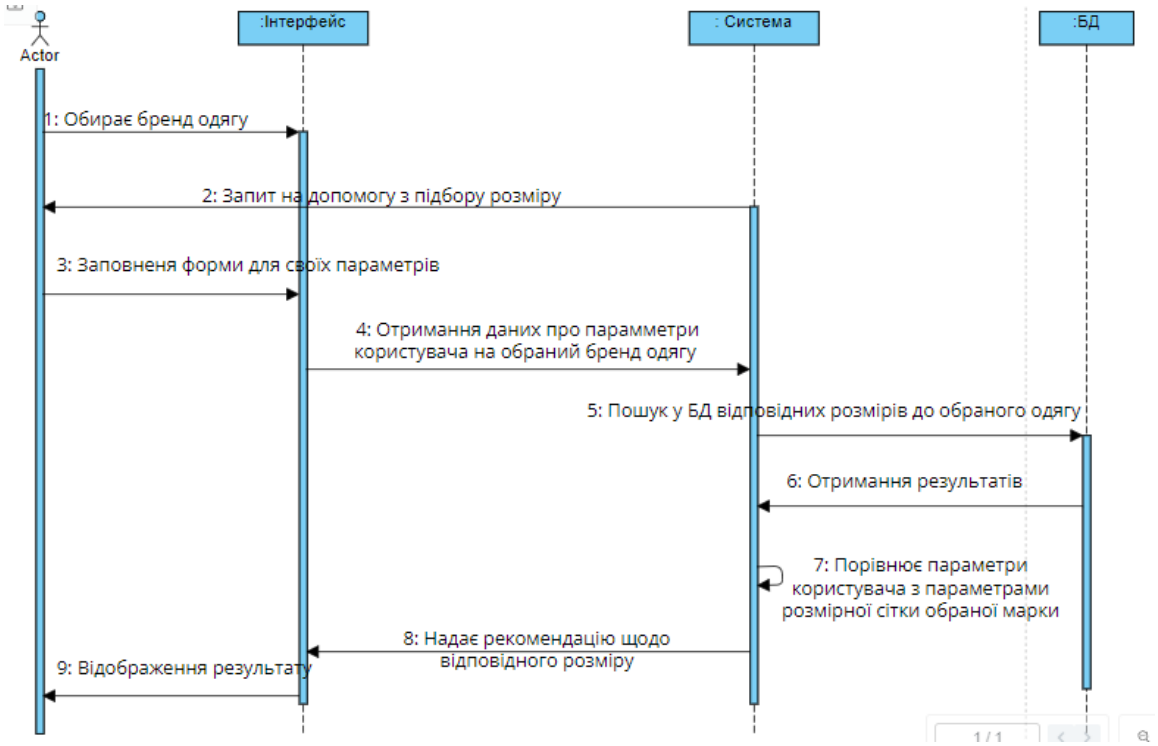

Рисунок 2.7 – Діаграма послідовності дії для вибору розміру одягу

На рис. 2.8 представлено діаграму станів для замовлення користувача.

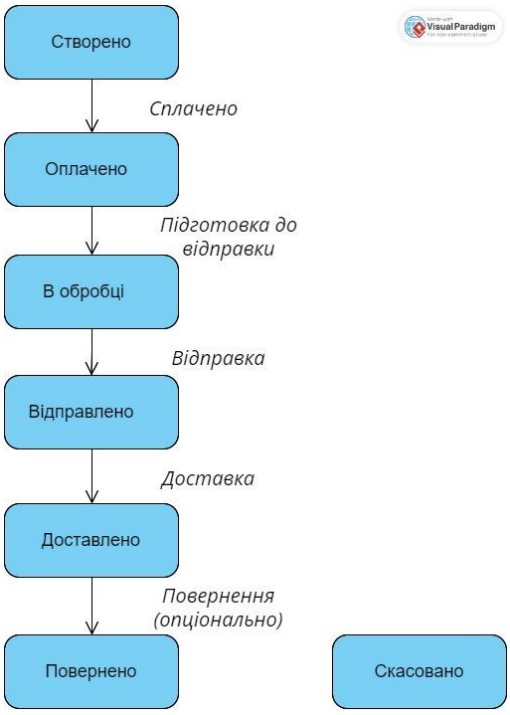

Рисунок 2.8 – Діаграма станів для замовлення

Ця діаграма допомагає візуалізувати життєвий цикл замовлення та переходи між різними станами в процесі обробки замовлень в онлайн-магазині.

Пояснення діаграми станів:

1. «Створено» (Created): замовлення створено після того, як користувач додає товари до кошика і підтверджує замовлення.

Перехід: при оплаті замовлення стан змінюється на "Оплачено".

2. «Оплачено» (Paid): замовлення отримує оплату від користувача.

Перехід: Після підтвердження оплати замовлення переходить у стан "В обробці".

3. «В обробці» (Processing): замовлення обробляється, включаючи перевірку наявності товарів та підготовку до відправлення.

Перехід: після завершення обробки замовлення переходить у стан "Відправлено".

4. «Відправлено» (Shipped): замовлення відправлено користувачу через службу доставки.

Перехід: після успішної доставки замовлення переходить у стан "Доставлено".

5. «Доставлено» (Delivered): замовлення доставлено та отримано користувачем.

Перехід: якщо користувач повертає товар, замовлення переходить у стан "Повернено".

6. «Скасовано» (Cancelled): замовлення скасовано користувачем або адміністратором до відправки.

Перехід: Це кінцевий стан для скасованих замовлень.

7. «Повернено» (Returned): Користувач повернув товар після отримання. Це кінцевий стан для повернених замовлень.

Ця діаграма станів допомагає візуалізувати життєвий цикл замовлення та переходи між різними станами в процесі обробки замовлень в онлайн-магазині.

3 ВИБІР ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ РОЗРОБКИ

Для розробки онлайн-магазину з мікросервісною архітектурою важливо обрати правильні програмні засоби та технології, які забезпечать масштабованість, гнучкість і надійність системи. Програмна платформа, або фреймворк, – це платформа для розробки програмних додатків. Це абстракція, яка дозволяє змінювати програмне забезпечення з загальними функціональними можливостями, додаючи додатковий користувацький код, тим самим створюючи програмне забезпечення для конкретних застосувань. Вона надає розробникам основу для створення та розгортання своїх додатків, представляючи універсальне програмне середовище для багаторазового використання.

3.1 Обґрунтування вибору технологій для фронтенду

Вибір засобів для розробки фронтенду є критичним, оскільки саме фронтенд визначає взаємодію користувача з сайтом. Головна ідея фронтенду – зробити веб-сторінки красивими та функціональними. Для цього потрібно знати кілька мов програмування та технологій:

- HTML (HyperText Markup Language): це скелет веб-сторінки. Він визначає, які елементи на сторінці мають бути, як вони пов'язані між собою та як виглядають;
- CSS (Cascading Style Sheets): це інструмент, який відповідає за стиль та зовнішній вигляд веб-сторінки. Він вирішує, який колір буде у тексту, які шрифти використовувати, які відступи та рамки додати – загалом він займається всім, що робить сторінку красивою;
- JavaScript: мова програмування робить веб-сторінки живими. JS дозволяє створювати інтерактивні елементи, анімацію, перевіряти форми та багато іншого [7].

Розглянемо декілька найпопулярніших фронтенд-фреймворків, щоб проаналізувати їх та обрати отпимальний для даного проекту:

1. React.js.

Це бібліотека JavaScript з відкритим вихідним кодом для створення компонентів користувацького інтерфейсу. Вона підтримується Facebook, а також спільнотою окремих компаній і розробників. React можна використовувати як основу при розробці мобільних або односторінкових додатків. Однак його використовують лише для управління станом і передачі цього стану в DOM. Це означає, що для створення додатків на React знадобляться додаткові бібліотеки для маршрутизації та певні функції на стороні клієнта.

Переваги React:

- безперебійна і стабільна продуктивність завдяки використанню віртуального DOM;
- можливість повторного використання компонентів полегшує спільну роботу, а також їх повторне використання в інших частинах додатка;
- ідеальна альтернатива написанню компонентів при використанні React Hook, оскільки це дозволяє розробникам писати компоненти без класів, а також легко вчити React;
- загальний процес створення компонентів спрощується з використанням JSX, безкоштовного розширення синтаксису;
- інструменти розробки react корисні та просунуті;
- react орієнтований на SEO, а платформа постачається з повним набором інструментів для розробників [8].

Характеристики React:

- стабільний і безпечний код;
- гнучкість у використанні як в інтернеті, так і на мобільних пристроях;
- односпрямована прив'язка даних;
- віртуальний DOM;

- підвищена продуктивність і швидкість;
- легкість у розширенні та навчанні;
- висока сумісність з обширними бібліотеками;
- JSX – розширення синтаксису javascript;
- відлагодження виконується швидше та легше.

