ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка web-застосунку для магазину вечірніх суконь з автоматизованим підбором товару»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Таіра РУДСЬКА
(підпис)

Виконала: здобувачка вищої освіти групи ПД-41

Таіра РУДСЬКА

Керівник: Оксана ЗОЛОТУХІНА
к.т.н., доцент

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Рудській Таірі Анатоліївні

1. Тема кваліфікаційної роботи: «Розробка web-застосунку для магазину вечірніх суконь з автоматизованим підбором товару»
керівник кваліфікаційної роботи к.т.н., доцент Оксана ЗОЛОТУХІНА,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: офіційна документація фреймворків Spring Boot, Thymeleaf і MySQL, наукові джерела з персоналізації в e-commerce, матеріали з класифікації типів фігури та кольоротипів, приклади реалізації систем рекомендацій у fashion-індустрії.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд предметної галузі та аналіз персоналізованих підходів у сфері онлайн-продажу одягу.
2. Аналіз інструментів реалізації та архітектурного проєктування web-застосунку.

3. Програмна реалізація застосунку для автоматизованого підбору вечірніх суконь.
4. Візуалізація інтерфейсу та тестування функціональності web-застосунку.

5. Перелік графічного матеріалу: *презентація*
1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання (гість та користувач).
5. Діаграма варіантів використання (адміністратор).
6. Діаграма класів підбору товарів за типом фігури та кольоротипом.
7. Діаграма структури бази даних.
8. Екранні форми.
9. Апробація результатів дослідження.

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2025	
3	Огляд алгоритмів визначення типів фігури та кольоротипу, технологій побудови систем персоналізованих рекомендацій	14.03-21.03.2025	
4	Проєктування web-застосунку для онлайн-магазину вечірніх суконь	22.03-04.04.2025	
5	Програмна реалізація web-застосунку для магазину вечірніх суконь	05.04-30.04.2025	
6	Тестування web-застосунку	01.05-05.05.2025	
7	Оформлення роботи: вступ, висновки, реферат	06.05-11.05.2025	
8	Розробка демонстраційних матеріалів	12.05-18.05.2025	
9	Попередній захист роботи	19.05-30.05.2025	

Здобувачка вищої освіти _____ Таіра РУДСЬКА
(підпис)

Керівник
кваліфікаційної роботи _____ Оксана ЗОЛОТУХІНА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: _52_ стор., __2__ табл., __41__ рис., __24__ джерел.

Мета роботи – автоматизація процесу підбору моделей вечірніх суконь в онлайн-магазині з урахуванням індивідуальних параметрів користувача.

Об'єкт дослідження – процес підбору моделей вечірніх суконь для клієнтів магазину.

Предмет дослідження – програмне забезпечення для автоматизованого підбору моделей вечірніх суконь з урахуванням індивідуальних параметрів користувача.

Короткий зміст роботи: В роботі проаналізовано сучасні підходи до автоматизованого підбору одягу в сфері електронної комерції з урахуванням параметрів фігури та зовнішності користувача. Досліджено наявні системи персоналізованих рекомендацій, виявлено їх переваги та обмеження. Розроблено архітектуру web-застосунку для магазину вечірніх суконь із реалізацією функціоналу визначення типу фігури та кольоротипу на основі нечіткої логіки. Побудовано базу даних для зберігання користувачів, товарів і параметрів підбору, реалізовано REST-контролери та модулі безпеки за допомогою Spring Boot, Spring Security, Thymeleaf та MySQL. Система забезпечує фільтрацію товарів, індивідуальні рекомендації, а також персоналізовані кабінети користувачів і адміністратора. Проведено кросбраузерне тестування, юніт-тестування логіки підбору, протестовано інтерфейс і функціональність на різних пристроях.

Сферою використання застосунку є автоматизація процесу підбору вечірніх суконь у системах електронної комерції з урахуванням індивідуальних параметрів фігури та зовнішності покупців.

КЛЮЧОВІ СЛОВА: WEB-ЗАСТОСУНОК, ПЕРСОНАЛІЗОВАНИЙ ПІДБІР, ТИП ФІГУРИ, КОЛЬОРОТИП, НЕЧІТКА ЛОГІКА, SPRING BOOT.

ЗМІСТ

ВСТУП

Актуальність: у сучасних умовах стрімкого розвитку електронної комерції особливої популярності набувають онлайн-магазини, які пропонують не лише зручність покупки, а й персоналізований підхід до кожного клієнта. Сегмент продажу жіночого одягу, зокрема вечірніх суконь, потребує особливої уваги до індивідуальних параметрів покупця, таких як тип фігури та зовнішність. Відсутність можливості примірки в онлайн-магазинах призводить до великої кількості повернень товарів, що негативно впливає на ефективність роботи платформи та знижує рівень задоволеності користувачів. Саме тому актуальним є створення системи, яка б автоматизувала процес підбору товару на основі антропометричних і візуальних характеристик покупця.

Об'єкт дослідження – процес підбору моделей вечірніх суконь для клієнтів магазину.

Предмет дослідження – програмне забезпечення для автоматизованого підбору моделей вечірніх суконь з урахуванням індивідуальних параметрів користувача.

Мета роботи – автоматизація процесу підбору моделей вечірніх суконь в онлайн-магазині з урахуванням індивідуальних параметрів користувача.

Для досягнення поставленої мети в бакалаврській роботі виконано наступні задачі:

1. Провести аналіз існуючих рішень у сфері онлайн-продажу одягу, зокрема систем підбору товарів за параметрами фігури та кольоротипом користувача.
2. Визначити ключові антропометричні параметри для класифікації типів фігури та дослідити їхній вплив на вибір фасонів вечірніх суконь.
3. Розробити алгоритм автоматизованого підбору моделей відповідно до типу фігури та сезонного кольоротипу.

4. Сформувати базу даних для збереження користувачів, товарів і параметрів підбору.

5. Реалізувати web-застосунок для онлайн-магазину вечірніх суконь із функціоналом каталогу, фільтрації товарів, особистого кабінету та адміністративної панелі.

6. Провести тестування системи, перевірити точність підбору, зручність використання та загальну ефективність розробленого рішення.

Методи дослідження: аналіз наукових джерел у галузі електронної комерції, класифікації типів фігури, методів визначення кольоротипу; порівняльний аналіз існуючих web-платформ; методи системного аналізу, проєктування програмного забезпечення та побудови інформаційних систем; а також практичні методи розробки і тестування web-застосунків на базі фреймворку Spring Boot.

Наукова новизна роботи полягає у поєднанні методів персоналізованого підбору одягу з автоматизованим визначенням типу фігури та кольоротипу користувача для генерації точних рекомендацій у межах одного web-застосунку. У межах кваліфікаційної роботи запропоновано алгоритм, що використовує нечітку логіку для визначення типу фігури на основі антропометричних даних, а також підбір суконь з урахуванням особливостей зовнішності за сезонною класифікацією.

Практична значущість результатів полягає у створенні прототипу web-застосунку, який буде використаний реальною компанією, що працює у сфері fashion e-commerce. Застосунок дозволяє зменшити кількість повернень, підвищити точність вибору одягу та забезпечити користувачам більш зручний, інформативний і персоналізований досвід онлайн-покупок.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Характеристика предметної галузі

Сфера електронної комерції в сегменті продажу жіночого одягу, зокрема вечірніх суконь, активно розвивається як в Україні, так і за її межами. Онлайн-магазини пропонують широкий вибір моделей, однак відсутність можливості фізичної примірки створює низку проблем — високу частку повернень, невдоволення клієнтів, зниження лояльності до бренду. Особливо критичним є питання точного підбору товарів відповідно до індивідуальних параметрів тіла та зовнішності покупця.

Одним із напрямів вирішення цієї проблеми є впровадження інтелектуальних рекомендаційних систем, що використовують дані про тип фігури, кольоротип, розмір та уподобання користувача [1]. Автоматизовані підказки дозволяють не лише покращити досвід покупця, а й знизити кількість повернень, збільшити конверсію та покращити імідж онлайн-платформи [3].

Предметна галузь охоплює міждисциплінарну область, яка включає елементи e-commerce, UX-дизайну, машинного навчання та обробки антропометричних даних. Особливу актуальність мають технології, що дозволяють на основі вхідних параметрів (обхватів, кольору очей, волосся, тону шкіри) автоматично формувати персоналізовані рекомендації [2; 4; 7].

1.2 Аналіз існуючих аналогів для автоматизованого підбору товарів

У сфері онлайн-продажу жіночого одягу, зокрема вечірніх суконь, спостерігається підвищений попит на інноваційні рішення, що забезпечують персоналізований підбір товарів. У цьому розділі проведено аналіз існуючих програмних рішень, які частково або повністю реалізують функціональність, аналогічну до тієї, що розробляється у межах даної бакалаврської роботи. Зокрема, розглянуто платформи Oksana Mukha [8], Rozetka [9] та Kasta [10].

1.2.1 Аналіз платформи Oksana Mukha

Oksana Mukha — український бренд дизайнерських весільних і вечірніх суконь, представлений на ринку понад 20 років. Web-сайт бренду виконаний у сучасному стилі з акцентом на візуальну складову. Каталог товарів представлено у вигляді галереї з якісними зображеннями, і містить базову фільтрацію за категорією (вечірні, весільні сукні), кольором, довжиною та типом тканини. Форма подання дозволяє користувачу швидко переглядати колекції, однак взаємодія з платформою носить односторонній характер — уся логіка взаємодії базується на виборі зі списку або перегляді зображень.

Для вирішення завдань, пов’язаних із автоматизованим підбором товару за параметрами фігури, кольоротипом або індивідуальними вподобаннями користувача, web-платформа Oksana Mukha функціонально не підходить. Вона не підтримує введення анкетних або фізичних даних користувача, не реалізує алгоритмів аналізу типу фігури (наприклад, через вимірювання ліній плечей, талії, стегон), не пропонує визначення кольоротипу за ознаками зовнішності (тон шкіри, очі, волосся), та не містить рекомендаційного механізму для автоматичного добору моделі сукні.

Окремо варто зазначити, що на сайті відсутня функція рекомендованого розміру: користувачу не пропонується ввести свої параметри для отримання персональної поради щодо найбільш ймовірного варіанта. Розмірна сітка надається у вигляді таблиці, яку користувач змушений самостійно зіставляти зі своїми обхватами.

Також відсутні можливості збереження історії переглянутих товарів, що унеможливлює повторне повернення до вподобаних моделей без попередніх дій із боку користувача (наприклад, запису або закладок у браузері). Крім того, на платформі загалом відсутній особистий кабінет користувача**,** а також не передбачено реєстрації покупців. Форма реєстрації на сайті доступна лише для тих, хто бажає стати партнером бренду, наприклад дистриб’ютором або власником шоуруму. Це ще більше обмежує персоналізовану взаємодію з кінцевими споживачами в онлайн-середовищі.

До переваг web-платформи Oksana Mukha можна віднести:

– високу якість візуального представлення товару — професійні фото, наближені до лукбуків;

– естетичний та логічно структурований інтерфейс, орієнтований на преміум-сегмент;

– зручну фільтрацію по базових категоріях (колір, довжина, колекція).

Недоліки, що впливають на ефективність для завдань, що вирішуються в межах бакалаврської роботи:

– відсутність будь-якої персоналізації процесу підбору сукні;

– неможливість автоматичного визначення розміру на основі параметрів користувача;

– відсутність системи підбору за типом фігури та кольоротипом;

– немає інтерактивного інтерфейсу взаємодії з користувачем (наприклад, чат-ботів, віртуальних примірок, рекомендацій на основі переглядів або історії);

– відсутність реєстрації для звичайних покупців.

Приклад каталогу товарів та приклад картки товару без рекомендаційного функціоналу зображено на рисунках 1.1 та 1.2 відповідно.

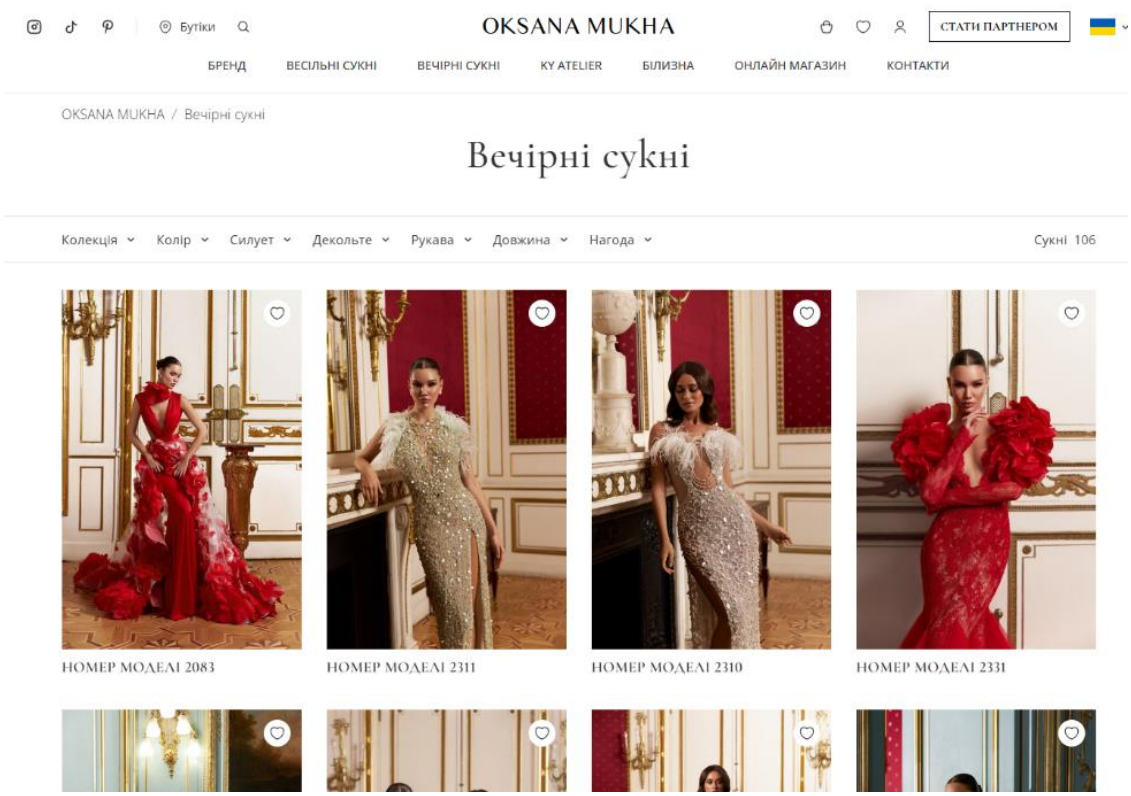


Рис. 1.1 Головна сторінка каталогу товарів на сайті oksana-mukha

Рис. 1.2 Приклад картки товару без рекомендаційного функціоналу

1.2.2 Аналіз платформи Rozetka

Rozetka — один із найбільших українських маркетплейсів, який надає користувачам широкий спектр товарів, у тому числі жіночий одяг. Розділ одягу, зокрема вечірніх суконь, представлений у форматі багаторівневого каталогу з розширеною фільтрацією, персональним кабінетом, відгуками, а також рядом функцій, що підвищують зручність вибору товару.

На сайті Rozetka, каталог вечірніх суконь містить фільтрацію за категорією, брендом, кольором, розміром, матеріалом, фасоном, ціною, стилем, країною виробника тощо. Це дозволяє звузити вибір і полегшити навігацію.

Однією з важливих функцій, яка вирізняє Rozetka серед конкурентів, є визначення рекомендованого розміру. При перегляді товару користувач має можливість натиснути на кнопку «Не впевнені який розмір обрати», після чого відкривається форма введення параметрів (зріст, обхват грудей, обхват талії, обхват стегон). На основі цих даних система формує рекомендацію щодо оптимального розміру. Подібні підходи описані в науковій літературі [2], де розглядається застосування алгоритмів машинного навчання для прогнозування відповідного розміру та зменшення кількості повернень.

Ще однією перевагою є відображення рекомендованого розміру безпосередньо в інтерфейсі вибору: після введення параметрів поряд із доступними

розмірами з’являється позначка із зазначенням рекомендованого варіанту. Це допомагає користувачеві швидше зорієнтуватися у виборі, покращує взаємодію з сайтом та зменшує ймовірність помилки.

Також, Rozetka реалізує функцію контролю залишків товару за кожним розміром: якщо конкретний розмір відсутній на складі, кнопка додавання до кошика блокується, а система пропонує повідомлення про надходження. Це важливий аспект для реалізації ефективної логістики та уникнення ситуацій з «фантомними» товарами.

Платформа підтримує особистий кабінет користувача, де зберігається історія замовлень, переглядів, вподобань. Завдяки цьому користувач може швидко повернутися до раніше переглянутих моделей або повторити замовлення. Саме такі елементи персоналізації — збереження активності користувача, надання рекомендацій на основі його дій — суттєво підвищують конверсію, сприяють зростанню задоволеності клієнтів та зменшенню кількості відмов і повернень [3].