2. Vue.JS.

Vue є моделлю представлення-viewmodel (MVVM), фреймворком JavaScript з відкритим вихідним кодом для розробки односторінкових додатків та інтерфейсів користувача. Vue – це проста та зрозуміла платформа, яка добре справляється з проблемами, з якими стикаються розробники Angular. Вона допомагає користувачам виконувати безліч завдань і з легкістю справляється з динамічними і простими процесами, включаючи мобільні та веб-додатки, а також веб-додатками, що розвиваються.

Переваги Vue.JS:

- простота – це величезна перевага фреймворку, яка дозволяє розробникам досягати хороших результатів за необхідності кодувати лише кілька рядків;
- однофайлові компоненти можуть зберігати всі коди в одному файлі та вимагають відносно невеликих витрат;
- Vue.JS може бути легко інтегрований в інші фреймворки, такі як React;
- зручний і простий у освоєнні, оскільки програмістам потрібно лише знати основи CSS, HTML та JavaScript;
- фреймворк можна використовувати із звичайними редакторами;
- всі його функції легко доступні, і програми можуть бути налаштовані відповідно до конкретних потреб;
- забезпечує більшу гнучкість і менше обмежень;
- хороша документація;
- велика спільнота, яка пропонує підтримку в навчанні та часто публікує оновлену інформацію [9].

Характеристики Vue.JS:

- віртуальний DOM;
- вбудовані CSS-переходи та анімації;
- прив'язка даних за допомогою v-bind;
- шаблони на основі HTML;
- мінімальний розмір (велика сумісність);
- простий синтаксис та інтеграція;
- пропонує організовану документацію;
- легко зрозуміти;
- спостерігачі (обробляє зміни даних);
- Vue-маршрутизатор (виконує навігацію між сторінками).

3. jQuery.

Запущений у 2006 році, jQuery є одним із найраніших фроненд-фреймворків, і, незважаючи на дату запуску, він продовжує залишатися актуальним у сучасному світі технологій. Цей фреймворк пропонує легкість і простоту використання, а також мінімізує необхідність написання великого коду JavaScript. Крім того, існує широке співтовариство jQuery, на яке розробники можуть покластися на пошуки рішень.

Його основна перевага – це простота використання. За допомогою jQuery розробники можуть звертатися до елементів DOM за допомогою простого та лаконічного синтаксису, що значно скорочує кількість коду та спрощує розробку [10].

Однією з особливостей jQuery є здатність обробляти події на веб-сторінці. За допомогою простих методів, таких як click(), hover() та keypress(), можна легко додавати інтерактивність до веб-сторінки та реагувати на дії користувача. jQuery також пропонує багату колекцію плагінів, які розширюють його функціональність та додають нові можливості. Це дозволяє розробникам швидко реалізовувати слайдери, спливаючі вікна, анімації та інші інтерактивні елементи на веб-сайтах.

Ще однією перевагою jQuery є його сумісність із безліччю браузерів. Він забезпечує однаковість роботи на різних платформах і гарантує, що функціональність однаково добре працюватиме в різних браузерах.

Характеристики jQuery: маніпуляція DOM;

- CSS маніпуляція;
- робота з HTML;
- вибір елемента DOM;
- утиліти;
- анімації та ефекти;
- методи подій HTML;
- AJAX;
- розширюваність за допомогою модулів, що підключаються;
- розбір JSON [11].

3.2 Обґрунтування вибору технологій для бекенду

Бекенд-розробка включає в себе декілька моментів:

1. Обробка даних.

Бекенд відповідає за отримання та обробку даних. Коли ви, наприклад, надсилаєте повідомлення в чаті або робите замовлення в інтернет-магазині, бекенд обробляє цю інформацію, зберігає її в базі даних і робить все, щоб ваш запит було виконано.

2. Управління базами даних: бекенд зберігає та керує даними, щоб ви могли їх отримувати, змінювати та видаляти.
3. Серверна логіка: тут відбувається «мислення» програми. Бекенд вирішує, як обробляти запити від фронтенду, що робити за певних подій і як надсилати дані назад на веб-сторінку.

Для вибору інструментальних засобів щодо реалізації бекенду, слід проаналізувати доступні з них.

1. Django.

Це безкоштовний веб-фреймворк на основі Python з відкритим кодом. Він слідує архітектурному шаблону MTV (модель-шаблон-вистави). Django Software Foundation або DSF – це американська незалежна організація, яка підтримує Django. Основна його мета – спростити створення складних веб-сайтів, керованих базами даних. Фреймворк наголошує на підключності та повторному використанні компонентів, низькому рівні зв'язності, меншій кількості коду, принципі “не повторюйся” та швидкій розробці.

Python використовується повсюдно, навіть для файлів, налаштувань та моделей даних. Фреймворк також надає адміністративний інтерфейс створення, читання, оновлення та видалення, який є необов'язковим та динамічно генерується за допомогою самодіагностики. Інтерфейс налаштовується за допомогою моделей адміністратора [12].

До переваг Django можна віднести:

- Django написаний мовою Python, що означає, що його легко вивчити;
- Django та Python є основними рішеннями в IT-гігантах;
- Включає ряд функціональних можливостей, таких як MVC-верстка, безкоштовний API, неймовірний ПЗУ, підтримку кількох мов та декількох сайтів, підтримка AJAX, робота з сесіями, легка міграція баз даних та багато іншого;
- великі посібники та документація, а також інтерфейс адміністрування;
- величезне співтовариство, яке пропонує ресурси та оновлення;
- масштабованість, безпека, вбудована система шаблонів.

Характеристики Django:

- Python веб-фреймворк;
- відмінна документація;
- висока масштабованість;
- SEO-підтримка;
- забезпечує високу безпеку;

- універсальна природа;
- ретельно протестовано на динамічні зміни в галузі;
- сприяє швидкому розвитку [13].

2. Laravel.

Laravel це безкоштовний веб-фреймворк PHP з відкритим вихідним кодом, заснований на Symfony. Його вихідний код розміщено на GitHub. Платформа була призначена для розробки веб-застосунків за архітектурним шаблоном MVC (модель-представлення-контролер). Деякі з функцій платформи – це утиліти, які допомагають в обслуговуванні та розгортанні додатків, різні способи доступу до реляційних баз даних, орієнтація на синтаксичну зручність та модульну систему упаковки з виділеним менеджером залежностей.

Переваги Laravel:

- Laravel спрощує реалізацію автентифікації, поряд з організацією автентичної логіки та контролем доступу до ресурсів;
- доступні драйвери для Mailgun, SMTP, SparkPost, поштової функції PHP, Amazon SES та Sendmail, а також простий та зрозумілий API через бібліотеку SwiftMailer;
- підтримує популярні бекенди кешу, такі як Redis з коробки та Memcached;
- високозахищені веб-застосунки, оскільки Laravel захищає від підробки міжсайтових запитів, впровадження SQL та міжсайтових сценаріїв;
- обробка винятків та помилок вже налаштована для всіх нових проектів на базі Laravel;
- робота з тестування автоматизована і файл phpunit.xml вже налаштований для програми;
- уніфікований API для широкого спектру різних бекендів черг.

Характеристики Laravel:

- вбудовані полегшені шаблони та безліч віджетів, що включають код JS та CSS;
- підтримка архітектури MVC;
- надійний захист веб-додатків;
- функціонально важливе ORM (об'єктно-реляційне зіставлення);
- вбудовані командні інструменти Artisan;
- попередньо встановлені модульні та об'єктно-орієнтовані бібліотеки.

Для розробки інтернет-магазину обрано jQuery Django.

3.3 Обґрунтування вибору СУБД

MySQL це одна з систем управління реляційними базами даних, що найчастіше використовуються – СУРБД або RDBMS (Relational Database Management System). Її відмінною особливістю є відкритий вихідний код, підтримка та швидкодія, що дозволило їй стати відмінною альтернативою комерційним системам. MySQL підтримується численними мовами програмування, зокрема PHP.

Найбільш очевидним плюсом є можливість безкоштовного користування. MySQL поряд з іншими супутніми інструментами можна безкоштовно завантажити з офіційного сайту та використовувати у своїй роботі, для розробки сайтів, наприклад, або просто для вивчення SQL.

MySQL дуже популярна система, а це означає, що вона доступна на будь-яких операційних системах. Наприклад, якщо ви використовуєте Windows, то з офіційного сайту можете завантажити MySQL Installer і в кількох кліках мати на своїй машині всі продукти MySQL.

Популярність і відкритий вихідний код породжує створення численних інструментів, пов’язаних з MySQL і, зокрема, з розробкою сайтів, від інших розробників. Яскравим прикладом може бути phpMyAdmin або XAMPP [14].

4 ОПИС РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Огляд структури сайту

Структура сайту – це організація і взаємозв'язок між різними сторінками, секціями та елементами сайту, що дозволяє користувачам легко знаходити потрібну інформацію та забезпечує зручну навігацію. Структура сайту визначає, як контент розміщується на різних сторінках і як ці сторінки пов'язані між собою. Правильна структура сайту важлива як для користувачів, так і для пошукових систем.

Розглянемо структуру сайту більш детально. Головна сторінка (Homepage) (рис. 4.1) – це центральна сторінка, з якої користувачі починають навігацію. Вона містить логотип магазину, загальний огляд сайту, основні категорії та посилання на важливі розділи.

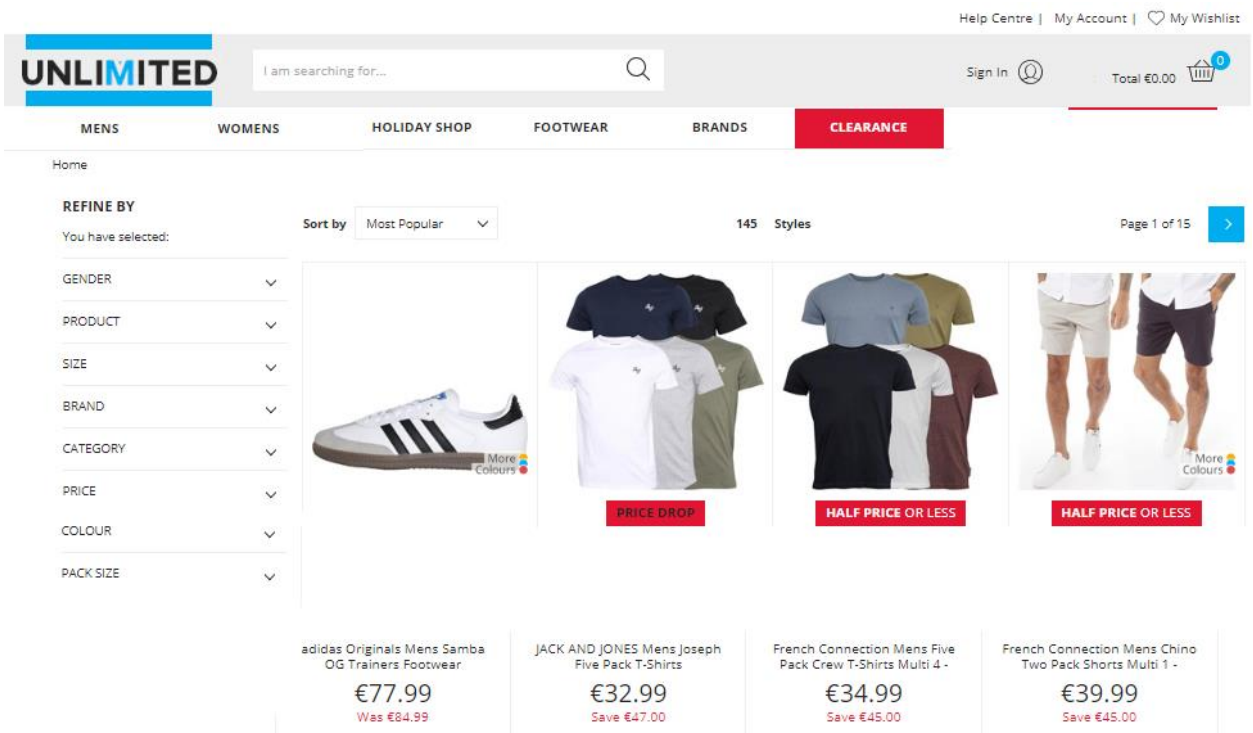

Рисунок 4.1 – Головна сторінка сайту

У хедері розміщено логотип магазину «Unlimited», строка пошуку з кнопкою її активації, меню з «Help Center», де користувачі можуть отримати відповіді на гарячі питання, посилання на власний кабінет зареєстрованого користувача, а також перехід до категорії обраних товарів «My Wishlist». Також у хедері є посилання на форму реєстрації/авторизації користувача і сторінки для перегляду кошика покупок та оформлення замовлення.

Меню має 6 основних категорій:

- Mens – меню сторінок з одягом для чоловіків (рис. 4.2);
- Womens – меню сторінок з одягом для жінок;
- Holiday Shop – актуальні пропозиції щодо обраних свят;
- Footwear – сторінка з різноманіттям взуття;
- Clearance – сторінка розродажів.

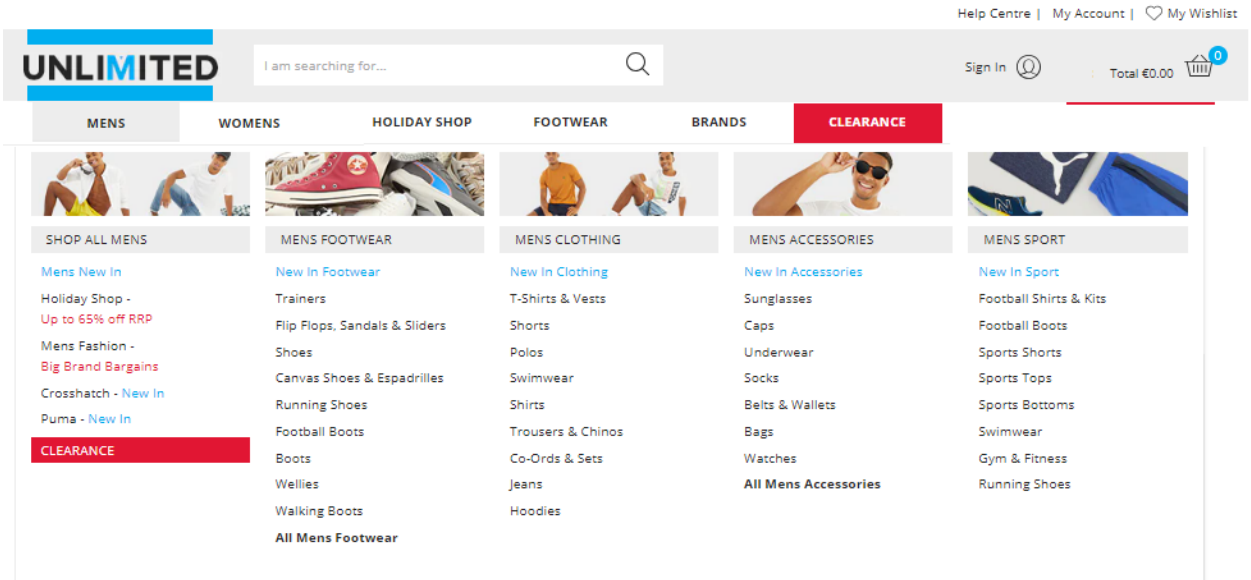

Рисунок 4.2 – Меню «Mens» з переліком доступних категорій товарів

Сторінка «Clearance» (рис. 4.3) є важливим елементом будь-якого інтернет-магазину одягу, оскільки вона сприяє залученню клієнтів та ефективному управлінню запасами товарів. Ця сторінка на сайті з продажу одягу призначена для розміщення товарів, які продаються зі знижками або

перебувають на розпродажі. Сторінка «Clearance» зазвичай має декілька основних функцій:

1) залучення покупців: знижки та розпродажі привертають увагу клієнтів, заохочуючи їх робити покупки. Це ефективний спосіб залучити нових покупців та збільшити продажі;
2) звільнення складу: допомагає позбутися старих колекцій або товарів, які не користуються попитом. Це звільняє складські приміщення для нових надходжень;
3) підвищення лояльності клієнтів: постійні покупці часто шукають вигідні пропозиції. Наявність сторінки «Clearance» може підвищити їхню лояльність до бренду.