Проте, незважаючи на широкі функціональні можливості, Rozetka не реалізує ключових елементів, пов’язаних із інтелектуальним підбором товару за типом фігури чи кольоротипом. Користувач не має можливості ввести параметри фігури (лінії плечей, талії, стегон) для автоматичного визначення типу силуету (наприклад, "пісочний годинник", "груша" тощо), що значно обмежує точність візуального підбору. Хоча у дослідженнях [2], розроблено ефективні підходи до рекомендацій за фізичними параметрами тіла, реалізація таких технологій на платформі Rozetka відсутня.

Rozetka частково задовольняє завдання, що розглядаються у межах бакалаврської роботи: система підбору розміру та управління залишками дійсно може бути корисною при реалізації власного застосунку. Однак відсутність глибинної персоналізації (аналіз фігури, кольоротипу) робить цю платформу неповноцінним рішенням для високоточних fashion-рекомендацій, орієнтованих на індивідуальні особливості покупця.

Переваги платформи Rozetka:

– реалізовано систему підбору рекомендованого розміру;
– автоматичне підсвічування рекомендованого розміру;
– контроль залишків товару за розмірами;
– особистий кабінет з історією замовлень і переглядів;
– зручна фільтрація та адаптивний інтерфейс.

Недоліки:

– відсутність автоматизованого підбору за типом фігури;
– відсутність аналізу кольоротипу;
– немає взаємодії з користувачем у форматі інтелектуального помічника або стиліста.

Каталог вечірніх суконь зображено на рисунку 1.3.

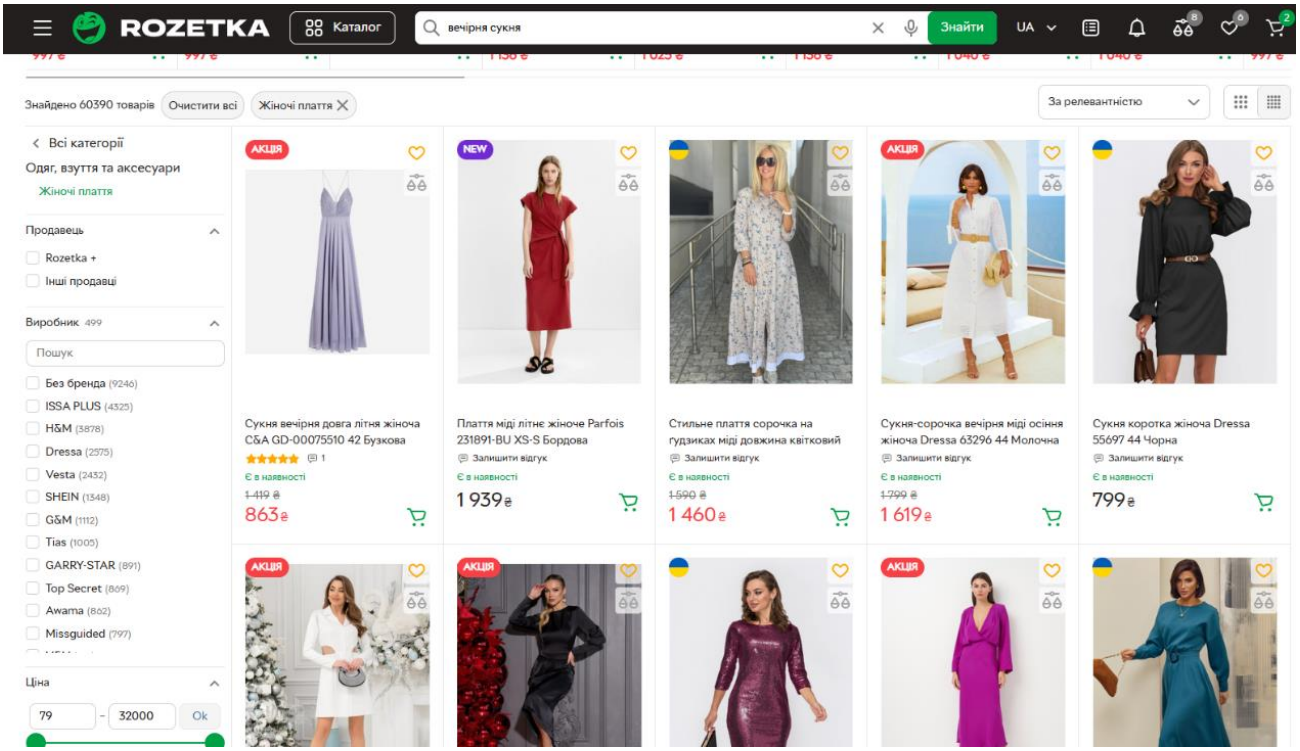


Рис. 1.3 Каталог вечірніх суконь на сайті Rozetka

Приклади форми підбору розміру, сторінки товару з підсвіченим рекомендованим розміром та вікно повідомлення про відсутність певної кількості розміру зображено на рисунках 1.4 - 1.6.

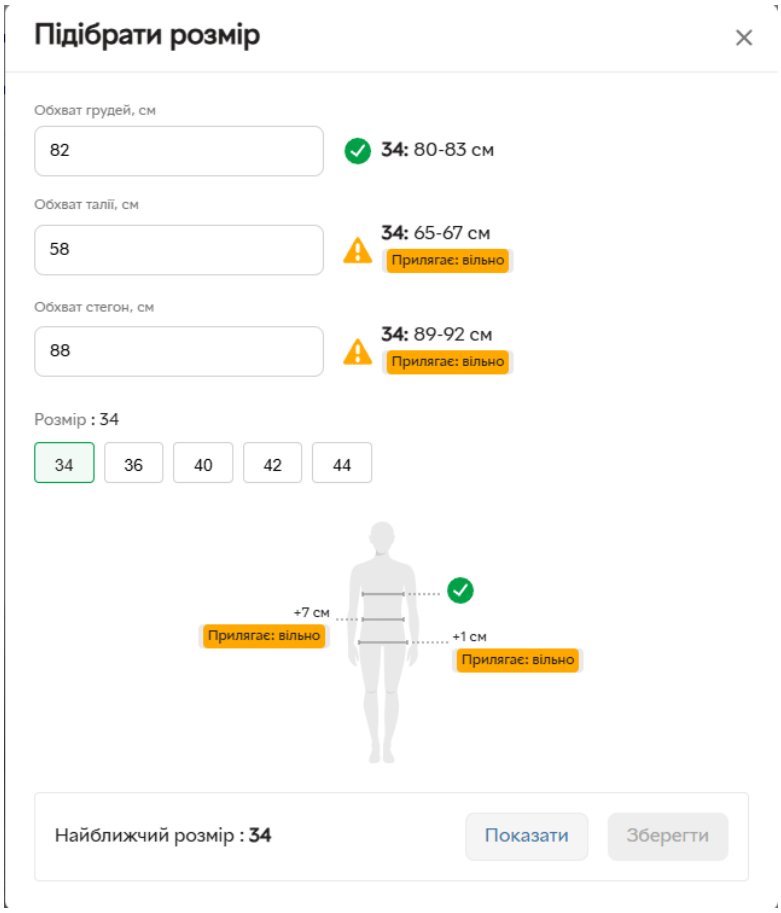


Рис. 1.4 Форма підбору розміру на сайті Rozetka

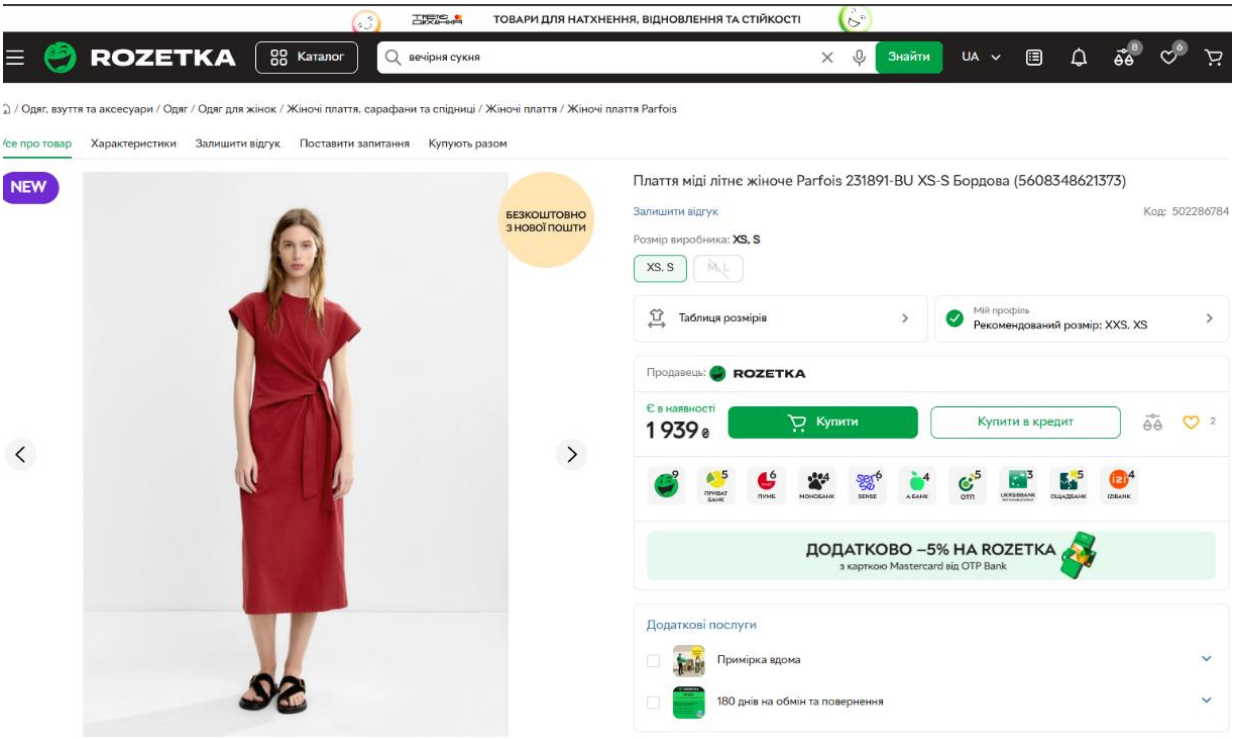


Рис. 1.5 Сторінка товару з підсвіченим рекомендованим розміром

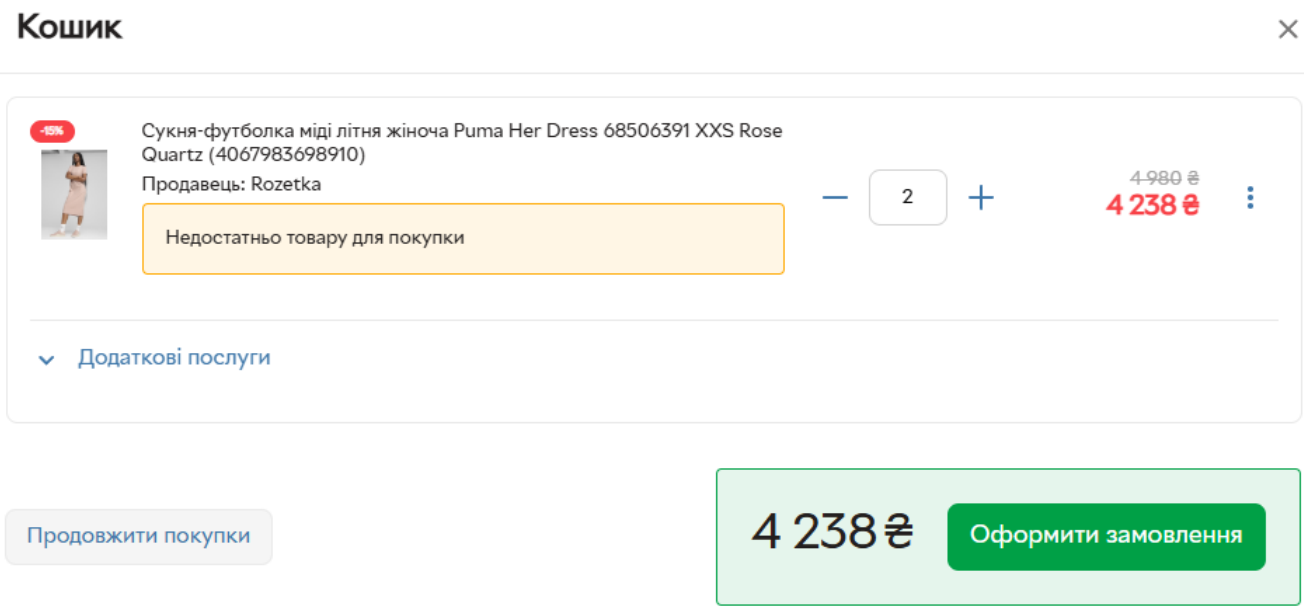


Рис. 1.6 Вікно повідомлення про відсутність певної кількості розміру

1.2.3 Аналіз платформи Kasta

Kasta — один із провідних українських онлайн-ретейлерів у сфері fashion e-commerce, що пропонує широкий асортимент товарів: одяг, взуття, аксесуари, косметику, товари для дому тощо. Інтерфейс сайту Kasta побудований за принципами класичного онлайн-магазину: у користувача є доступ до багаторівневого каталогу з розширеною системою фільтрації — за категорією товару, брендом, кольором, розміром, ціною, країною виробника, типом тканини та іншими параметрами. У розділі жіночого одягу, включно з вечірніми сукнями, фільтри дозволяють вручну обрати бажані характеристики виробу, що полегшує навігацію при великому асортименті.

Однак усі фільтри працюють виключно вручну — користувачеві необхідно самостійно визначити свій розмір, колір, фасон тощо. Платформа не має індивідуального підбору товарів за параметрами тіла (плечі, талія, стегна), не визначає тип фігури (наприклад, “прямокутник”, “пісочний годинник” тощо), не пропонує рекомендований розмір на основі введених даних і не використовує алгоритми персоналізації на основі зовнішності (кольоротипу).

Функціональність особистого кабінету на платформі є базовою — він містить дані про замовлення, адреси доставки, платежі. Однак немає історії переглядів,

персональних рекомендацій, які б формувалися на основі активності користувача. Це обмежує глибину персоналізації, що, згідно з дослідженнями [1][3], є критично важливим фактором для збільшення лояльності покупців у сфері онлайн-моди.

Платформа Kasta підходить для ручного підбору товарів, але не вирішує завдання автоматизації процесу персоналізованого підбору суконь. Відсутність рекомендаційного модуля, алгоритмів визначення типу фігури та аналізу зовнішності суттєво знижує ефективність сайту для задач індивідуального підбору в fashion-сегменті.

Переваги платформи Kasta:

- зручна багаторівнева фільтрація товарів;
- автоматичне блокування кнопки купівлі при відсутності товару у вибраному розмірі;
- регулярне оновлення асортименту та акції.

Недоліки:

- повна відсутність інтелектуального підбору за параметрами фігури або зовнішності;
- немає рекомендованого розміру;
- особистий кабінет не зберігає історію переглядів або вподобань;
- відсутня взаємодія з користувачем у форматі помічника або стиліста.

Приклади каталогу вечірніх суконь на сайті Kasta.ua та приклад картки товару без рекомендаційного функціоналу зображено на рисунках 1.7 та 1.8 відповідно.