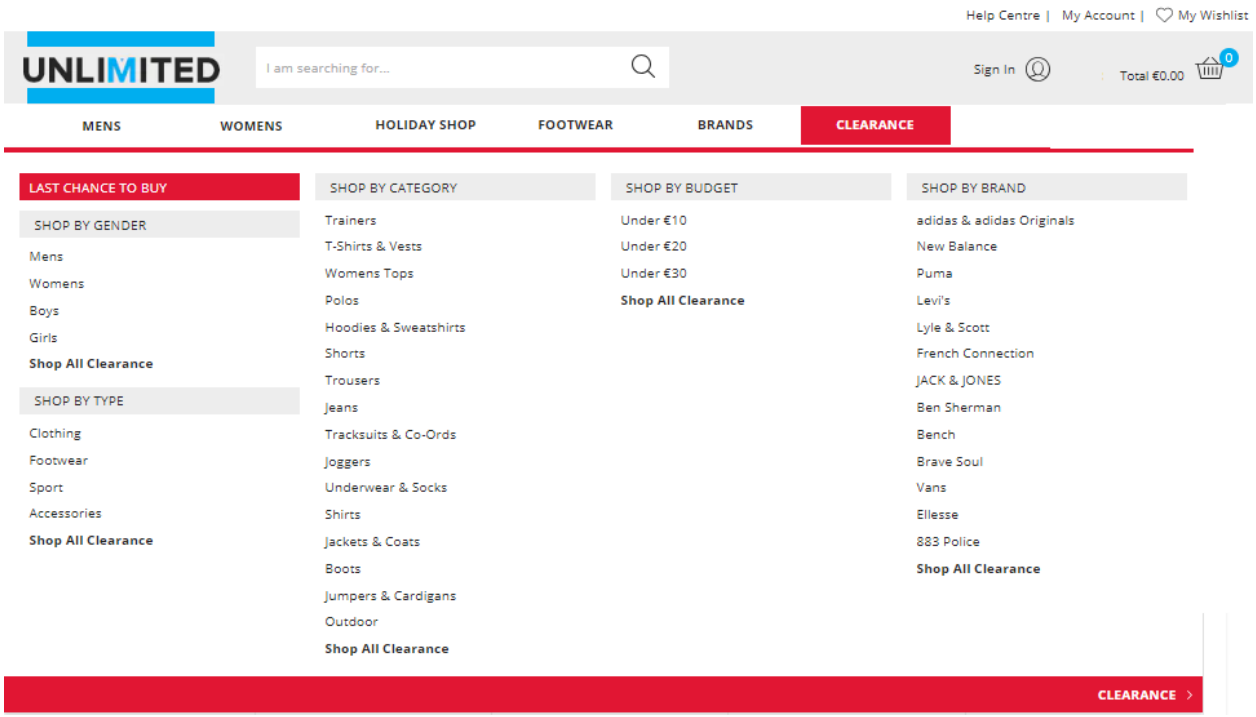

Рисунок 4.3 – Розділ головного меню «Clearance»

Мета меню сортування на сторінці інтернет-магазину одягу полягає в тому, щоб полегшити користувачам пошук товарів, які відповідають їхнім уподобанням та потребам (рис. 4.4). Це підвищує зручність користувацького

досвіду та сприяє швидшому прийняттю рішень щодо покупки. Сортування на сайті відбувається за такими параметрами, як:

- Gender (Men/Women);
- Product – за назвою продукту;
- Size – за розміром (одягу чи взуття) (рис. 4.5);
- Brand – за брендом товару;
- Category – за категорією (Accessories, Clothing, Footwear);
- Price – за ціною (Under €15, Under €25, Under €35, Under €50, Over €50);
- Colour – за кольором виробу (рис. 4.5);
- Pack size – за кількістю в одній упаковці (від 1 до 12).

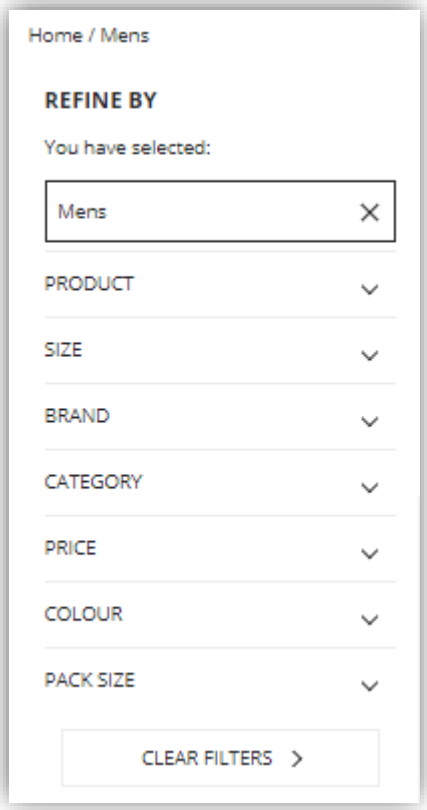

Рисунок 4.4 – Меню сортування товару

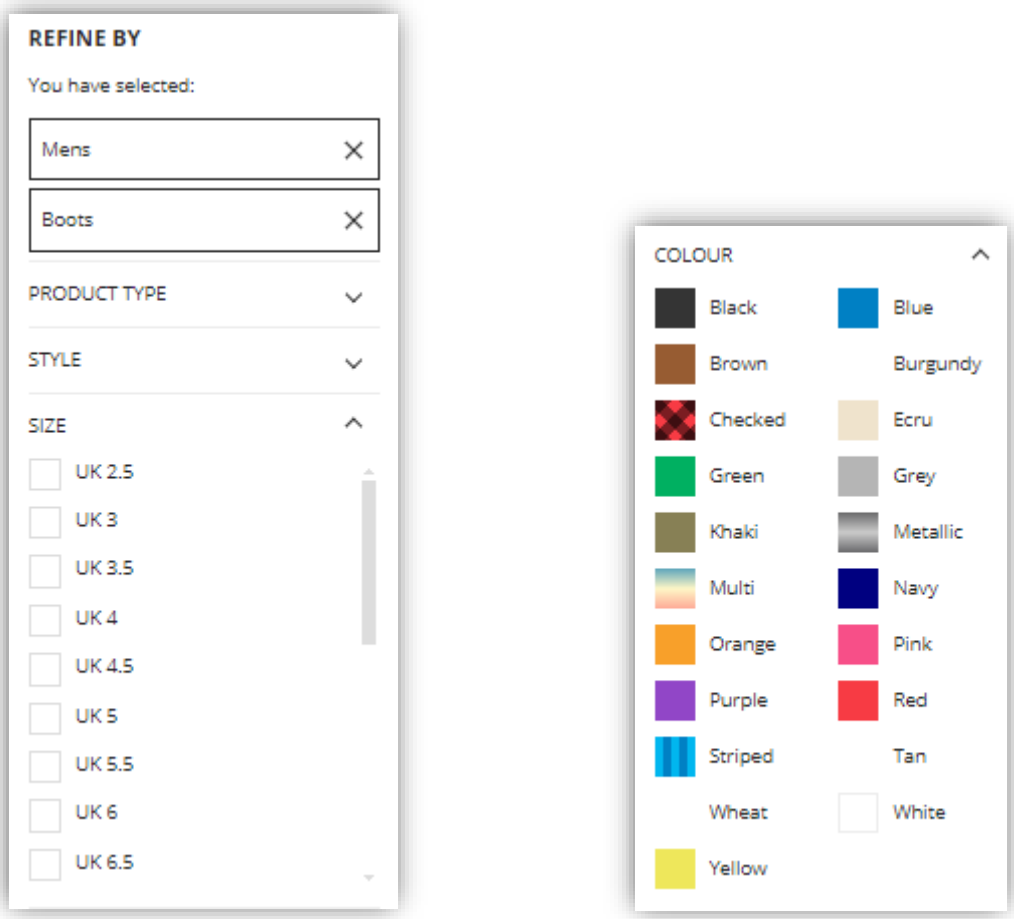

Рисунок 4.5 – Меню сортування за розміром та кольором товару

Меню сортування значно покращує зручність користувачів, дозволяючи їм швидко знаходити та купувати бажані товари, що в свою чергу підвищує задоволеність клієнтів та конверсії на сайті.

4.2 Опис картки товару

Картка товару на сайті з продажу одягу є ключовим елементом, що дозволяє користувачам ознайомитися з деталями товару, побачити його зображення, дізнатися ціну та іншу важливу інформацію. Вона також забезпечує можливість додавання товару до кошика або бажаного списку.

Приклад картки товару представлено на рисунку 4.6: на картці розміщено фото товару з позначкою можливих кольорів, повна назва товару, ціна товару за замовчуванням, ціна зі знижкою і розмір самої знижки на цю пропозицію.

Рисунок 4.6 – Картка товару

Кнопка «Quick Buy» допомагає придбати товар у один клік. Після її натискання відкривається додаткова панель з переліком розмірів обраної позиції. Клік на одній з цих значень переліку (наприклад, потрібний розмір взуття) кнопка «Add To Basket» стає активною (рис. 4.7).

Після кожного додавання товару у кошик, у хедері праворуч з’являється біля іконки кошика значення кількості обраних товарів для покупки, а також загальна сума покупки (рис. 4.8). Червоним кольором завжди виділяється окремим записом загальна знижка – це допомагає стимулювати користувача робити більше покупок та отримувати задоволення від гарних пропозицій.