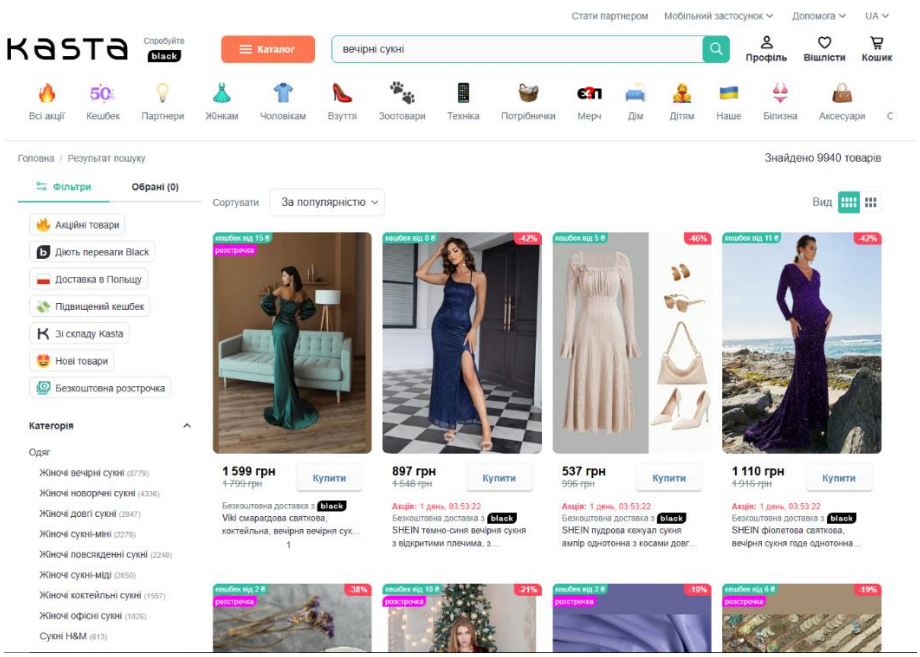


Рис. 1.7 Каталог вечірніх суконь на сайті Kasta.ua

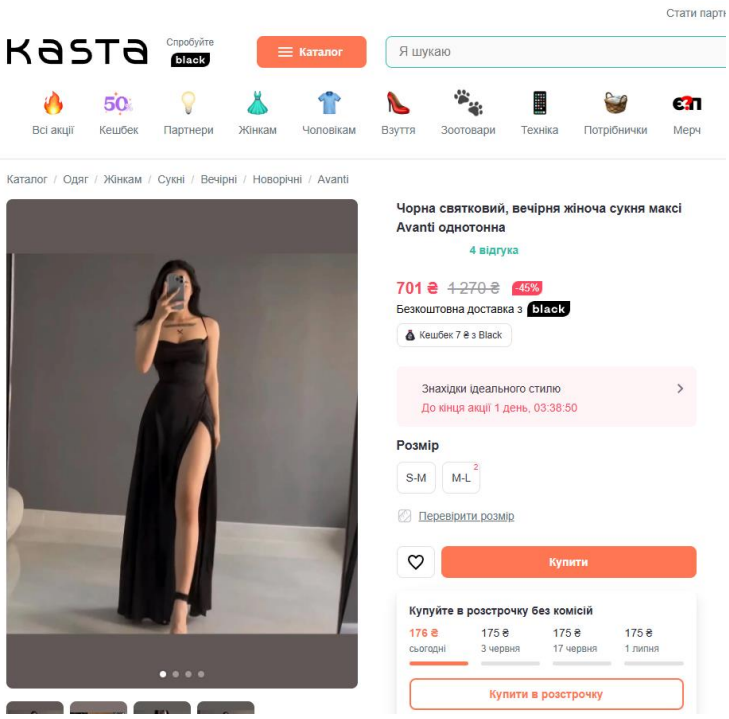


Рис. 1.8 Приклад картки товару без рекомендаційного функціоналу

Зведена оцінка можливостей персоналізації та функціоналу web-додатків наведено у таблиці 1.1.

Таблиця 1.1

Зведена оцінка можливостей персоналізації та функціоналу web-додатків

Функціональність/Показник	**Oksana Mukha**	**Rozetka**	**Kasta**
Наявність web-платформи	+	+	+
Каталог товарів з фільтрацією (за кольором, категорією)	+	+	+
Підбір товару за типом фігури (автоматичний аналіз)	-	-	-
Підбір товару за кольоротипом (тон шкіри, очі, волосся)	-	-	-
Рекомендований розмір на основі введених параметрів	-	+	-
Підсвічування рекомендованого розміру на сторінці товару	-	+	-
Підрахунок залишку товару за розміром та блокування кнопки	-	+	+
Особистий кабінет користувача з історією перегляду товарів	-	+	-

1.3 Визначення проблем предметної галузі

У процесі аналізу предметної галузі та вивчення особливостей існуючих платформ, що реалізують онлайн-продаж жіночого одягу, було виявлено низку типових проблем, які стримують розвиток персоналізованих сервісів у сфері fashion e-commerce та знижують якість користувацького досвіду.

Насамперед, однією з головних проблем є відсутність інтелектуального підбору товарів на основі параметрів фігури. Переважна більшість онлайн-

магазинів пропонує лише базову фільтрацію за розміром, без урахування особливостей силуету (наприклад, "пісочний годинник", "прямокутник", "груша" тощо), хоча саме форма тіла відіграє ключову роль у виборі фасону сукні. Врахування типу фігури дозволяє підвищити ймовірність того, що обрана модель буде відповідати очікуванням покупця як за посадкою, так і за стилістичним сприйняттям [6; 11].

Іншою важливою проблемою є ігнорування кольоротипу користувача під час вибору товарів. Кольоротип (поєднання тону шкіри, кольору очей і волосся) впливає на те, як певні відтінки виглядають на людині, однак ця характеристика практично не використовується в українських web-магазинах. Натомість іноземні системи, такі як ViBE, уже починають інтегрувати механізми візуального аналізу та персональних рекомендацій з урахуванням зовнішності [11].

Також спостерігається відсутність комплексної системи персоналізації. Наприклад, лише одна з аналізованих платформ — Rozetka — реалізує механізм визначення рекомендованого розміру на основі параметрів (ог, от, ос), але й там немає врахування співвідношень, фігури чи кольоротипу. У більшості інших випадків розмірна сітка подається у вигляді таблиці, що потребує ручного аналізу з боку користувача, що часто призводить до помилок і невідповідностей.

Крім цього, до ключових проблем галузі належать:

- відсутність інтерактивної взаємодії з користувачем: немає чат-ботів, стилістів-помічників або рекомендаційних блоків, що формуються динамічно;
- відсутність накопичення персональних даних користувача для подальшого аналізу та побудови профілю покупця;
- низький рівень гнучкості та адаптивності фільтрів — навіть розширена фільтрація обмежується категоріями, кольором, брендом, але не підтримує складні запити (наприклад, "моделі для фігури типу 'груша' у холодному відтінку червоного").

Застосування алгоритмів глибокого навчання, класифікації фігур та кольорової персоналізації дозволяє досягти значного зниження кількості повернень (до 25–30%) та збільшення середнього чека, особливо у сегменті вечірнього одягу, який вимагає високої відповідності очікуванням покупця [4; 7].

Отже, можна виділити основні напрями, які потребують вирішення в межах розробки нового web-застосунку:

- реалізація механізму автоматичного визначення типу фігури на основі антропометричних параметрів;
- визначення кольоротипу користувача за допомогою класифікації візуальних характеристик;
- поєднання обох параметрів для формування релевантних рекомендацій;
- підтримка особистого кабінету з історією вибору та попередніми параметрами користувача.

Ці проблеми і визначають обґрунтовану потребу в розробці персоналізованої системи, яка забезпечуватиме якісно новий рівень підбору вечірніх суконь в онлайн-середовищі.

1.4 Постановка завдання на розробку системи

На основі проведеного аналізу предметної галузі, вивчення наукових джерел [1–4; 6–7], а також результатів порівняльного аналізу існуючих аналогів (платформ Oksana Mukha, Rozetka, Kasta), було виявлено низку проблем, які на сьогодні не вирішуються в межах більшості українських web-рішень. Ці проблеми стосуються насамперед низького рівня персоналізації при підборі одягу, відсутності автоматизованого аналізу типу фігури та кольоротипу, а також обмеженої інтерактивності користувацької взаємодії з платформою.

У межах цієї кваліфікаційної роботи ставиться завдання створити web-застосунок для онлайн-магазину вечірніх суконь, який би забезпечував персоналізований підбір моделей на основі індивідуальних особливостей користувача — зокрема параметрів фігури (лінії плечей, талії та стегон) та характеристик зовнішності (тон шкіри, колір очей і волосся).

Для реалізації поставленої задачі передбачається:

- реалізація модуля визначення типу фігури на основі введених параметрів із застосуванням нечіткої логіки та правил, що враховують співвідношення між параметрами;

- розробка алгоритму визначення кольоротипу користувача на основі ознак зовнішності за сезонною класифікацією (зима, весна, літо, осінь);
- формування блоку персоналізованих рекомендацій, який буде автоматично пропонувати користувачеві ті моделі суконь, що найкраще відповідають його фігурі та кольоротипу;
- інтеграція функціоналу класичного онлайн-магазину: каталог, фільтрація за категоріями, розмірами, кольорами, сторінка товару, кошик, оформлення замовлення;
- створення особистого кабінету користувача, у якому зберігатиметься історія замовлень, введені параметри, обрані моделі;
- реалізація адміністративної панелі з можливістю керування товарами, їх категоризацією за типами фігури та кольоротипами, оновленням контенту;
- використання сучасного технологічного стеку, зокрема Java Spring Boot, Thymeleaf, MySQL, HTML/CSS, що дозволяє реалізувати безпечний, масштабований і продуктивний web-застосунок;
- забезпечення захисту персональних даних користувача та інтеграція механізмів безпеки через Spring Security.

Поставлене завдання передбачає комплексний підхід до реалізації системи, яка поєднує інструменти електронної комерції та інтелектуальних рекомендацій. Особлива увага буде приділена користувацькому досвіду (UX), зручності взаємодії, швидкості відповіді системи та точності запропонованих моделей.

У результаті реалізації проєкту очікується створення повнофункціонального web-застосунку, здатного якісно підвищити рівень персоналізації онлайн-покупок у сегменті вечірнього жіночого одягу.

1.5 Визначення вимог до застосунку

На основі попереднього аналізу предметної галузі, виявлених проблем та сформульованої мети проєкту, сформульовано вимоги до функціональності та якості розроблюваного web-застосунку для автоматизованого підбору вечірніх суконь відповідно до параметрів користувача. Усі вимоги поділено на функціональні та нефункціональні.

Функціональні вимоги:

1. Реєстрація та авторизація користувачів із чітким розподілом ролей (користувач та адміністратор).
2. Введення параметрів тіла та зовнішності (лінії плечей, талії та стегон, колір очей, волосся та тон шкіри) для подальшої обробки.
3. Автоматичне визначення типу фігури з використанням алгоритму на базі нечіткої логіки.
4. Автоматичне визначення кольоротипу користувача за введеними характеристиками зовнішності.
5. Формування персоналізованого списку товарів, рекомендованих користувачу на основі його фігури та кольоротипу.
6. Доступ до каталогу товарів з можливістю фільтрації за категоріями, кольором, ціною та розміром.
7. Особистий кабінет користувача, у якому зберігається історія замовлень, введені параметри та переглянуті товари.
8. Адміністративна панель, яка надає можливість адміністратору додавати, редагувати та видаляти товари, переглядати користувачів і замовлення.

Нефункціональні вимоги:

1. Контроль доступу до адміністративного інтерфейсу, обмежений лише для користувачів з роллю адміністратора.
2. Забезпечення безпеки облікових записів шляхом шифрування паролів за допомогою bcrypt-алгоритму.
3. Оптимізація швидкодії: час завантаження сторінок не повинен перевищувати 2 секунд навіть при навантаженні.
4. Кросбраузерна сумісність — коректне відображення та робота інтерфейсу у сучасних версіях браузерів (Google Chrome, Mozilla Firefox, Safari).

Визначені вимоги ляжуть в основу подальшого етапу проєктування системи, побудови її архітектури та реалізації функціональних модулів. Вони спрямовані на забезпечення високого рівня персоналізації, безпеки, зручності використання та стабільності web-застосунку.

2 АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ ТА СТРУКТУРИ ПРОГРАМНОЇ СИСТЕМИ

2.1 Середовище розробки

У процесі створення web-застосунку для автоматизованого підбору вечірніх суконь було обрано середовище розробки IntelliJ IDEA — професійну інтегровану середу від компанії JetBrains, яка є одним із найпопулярніших рішень для Java-розробки в індустрії. Це середовище забезпечує високу продуктивність розробника, широкі можливості інтеграції з фреймворками та бібліотеками, а також зручні інструменти налагодження, рефакторингу та роботи з проєктами великого обсягу.

IntelliJ IDEA підтримує роботу з найпоширенішими технологіями сучасного Java-стека: Spring Boot, Spring Security, Thymeleaf, Maven, Gradle, JPA/Hibernate, а також системами контролю версій, такими як Git. Завдяки інтеграції зі Spring Initializr, розробник може швидко згенерувати структуру проєкту з урахуванням необхідних залежностей, що значно економить час на початкових етапах.

До основних переваг IntelliJ IDEA у контексті даного проєкту належать:

- автоматична генерація класів, конфігурацій та шаблонів. IDE пропонує готові шаблони коду та допомагає уникнути типових помилок під час написання бізнес-логіки або взаємодії з базою даних.
- інтелектуальне автодоповнення коду. IntelliJ здійснює глибокий аналіз структури коду, типів змінних та викликів методів, завдяки чому значно підвищується швидкість написання коду.
- вбудований SQL-редактор. Можна напряму працювати з базами даних MySQL із середовища, переглядати таблиці, виконувати запити та налагоджувати схему БД.
- налагодження (debugging) у реальному часі. Застосунок можна запускати у відладочному режимі, зупиняти виконання в певних точках (breakpoints) та поетапно відстежувати змінні.

– інструменти перевірки якості коду. Вбудовані механізми аналізу (inspections) сигналізують про потенційні помилки, дублювання, надмірності чи порушення стилю.

Середовище IntelliJ IDEA виступає основою для ефективної та швидкої розробки web-застосунків у галузі електронної комерції [12].

2.2 Вибір мови програмування та серверного фреймворку

Для реалізації серверної частини web-застосунку було обрано мову програмування Java у поєднанні з фреймворком Spring Boot [13]. Java є однією з найпоширеніших мов у галузі web-розробки, особливо для побудови масштабованих і стабільних систем. Вона підтримує об’єктно-орієнтовану модель, має сильну типізацію та багату екосистему інструментів і бібліотек, що робить її зручною для реалізації складної бізнес-логіки.

Spring Boot, у свою чергу, дозволяє значно пришвидшити розробку web-застосунків завдяки автоматичному налаштуванню, підтримці шаблонного коду, вбудованому серверу та зручній інтеграції з іншими компонентами Spring. Фреймворк дає змогу швидко створити REST API, налаштувати взаємодію з базами даних через Spring Data JPA, реалізувати рольову автентифікацію за допомогою Spring Security, а також організувати клієнт-серверну архітектуру.

У межах цього проєкту Spring Boot забезпечує роботу серверної логіки для визначення типу фігури та кольоротипу, обробку запитів від користувача, взаємодію з базою даних, а також забезпечення безпеки доступу. Інструменти Spring дозволяють дотримуватись принципів модульності, підтримувати чисту архітектуру проєкту та спростити супровід програмного коду.

Комбінація Java та Spring Boot є оптимальним вибором для реалізації функціонально насиченого та надійного web-застосунку в галузі електронної комерції, орієнтованого на персоналізований підбір товарів.

2.3 Реалізація безпеки системи

Для захисту персональних даних користувачів у системі використано бібліотеку Spring Security [14], яка забезпечує комплексну реалізацію безпеки. Основним механізмом автентифікації є вхід за email та паролем, причому паролі зберігаються у базі у вигляді bcrypt-хешів, що значно підвищує стійкість до атак. Як відбувається хешування зображено на рисунку 2.1.

Рис. 2.1 Хешування паролю за допомогою bcrypt-алгоритму

Система підтримує авторизацію на основі ролей, зокрема розмежування прав доступу між користувачами (*ROLE_USER*) та адміністраторами (*ROLE_ADMIN*), що дозволяє захистити адміністративні функції від несанкціонованого доступу.

Крім того, Spring Security реалізує захист від CSRF-атак, автоматично додаючи токени до форм, а також дозволяє керувати користувацькими сесіями, обмежуючи кількість одночасних входів і час активності.

Використання цієї бібліотеки відповідає сучасним стандартам web-безпеки та гарантує надійний захист системи на всіх рівнях взаємодії з користувачем.

2.4 Клієнтська частина та шаблонізація

Інтерфейс розроблено з використанням HTML5, CSS3, JavaScript та фреймворку Bootstrap. Для побудови динамічних сторінок використано серверний шаблонізатор Thymeleaf, який забезпечує:

- зручне двостороннє зв’язування між HTML і Java-об’єктами;
- перевірку структури шаблону ще на етапі компіляції;

– підтримку фрагментів та умовного виводу елементів [15].

Завдяки цим властивостям вдалося створити адаптивний та логічно організований інтерфейс з інтерактивним функціоналом.

2.5 Робота з базою даних

У якості системи управління базами даних у проєкті використано MySQL — популярну реляційну СУБД, яка забезпечує надійність і високу продуктивність. MySQL підтримує транзакції, цілісність зв’язків між таблицями та ефективну індексацію, що критично важливо для e-commerce платформ [16].

Доступ до БД реалізується через Spring Data JPA, що працює поверх Hibernate. Це дозволяє автоматизувати збереження об’єктів Java у таблицях MySQL, уникнути написання SQL-запитів вручну та спростити взаємодію з базою.

Для розгортання локального середовища застосовується Open Server [17] — інструмент, який включає готові збірки Apache, MySQL, PHP і дозволяє швидко запускати проєкт без складної конфігурації. Він забезпечує гнучке керування сервісами та зручний доступ до логів і налаштувань сервера.

Для адміністрування БД використовується phpMyAdmin [18] — web-інтерфейс для управління MySQL. Через нього здійснюється створення таблиць, перегляд записів, резервне копіювання та виконання SQL-запитів. Це особливо корисно під час розробки та тестування, коли потрібен візуальний контроль над даними [16].

Зв’язка MySQL + Spring Data JPA + Open Server + phpMyAdmin створює гнучке середовище для збереження та обробки даних, що відповідає вимогам сучасного web-застосунку.

Діаграма взаємодії компонентів бази даних у системі наведено на рисунку 2.2.