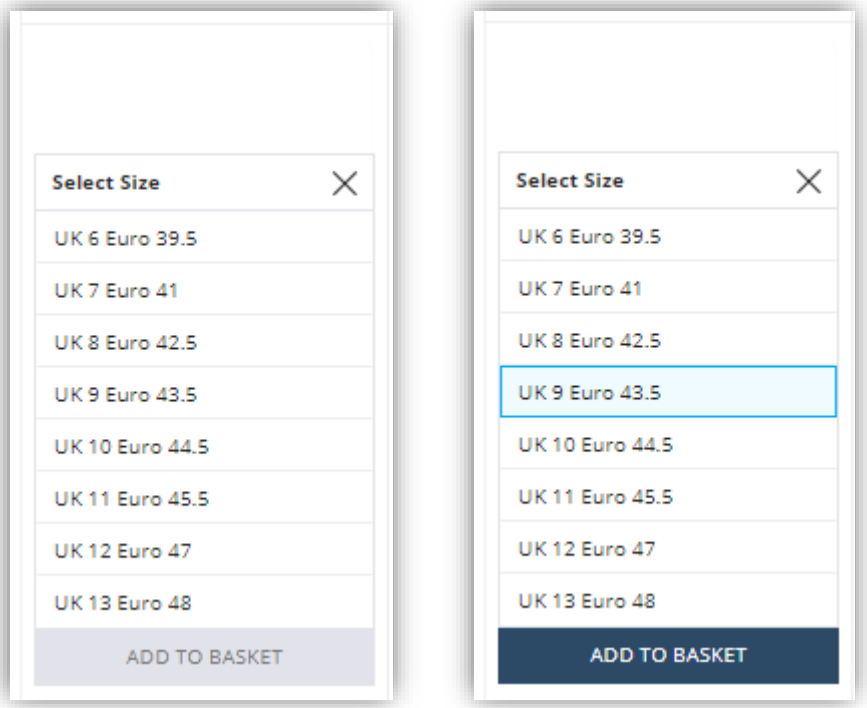

Рисунок 4.7 – Активація кнопки «Add To Basket»

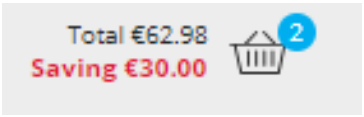

Рисунок 4.8 – Відображення стану кошика

При натисканні на іконку кошика відкривається додаткове вікно «Added To Your Basket» (рис. 4.9). Тут відображається повна назва обраних до кошика товарів, ціна, пропозиція щодо безкоштовної доставки при умові замовлення від 100 євро, також тут розміщено дві кнопки «Basket» – для переходу до змісту кошика, «Checkout» – для додавання додаткових товарів до замовлення. З точки зору реклами, користувачу висвічується інформація щодо гарячих розпродажів та знижок у даний час.

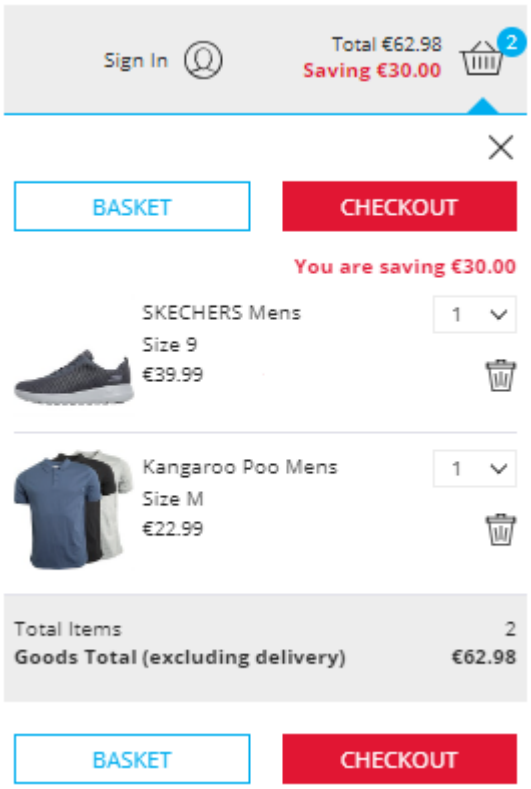

Рисунок 4.9 – Вікно зі швидким переглядом кошика

4.3 Форма для оформлення замовлення

Процес авторизації та реєстрації на сайті з продажу одягу відіграє важливу роль у забезпеченні зручного та безпечного оформлення замовлень. Ці процеси допомагають зібрати необхідну інформацію про клієнтів, забезпечують доступ до персональних кабінетів та історії замовлень, а також підвищують лояльність клієнтів.

Реєстрація нового користувача зазвичай включає такі кроки:

1. Форма реєстрації: користувач заповнює форму, вказуючи основні дані: ім'я, прізвище, електронну адресу, пароль, номер телефону тощо.
2. Підтвердження електронної пошти: користувач отримує електронний лист з підтвердженням реєстрації та посиланням для активації облікового запису.

3. Створення профілю: після підтвердження електронної адреси, створюється профіль користувача, де можна зберігати особисті дані, адресу доставки, переглядати історію замовлень тощо.

Під час початку оформлення замовлення система відкриває вікно для авторизації (рис. 4.10), де користувачу необхідно вказати чи є у нього зареєстрований акаунт (якщо так, то користувач вводить свою пошту і пароль, а система перевіряє наявність даних у своїй базі), чи він заходить під статусом гостя (достатньо ввести тільки поштову адресу).

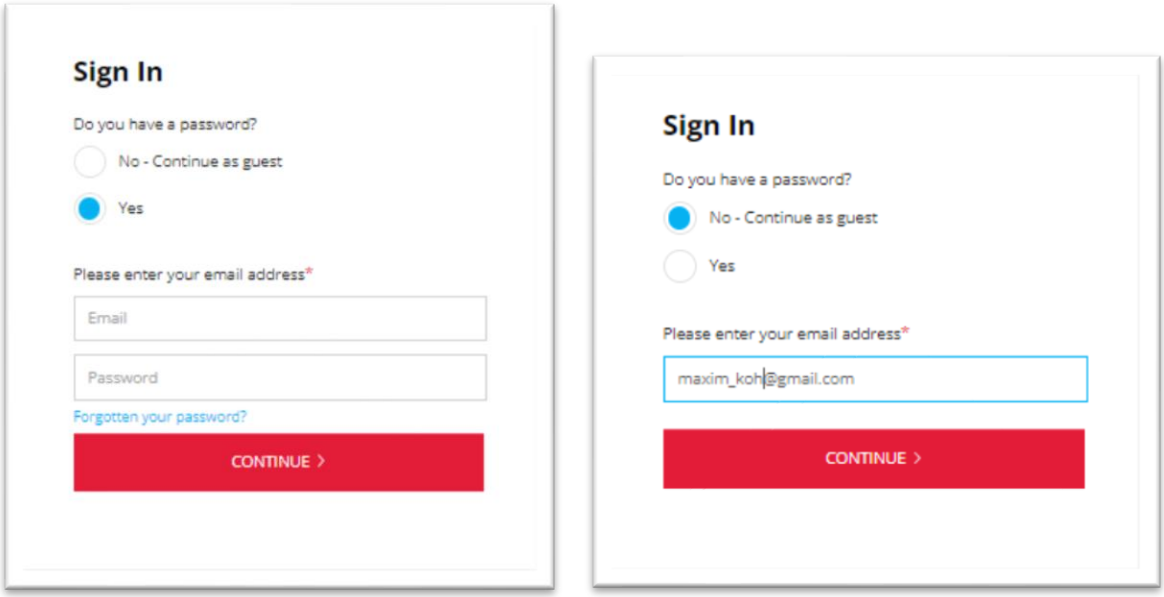

Рисунок 4.10 – Форми для авторизації користувачів

Для оформлення замовлення і внесення даних для доставки покупок користувачу слід заповнити форму замовлення, де присутні обов’язкові поля: ім’я, поштова адреса, куди прийде лист-підтвердження замовлення та інформація щодо статусу замовлення і його доставки (рис. 4.11).

Після цього користувач натискає на кнопку «Continue to billing address» і переходить до наступної форми де йому потрібно буде вказати адресу доставки та спосіб оплати свого замовлення.