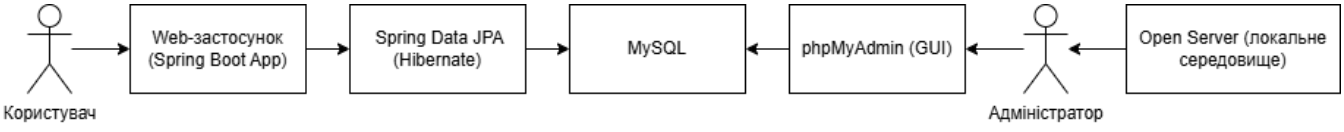


Рис. 2.2 Діаграма взаємодії бази даних у системі

3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-ЗАСТОСУНКУ

3.1 Загальна архітектура програмного забезпечення

Розроблений web-застосунок для автоматизованого підбору вечірніх суконь реалізовано на основі класичної клієнт-серверної архітектури, яка забезпечує чітке розділення відповідальностей між користувацьким інтерфейсом, бізнес-логікою та рівнем доступу до даних.

Система складається з трьох основних логічних рівнів:

- Клієнтський рівень (frontend) відповідає за взаємодію з користувачем. Він реалізований за допомогою HTML5, CSS3, JavaScript та фреймворку Bootstrap. Для рендерингу HTML-сторінок на стороні сервера застосовується шаблонізатор Thymeleaf, який дозволяє динамічно передавати об'єкти Java у вигляді HTML-елементів, що значно спрощує побудову інтерактивного інтерфейсу [15].
- Серверний рівень (backend) реалізовано з використанням Java та фреймворку Spring Boot, який надає зручне середовище для створення REST-контролерів, обробки запитів, реалізації бізнес-логіки та взаємодії з базою даних [13]. У цьому шарі реалізовані модулі автентифікації, підбору за фігурою та кольоротипом, управління каталогом товарів, обробки замовлень тощо.
- База даних (data access layer) базується на реляційній СУБД MySQL, інтеграція з якою здійснюється через Spring Data JPA — модуль, що працює поверх ORM-рішення Hibernate. Це дозволяє будувати об’єктно-орієнтовану модель доступу до даних без написання SQL-запитів вручну, що пришвидшує розробку та зменшує кількість помилок [16].

Комунікація між клієнтом і сервером відбувається за допомогою HTTP-запитів, що обробляються контролерами Spring. У відповідь сервер генерує HTML-сторінки або JSON-об’єкти (у випадку REST-запитів), які передаються клієнту.

Для захисту системи реалізовано Spring Security — компонент, який відповідає за автентифікацію користувачів, авторизацію за ролями, шифрування паролів, захист сесій та CSRF-захист [14].

Застосунок має модульну структуру, що спрощує супровід і розширення системи, дозволяє масштабувати функціональність та адаптувати її під потреби бізнесу без перегляду базової архітектури.

Архітектура клієнт-серверної взаємодії web-застосунку зображена на рисунку 3.1.

Рис. 3.1 Діаграма архітектури клієнт-серверної взаємодії web-застосунку

3.2 Сценарії взаємодії користувача із системою

У межах розробки web-застосунку для автоматизованого підбору вечірніх суконь була створена діаграма варіантів використання, яка ілюструє основні сценарії взаємодії користувача з системою.

Користувач має змогу ознайомитися з каталогом товарів, використовуючи фільтрацію за категоріями, розміром, кольором, ціною та іншими характеристиками. Після перегляду детальної інформації про обраний товар, він може визначити та обрати розмір, додати товар до кошика або до списку бажаного (вішлісту), а згодом оформити замовлення. Одним із ключових сценаріїв взаємодії є індивідуальний підбір сукні, який охоплює введення параметрів фігури (лінії плечей, талії та стегон) для визначення типу силуету, а також введення зовнішніх характеристик (тон шкіри, колір очей і волосся) з метою виявлення кольоротипу.

На основі цих параметрів система автоматично генерує добірку рекомендованих моделей, найбільш придатних для користувача.

Окрім цього, авторизований користувач отримує доступ до особистого кабінету, де може переглядати історію замовлень та переглянутих товарів, змінювати особисті дані, редагувати введені параметри фігури та зовнішності, оновлювати пароль і завантажувати аватар. Усі ці дії спрямовані на створення максимально персоналізованого досвіду онлайн-покупок та підвищення зручності використання системи.

Діаграма варіантів використання демонструє гнучку логіку поведінки користувача в системі, враховуючи як звичні функції інтернет-магазину, так і нові елементи, пов’язані з автоматизованим підбором суконь, що робить застосунок сучасним, зручним і орієнтованим на індивідуальні потреби кожного покупця.

На рисунку 3.2 наведено діаграма варіантів використання користувачем.

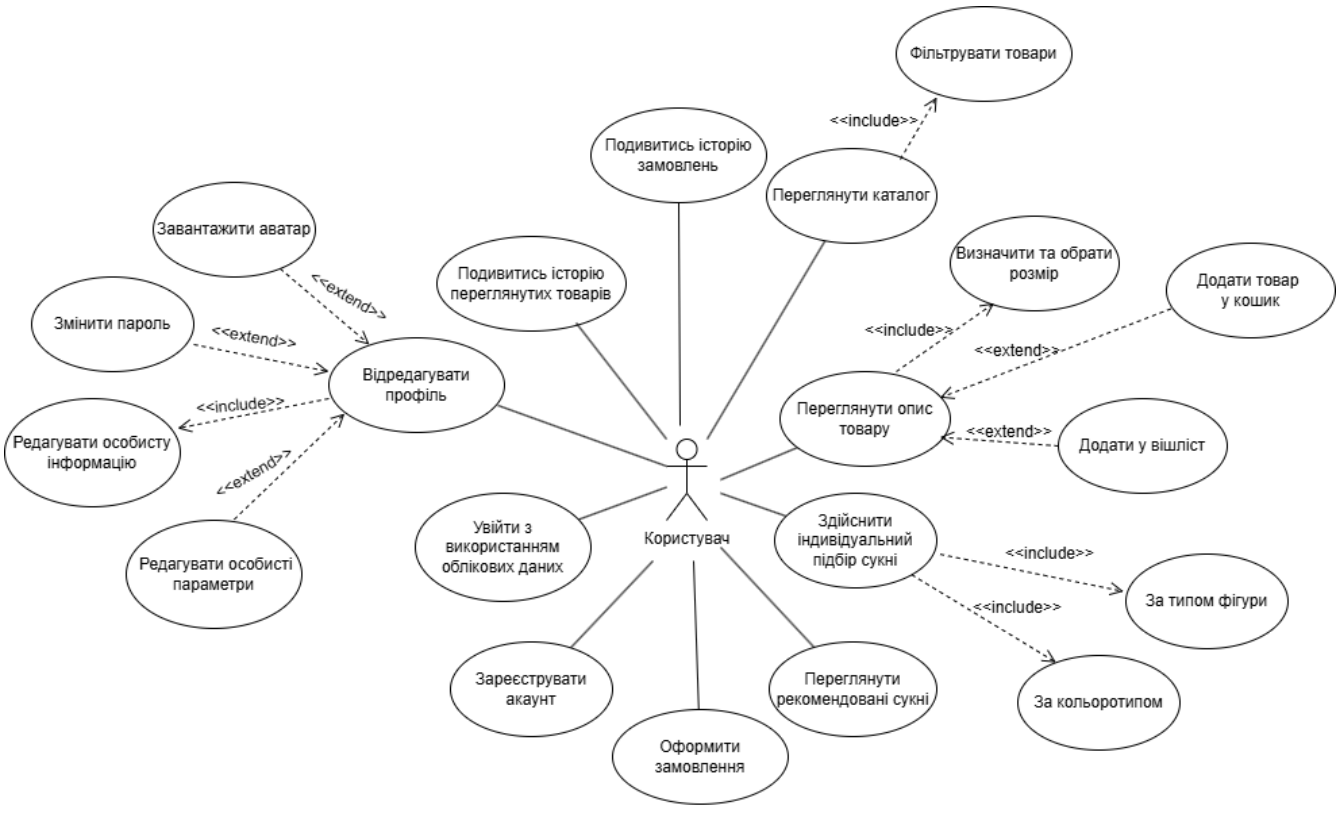


Рис. 3.2 Діаграма варіантів використання актора «Користувач»

3.3 Сценарії взаємодії адміністратора із системою

У цьому підрозділі розглянуто основні сценарії використання системи зі сторони адміністратора. На відповідній діаграмі варіантів використання зображено взаємодію між адміністратором і підсистемами web-застосунку, що відповідають за керування користувачами, товарами, замовленнями, а також особистим профілем адміністратора.

Адміністратор має доступ до функціональності перегляду зареєстрованих користувачів. Він також може керувати товарами, що включає додавання нових товарів до каталогу, їх редагування, видалення, зміну кількості доступних одиниць по розмірах, а також категоризацію товарів відповідно до типу фігури або кольоротипу. Така функціональність дозволяє підтримувати актуальність та якість асортименту, що напряму впливає на точність рекомендацій користувачам.