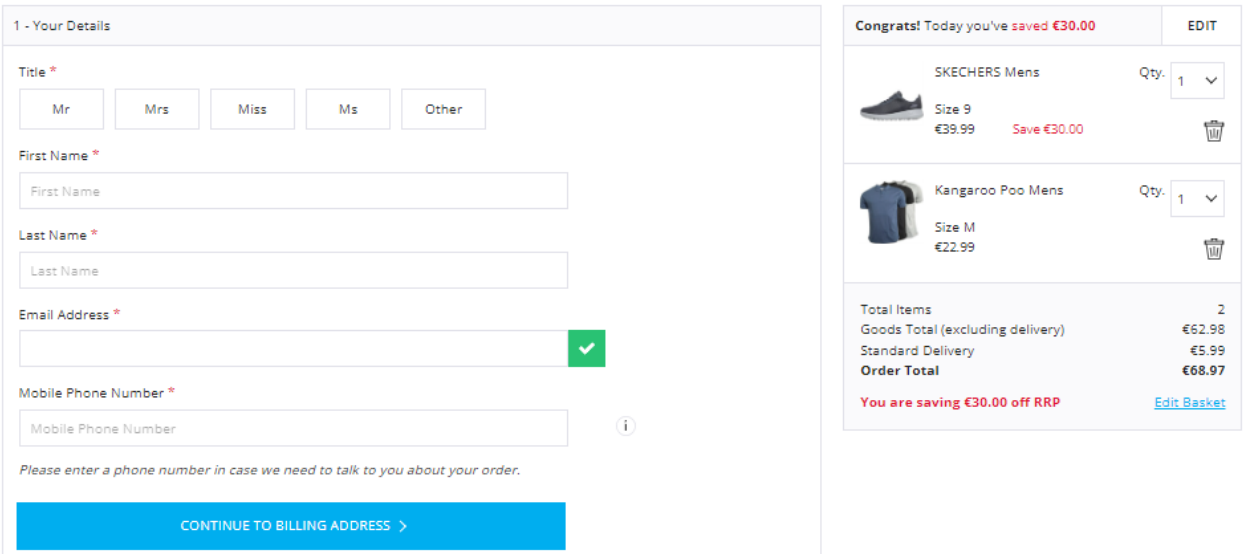

Рисунок 4.11 – Форма оформлення замовлення

4.4 Реалізація основних функцій

4.4.1 Авторизація користувача

Реалізація авторизації користувачів, що можуть обирати між входом як зареєстрований користувач або оформленням замовлення як гість, и виконана за допомогою Django для бекенду та HTML, CSS і JavaScript для фронтенду. У цьому випадку Django забезпечує обробку даних та взаємодію з базою даних, тоді як фронтенд частина відповідає за візуальне відображення та взаємодію з користувачем.

Django (бекенд):

- реєстрація та авторизація користувачів;
- обробка даних форми;
- робота з базою даних для зберігання користувацької інформації.

HTML/CSS/JavaScript (фронтенд):

- відображення форми авторизації/реєстрації;
- взаємодія з користувачем (перемикання між режимами авторизації та гостьового замовлення).

Послідовність виконання кроків для реалізації алгоритму:

1.Налаштування Django проекту – створюємо новий проект. Додаємо «users» до INSTALLED_APPS в «settings.py».

```
django-admin startproject shop
cd shop
django-admin startapp users
```

2. Моделі та форми для користувачів:

```
«models.py»
     from django.contrib.auth.models import AbstractUser
     from django.db import models

     class CustomUser(AbstractUser):
         email = models.EmailField(unique=True)

«forms.py»
     from django import forms
     from  django.contrib.auth.forms  import  UserCreationForm,
     AuthenticationForm
     from .models import CustomUser

     class CustomUserCreationForm(UserCreationForm):
         class Meta:
             model = CustomUser
             fields = ('email', 'password')

     class CustomAuthenticationForm(AuthenticationForm):
         username = forms.EmailField(label="Email")
```

3. Представлення (views) для реєстрації та авторизації:

```
     «views.py»
     from django.shortcuts import render, redirect
     from django.contrib.auth import login, authenticate
     from .forms import CustomUserCreationForm,
CustomAuthenticationForm

     def register(request):
         if request.method == 'POST':
             form = CustomUserCreationForm(request.POST)
             if form.is_valid():
                 user = form.save()
                 login(request, user)
```

```
                return redirect('home')
        else:
            form = CustomUserCreationForm()
        return render(request, 'register.html', {'form': form})

    def login_view(request):
        if request.method == 'POST':
            form = CustomAuthenticationForm(request,
data=request.POST)
            if form.is_valid():
                user = form.get_user()
                login(request, user)
                return redirect('home')
        else:
            form = CustomAuthenticationForm()
        return render(request, 'login.html', {'form': form})

    def guest_checkout(request):
        if request.method == 'POST':
            email = request.POST.get('email')

            # Обробка гостьового замовлення
            return redirect('home')
        return render(request, 'guest_checkout.html')
```

4. URL конфігурація:

```
«urls.py»
from django.urls import path
from . import views

urlpatterns = [
    path('register/', views.register, name='register'),
    path('login/', views.login_view, name='login'),
    path('guest-checkout/', views.guest_checkout,
name='guest_checkout'),
]
```

5. JavaScript для перемикання між формами:

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Sing In</title>
    <script>
        function toggleForm(formType) {
```

```
            document.getElementById('login-form').style.display
= formType === 'login' ? 'block' : 'none';
            document.getElementById('guest-form').style.display
= formType === 'guest' ? 'block' : 'none';
        }
    </script>
</head>
<body>
    <h2>Do you have a password&</h2>
    <div>
        <input type="radio" id="login" name="formType"
value="login" onclick="toggleForm('login')">
        <label for="login">No- Countinue as guest</label>
        <input type="radio" id="guest" name="formType"
value="guest" onclick="toggleForm('guest')">
        <label for="guest">Yes</label>
    </div>

    <div id="login-form" style="display:none;">
        <form action="/login/" method="post">
            {% csrf_token %}
            <label for="email">Please enter your e-mail
address:</label>
            <input type="email" id="email" name="username"
required>
            <label for="password">Passwordlabel>
            <input type="password" id="password" name="password"
required>
            <button type="submit">Continue “>”</button>
        </form>
    </div>

    <div id="guest-form" style="display:none;">
        <form action="/guest-checkout/" method="post">
            {% csrf_token %}
            <label for="guest-email">E-mail:</label>
            <input type="email" id="guest-email" name="email"
required>
            <button type="submit">continue as guest</button>
        </form>
    </div>
</body>
</html>
```

Приклад HTML коду меню сортування

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
```

```
    <title>Сортування товарів</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <nav>
            <a href="/">Головна</a>
            <a href="/clearance">Розпродаж</a>
            <!-- інші посилання -->
        </nav>
    </header>
    <main>
        <h1>Сортування товарів</h1>
        <div class="filters">
            <label for="size">Розмір:</label>
            <select id="size" name="size">
                <option value="all">Всі</option>
                <option value="s">S</option>
                <option value="m">M</option>
                <option value="l">L</option>
                <option value="xl">XL</option>
            </select>

            <label for="color">Колір:</label>
            <select id="color" name="color">
                <option value="all">Всі</option>
                <option value="red">Червоний</option>
                <option value="blue">Синій</option>
                <option value="green">Зелений</option>
                <option value="black">Чорний</option>
                <option value="white">Білий</option>
            </select>

            <label for="price">Ціна:</label>
            <select id="price" name="price">
                <option       value="default">Сортувати       за
ціною</option>
                <option   value="low_to_high">Від   дешевих   до
дорогих</option>
                <option   value="high_to_low">Від   дорогих   до
дешевих</option>
            </select>

            <label for="sort">Сортувати за:</label>
            <select id="sort" name="sort">
                <option value="popularity">Популярністю</option>
                <option value="newest">Новизною</option>
                <option value="discount">Розміром знижки</option>
            </select>
        </div>
        <div class="products">
            <!-- Приклади товарів -->
```

```
            <div class="product">
                <img src="product1.jpg" alt="Продукт 1">
                <h2>Футболка</h2>
                <p><span class="old-price">500 грн</span> <span
class="new-price">300 грн</span></p>
                <button>Додати до кошика</button>
            </div>
            <!-- інші товари -->
        </div>
    </main>
    <footer>
        <p>&copy; 2024 Магазин одягу</p>
    </footer>

    <!-- JavaScript для обробки сортування -->
    <script>

document.getElementById('size').addEventListener('change',
function() {
            // Додати логіку для сортування за розміром
        });


document.getElementById('color').addEventListener('change',
function() {
            // Додати логіку для сортування за кольором
        });


document.getElementById('price').addEventListener('change',
function() {
            // Додати логіку для сортування за ціною
        });


document.getElementById('sort').addEventListener('change',
function() {
            // Додати логіку для загального сортування
        });
    </script>
</body>
</html>
```

Таке рішення забезпечує зручний процес авторизації та оформлення замовлень для користувачів, надаючи їм вибір між входом у систему або оформленням замовлення як гість. Django на бекенді обробляє всі необхідні дані та взаємодіє з базою даних, а HTML/JavaScript на фронтенді відповідає за взаємодію з користувачем та зручність користування сайтом.

4.4.2 Створення БД

Структура бази даних для інтернет-магазину одягу з використанням MySQL:

```
CREATE TABLE Users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    email VARCHAR(255) UNIQUE,
    password VARCHAR(255),
    delivery_address TEXT,
    phone VARCHAR(20),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
CREATE TABLE Products (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    description TEXT,
    category VARCHAR(100),
    price DECIMAL(10, 2),
    image_url VARCHAR(255),
    brand_id INT,
    FOREIGN KEY (brand_id) REFERENCES Brands(brand_id)
);
CREATE TABLE Orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(50),
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
CREATE TABLE OrderItems (
    order_item_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT,
    unit_price DECIMAL(10, 2),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
CREATE TABLE Sizes (
    size_id INT AUTO_INCREMENT PRIMARY KEY,
    size_name VARCHAR(50),
    category VARCHAR(100)
);
CREATE TABLE Brands (
    brand_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    brand_name VARCHAR(100),
    brand_info TEXT,
    logo_url VARCHAR(255)
);
CREATE TABLE SizeSelection (
    size_selection_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    brand_id INT,
    size_id INT,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (brand_id) REFERENCES Brands(brand_id),
    FOREIGN KEY (size_id) REFERENCES Sizes(size_id)
);
CREATE TABLE OrderHistory (
    order_history_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    event_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    event_description TEXT,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id)
);
CREATE TABLE Reviews (
    review_id INT AUTO_INCREMENT PRIMARY KEY,
    product_id INT,
    user_id INT,
    rating INT CHECK (rating >= 1 AND rating <= 5),
    review_text TEXT,
    review_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (product_id) REFERENCES Products(product_id),
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```

4.4.3 Збереження замовлення користувача

Збереження замовлення користувача в базі даних передбачає кілька кроків, включаючи збереження інформації про саме замовлення та окремі елементи замовлення (товари, що були додані до кошика). Процес збереження замовлення має такі кроки:

1. Створення нового запису в таблиці «Orders»: зберігається загальна інформація про замовлення, така як ідентифікатор користувача, дата та час замовлення, статус замовлення тощо.
2. Збереження елементів замовлення в таблиці «OrderItems»: для кожного товару в замовленні створюється окремий запис, що містить

ідентифікатор замовлення, ідентифікатор товару, кількість та ціну за одиницю товару.

3. Додавання записів в таблицю «OrderHistory» для збереження подій, пов’язаних з замовленням.

Розглянемо програмну реалізацію більш детально.

1. Створення нового замовлення:

«views.py» (Django)

```
from django.shortcuts import render, redirect
from .models import Orders, OrderItems, Products
from django.utils import timezone
from django.contrib.auth.decorators import login_required

@login_required
def create_order(request):
    if request.method == 'POST':
        user = request.user
        # Збереження замовлення
        order = Orders(user=user, order_date=timezone.now(),
status='new')
        order.save()

        # Отримання товарів з кошика (припустимо, що є функція
get_cart_items())
        cart_items = get_cart_items(user)

        # Збереження кожного елемента замовлення
        for item in cart_items:
            product = Products.objects.get(pk=item.product_id)
            order_item = OrderItems(
                order=order,
                product=product,
                quantity=item.quantity,
                unit_price=product.price
            )
            order_item.save()

        # Очищення кошика після створення замовлення (припустимо,
що є функція clear_cart())
        clear_cart(user)

        return redirect('order_confirmation', order_id=order.id)
        return render(request, 'checkout.html')
```

2. Функції для отримання товарів з кошика та очищення кошика:

```
«utils.py»
from .models import Cart, CartItems

def get_cart_items(user):
    cart = Cart.objects.get(user=user)
    return CartItems.objects.filter(cart=cart)

def clear_cart(user):
    cart = Cart.objects.get(user=user)
    CartItems.objects.filter(cart=cart).delete()
```

3.Далі в роботу вступають декілька SQL запитів для збереження замовлення. Вставка нового замовлення в таблицю «Orders» та вставка елементів замовлення в таблицю «OrderItems»:

```
INSERT INTO Orders (user_id, order_date, status) VALUES (1,
NOW(), 'new');

INSERT INTO OrderItems (order_id, product_id, quantity,
unit_price) VALUES (LAST_INSERT_ID(), 1, 2, 19.99);
INSERT INTO OrderItems (order_id, product_id, quantity,
unit_price) VALUES (LAST_INSERT_ID(), 2, 1, 49.99);
```

Таблиця «Orders» зберігає загальну інформацію про замовлення, включаючи ідентифікатор користувача, дату та час створення замовлення, статус замовлення. Таблиця «OrderItems» зберігає інформацію про кожен товар, що був доданий до замовлення, включаючи ідентифікатор замовлення, ідентифікатор товару, кількість одиниць товару та ціну за одиницю.

Такий підхід дозволяє зберігати детальну інформацію про замовлення користувачів, забезпечуючи гнучкість та масштабованість системи.

5 ТЕСТУВАННЯ РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Тестування інформаційної системи з використанням тест-кейсів є важливою частиною процесу забезпечення якості програмного забезпечення. Тест-кейси допомагають визначити, чи відповідає система вимогам і чи працює вона належним чином.

Для основних варіантів використання інформаційної системи складено наступні тест-кейси, які описують передумови перевірки, головних акторів сценарію, очікуваний результат і кроки, які повинні до нього привести.

5.1 Перевірка реєстрації нового користувача

Тест-кейс №1 «Успішна реєстрація нового користувача»

Опис: перевірити можливість реєстрації нового користувача з валідними даними.

Передумови: Користувач знаходиться на сторінці реєстрації.

Кроки:

1. Ввести ім'я: "Іван".
2. Ввести прізвище: "Іванов".
3. Ввести електронну адресу: "ivanov@example.com".
4. Ввести пароль: "StrongPassword123".
5. Ввести підтвердження паролю: "StrongPassword123".
6. Натиснути кнопку "Continue".

Очікуваний результат: Користувач отримує повідомлення про успішну реєстрацію та перенаправляється на сторінку входу.

Тест-кейс №2 «Реєстрація з уже існуючою електронною адресою»

Опис: перевірити, що система не дозволяє реєстрацію з вже зареєстрованою електронною адресою.

Передумови: Користувач знаходиться на сторінці реєстрації, електронна адреса "ivanov@example.com" вже зареєстрована.

Кроки:

1. Ввести ім'я: "Іван".
2. Ввести прізвище: "Іванов".
3. Ввести електронну адресу: "ivanov@example.com".
4. Ввести пароль: "AnotherPassword123".
5. Ввести підтвердження паролю: "AnotherPassword123".
6. Натиснути кнопку " Continue".

Очікуваний результат: Користувач отримує повідомлення про те, що електронна адреса вже зареєстрована.

5.2 Перевірка авторизації користувача

Тест-кейс №3 «Успішна авторизація зареєстрованого користувача»

Опис:п можливість авторизації зареєстрованого користувача з валідними даними.

Передумови: Користувач знаходиться на сторінці авторизації, обліковий запис "ivanov@example.com" існує.

Кроки:

1. Ввести електронну адресу: "ivanov@example.com".
2. Ввести пароль: "StrongPassword123".
3. Натиснути кнопку " Continue".
4. Очікуваний результат: Користувач успішно входить у систему та перенаправляється на головну сторінку.

Тест-кейс №4 «Авторизація з невірним паролем»

Опис: перевірити, що система не дозволяє авторизацію з невірним паролем.

Передумови: Користувач знаходиться на сторінці авторизації, обліковий запис "ivanov@example.com" існує.

Кроки:

1. Ввести електронну адресу: "ivanov@example.com".

2. Ввести пароль: "WrongPassword".
3. Натиснути кнопку " Continue".
4. Очікуваний результат: Користувач отримує повідомлення про невірний пароль.

5.3 Перевірка оформлення замовлення

Тест-кейс №5 «Оформлення замовлення зареєстрованим користувачем»

Опис: Перевірити можливість оформлення замовлення зареєстрованим користувачем.

Передумови: Користувач авторизований та має товари у кошику.

Кроки:

1. Перейти до кошика.
2. Перевірити товари у кошику.
3. Натиснути кнопку "Оформити замовлення".
4. Ввести необхідні дані для доставки.
5. Підтвердити замовлення.

Очікуваний результат: замовлення успішно оформлене, користувач отримує повідомлення про успішне оформлення.

Тест-кейс №6 «Оформлення замовлення гостем»

Опис: перевірити можливість оформлення замовлення без реєстрації (як гість).

Передумови: Користувач має товари у кошику.

Кроки:

1. Перейти до кошика.
2. Перевірити товари у кошику.
3. Натиснути кнопку "Оформити замовлення".
4. Вибрати опцію "Оформити як гість".
5. Ввести необхідні дані для доставки.
6. Підтвердити замовлення.

Очікуваний результат: замовлення успішно оформлене, користувач отримує повідомлення про успішне оформлення.

5.4 Перевірка паролю

Тест-кейс №7 «Успішне відновлення паролю»

Опис: Перевірити можливість відновлення паролю для зареєстрованого користувача.

Передумови: Користувач знаходиться на сторінці відновлення паролю.

Кроки:

1. Ввести електронну адресу: "ivanov@example.com".
2. Натиснути кнопку "forgot password".
3. Перевірити електронну пошту на наявність листа для відновлення паролю.
4. Перейти за посиланням з листа.
5. Ввести новий пароль: "NewStrongPassword123".
6. Підтвердити новий пароль.

Очікуваний результат: Користувач отримує повідомлення про успішне відновлення паролю та може увійти в систему з новим паролем.

Тест-кейс №8 «Перевірка валідації полів»

Опис: перевірити валідацію полів на різних формах (реєстрація, авторизація, оформлення замовлення).

Передумови: Користувач знаходиться на відповідній сторінці.

Кроки:

1. Ввести некоректні дані (наприклад, невірний формат електронної адреси, слабкий пароль).
2. Натиснути кнопку підтвердження (зареєструватися, увійти, оформити замовлення).

Очікуваний результат: Користувач отримує відповідні повідомлення про помилки валідації.

Тест-кейс №9 «Перевірка безпеки паролів»

Опис: перевірити, що система не приймає слабкі паролі при реєстрації та зміні паролю.

Передумови: Користувач знаходиться на сторінці реєстрації або зміні паролю.

Кроки:

1. Ввести слабкий пароль (наприклад, "12345").
2. Підтвердити дію.

Очікуваний результат: Користувач отримує повідомлення про те, що пароль недостатньо безпечний, і вимагає введення більш складного паролю.

5.5 Перевірка роботи з замовленнями

Тест-кейс №10 «Попереднє замовлення товару до виходу в офіційний продаж»

Основний актор: Зареєстрований користувач.

Передумови:

1. Користувач повинен бути зареєстрованим та авторизованим у системі.
2. Товар повинен мати статус "Доступний для попереднього замовлення".

Основний сценарій:

1. Користувач переглядає каталог товарів.
2. Користувач обирає товар, доступний для попереднього замовлення.
3. Користувач додає товар до кошика.
4. Користувач переходить до оформлення замовлення.
5. Система пропонує обрати спосіб оплати та доставки.
6. Користувач вводить платіжні та контактні дані.
7. Система зберігає замовлення та надсилає підтвердження на електронну пошту користувача.

8. Після офіційного виходу товару в продаж, система автоматично обробляє замовлення та відправляє товар користувачеві.

Альтернативні сценарії:

1. Товар стає недоступним до моменту завершення попереднього замовлення. Система повідомляє користувача про зміни і пропонує альтернативні варіанти.
2. Користувач змінює рішення та видаляє товар з кошика перед завершенням замовлення.

Тест-кейс №11 «Система знижок»

Основний актор: Зареєстрований користувач, Гість.

Передумови: знижки повинні бути налаштовані в системі.

Основний сценарій:

1. Користувач переглядає каталог товарів зі знижками.
2. Користувач додає товар зі знижкою до кошика.
3. Користувач переходить до оформлення замовлення.
4. Система автоматично застосовує знижку до товарів у кошику.
5. Користувач бачить остаточну ціну зі знижкою.
6. Користувач вводить платіжні та контактні дані.
7. Система зберігає замовлення та надсилає підтвердження на електронну пошту користувача.

Альтернативні сценарії: знижка не застосовується через помилку системи. Система повідомляє користувача та пропонує звернутися до служби підтримки.

Тест-кейс №12 «Функція визначення розміру одягу»

Основний актор: Користувач (зареєстрований чи гість).

Передумови: Користувач має доступ до таблиці розмірів і інструментів для визначення розміру.

Основний сценарій:

1. Користувач переглядає товар.
2. Користувач переходить до інструменту визначення розміру.

3. Користувач вводить свої параметри (зріст, вага, об'єм грудей, талії тощо).
4. Система аналізує введені дані та пропонує оптимальний розмір для обраної моделі.
5. Користувач підтверджує вибір розміру та додає товар до кошика.
6. Користувач продовжує процес оформлення замовлення.

Альтернативні сценарії: Користувач не задоволений запропонованим розміром та обирає інший розмір вручну.

5.6 Результати тестування

Проведено тестування роботи інформаційної системи відповідно до створених тест-кейсів. Усі функціональні вимоги були виконані. У таблиці 5.1 представлені результуючі дані щодо кожного з сценаріїв тестування.

Таблиця 5.1 – Результати тестування

№ ТК	Назва тест-кейсу	Підтвердження очікуваного результату
1	Успішна реєстрація нового користувача	Підтверджено
2	Реєстрація з уже існуючою електронною адресою	Підтверджено
3	Успішна авторизація зареєстрованого користувача	Підтверджено
4	Авторизація з невірним паролем	Підтверджено
5	Оформлення замовлення зареєстрованим користувачем	Підтверджено
6	Оформлення замовлення гостем	Підтверджено
7	Успішне відновлення паролю	Підтверджено
8	Перевірка валідації полів	Підтверджено
9	Перевірка безпеки паролів	Підтверджено

Продовження таблиці 5.1

10	Попереднє замовлення товару до виходу в офіційний продаж	Підтверджено
11	Система знижок	Підтверджено
12	Функція визначення розміру одягу	Підтверджено

В даному розділі описано функціональне тестування інформаційної системи. Для цього етапу розробки ПЗ було створено 12 тест-кейсів, які описують основні варіанти використання для об’єкту розробки. Результати тестування демонструють гарну роботу інформаційної системи з онлайн продажу одягу.

ВИСНОВКИ

Для досягнення мети, а саме розробка та впровадження інформаційної системи для онлайн продажу одягу, яка забезпечить зручний, швидкий та безпечний процес покупки для користувачів, а також ефективне управління асортиментом і замовленнями, були виконані всі етапи розробки.

На етапі аналізу надано характеристику інформаційної системи з онлайн -продажу одягу, виконано порівняний аналіз існуючих рішень, за допомогою якого сформовані функціональні і нефункціональні вимоги до об’єкту розробки.

На етапі проектування виконана розробка плану робіт, обрано архітектуру для інформаційної системи, виконано проектування моделі даних і за допомогою UML-діаграм побудовані моделі поведінки системи та її компонентів. Крім цього, проведено аналіз інструментальних засобів розробки та обрані технології для реалізації фронтенду і бекенду, обрано систему управління базами даних MySQL.

Реалізовані основні функції для онлайн-продажу: розроблена структура сайту, наповнений контент, підключені функції сортування за декількома параметрами товару, додавання товару до категорії улюблених пропозицій, формування кошика та оформлення замовлення. Крім цього, реалізовано реєстрацію/авторизацію для користувачів.

Для підтвердження коректної роботи інформаційної системи були написані 12 тест-кейсів для функціонального тестування, перевірка яких демонструє гарну роботу онлайн магазину.

Відсутність адміністративної частини обмежує можливість управління контентом, користувачами та замовленнями. Це створює проблеми для підтримки та оновлення сайту. Як подальший розвиток ІС слід завершити реалізацію кабінету адміністратора, а саме додати інструменти для збору та аналізу даних про поведінку користувачів, продажі, ефективність маркетингових кампаній. Це допоможе приймати обґрунтовані бізнес-

рішення. Впровадити автоматизацію рутинних завдань, таких як обробка замовлень, управління запасами товарів, розсилки клієнтам тощо. Додати функції для підтримки клієнтів, такі як чати, можливість адміністраторів відповідати на відгуки та запити користувачів.

СПИСОК ДЖЕРЕЛ ПОСИЛАНЬ

1. Розробка інтернет магазину – основні характеристики та функціональність. URL: https://web24.pro/rozrobka-sajtiv-blog/rozrobka-internet-magazyn-osnovni-harakterystyky-ta-funkczionalnist/ (дата звернення 18.03.2024)
2. Що таке use case та для чого вони потрібні. URL: https://training.qatestlab.com/blog/technical-articles/what-is-a-use-case-and-what-are-they-for/ (дата звернення 18.03.2024)
3. Функціональні та нефункціональні вимоги. URL: https://www.guru99.com/uk/functional-vs-non-functional-requirements.html (дата звернення 24.03.2024)
4. 10 Major Requirements for E-commerce Website Built on Shopify and BigCommerce. URL: https://digitalsuits.co/blog/10-major-requirements-for-e-commerce-website/ (дата звернення 02.04.2024)
5. Create A Gantt Chart. URL: https://software.fish/project-management-software/gantt-chart-template (дата звернення 15.04.2024)
6. Мікросервісна архітектура. URL: https://foxminded.ua/mikroservisna-arkhitektura/ (дата звернення 22.04.2024)
7. Різниця між фронт і бекенд розробкою. URL: https://foxminded.ua/ru/front-end-back-end-raznica/ (дата звернення 02.05.2024)
8. React. URL: https://ru.legacy.reactjs.org/ (дата звернення 02.05.2024)
9. Що таке Vue? URL: https://ua.vuejs.org/guide/introduction.html (дата звернення 05.05.2024)
10. Стек для стартапів. URL: https://www.purrweb.com/ru/blog/tech-stack-dlya-startapov/ (дата звернення 11.05.2024)
11. ТОП-10 корисних інструментів для веб-мастерів. URL: https://kiev.itstep.org/ru/blog/top-10-useful-tools-for-webmasters (дата звернення 12.05.2024)

12. Django documentation. URL: https://docs.djangoproject.com/ (дата звернення 18.05.2024)
13. Django admin. URL: https://tutorial.djangogirls.org/en/django_admin/ (дата звернення 18.05.2024)
14. Що таке MySQL. URL: https://romul.name/mysql/ (дата звернення 21.05.2024)