Окрема гілка діаграми присвячена управлінню замовленнями. Адміністратор може переглядати всі замовлення, змінювати їх статус (наприклад, “у обробці”, “відправлено”, “завершено”) та видаляти при необхідності. Це забезпечує контроль над етапами обробки й логістики замовлень.

Крім того, адміністратор має доступ до адміністрування власного профілю, де реалізовано можливість зміни пароля, завантаження аватару. Це дозволяє підтримувати персоналізацію навіть у службовій ролі.

Дана діаграма дозволяє візуально оцінити повний спектр обов’язків адміністратора та рівень його інтеграції в усі процеси управління системою. Вона є важливою складовою при побудові модулів авторизації, реалізації інтерфейсів адміністративної панелі та визначенні політик доступу.

Діаграма варіантів використання адміністратором наведена на рисунку 3.3.

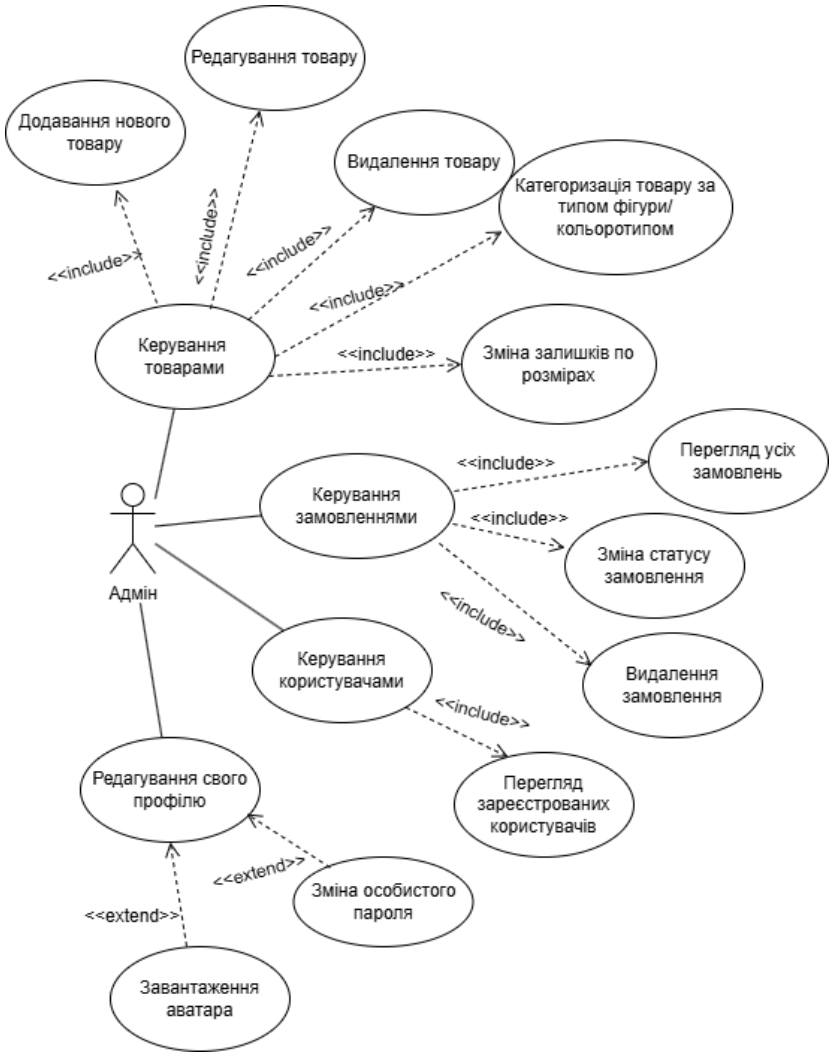


Рис. 3.3 Діаграма варіантів використання актора «Адміністратор»

3.4 Структурна діаграма класів системи персоналізованого підбору

Дана діаграма класів ілюструє основну логіку системи персоналізованого підбору суконь. У центрі структури знаходиться клас UserRegistration, який містить параметри тіла (плечі, талія, стегна) та зовнішності (тон шкіри, колір очей і волосся), а також результати обчислень — тип фігури (BodyType) і кольоротип (ColorType).

Для визначення типу фігури використовується сервіс BodyTypeDetectorService, що реалізує нечітку логіку на основі співвідношень параметрів. Аналогічно, сервіс ColorTypeDetectorService визначає кольоротип на основі класу ColorProfileInput, який містить основні характеристики зовнішності.

Типи фігури (BodyType) та кольоротипи (ColorType) реалізовані як енумерації, що включають відповідні варіанти (наприклад, PEAR, HOURGLASS, WINTER_DEEP, SUMMER_COOL тощо). Окремі енум-класи SkinTone, HairColor і EyeColor містять набір можливих значень з текстовими мітками.

Клас Dress зберігає інформацію про сукні та пов’язаний із множинами Set<BodyType> і Set<ColorType>, що дозволяє встановити відповідність товару до кількох типів фігури і зовнішності.

Представлена структура забезпечує модульність, масштабованість та підтримку персоналізованих рекомендацій у web-застосунку.

Діаграма класів системи персоналізованого підбору суконь наведено на рисунку 3.4.

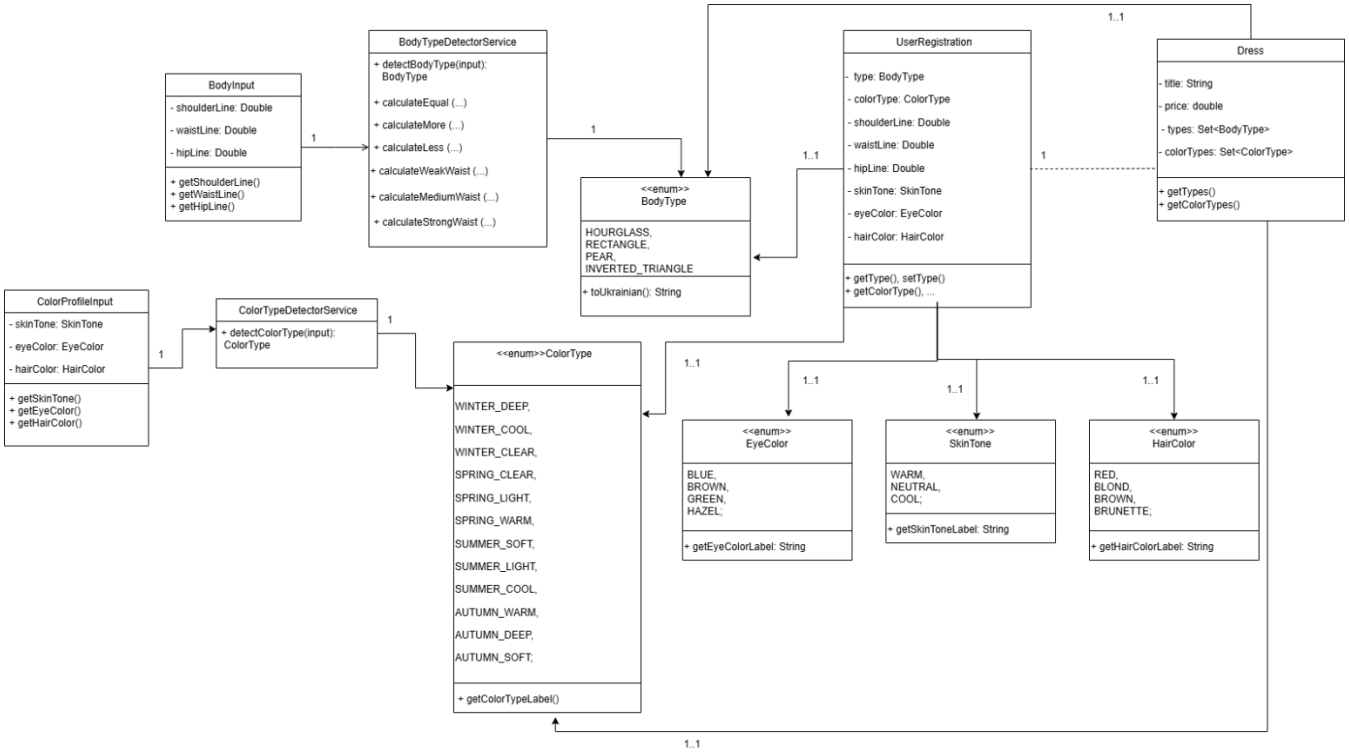


Рис. 3.4 Діаграма класів системи персоналізованого підбору суконь

3.5 Алгоритм класифікації типу фігури на основі нечітких правил

У системі персоналізованого підбору товарів важливе значення має визначення типу фігури користувача. Цей процес реалізовано за допомогою Java-

сервісу BodyTypeDetectorService, який використовує підхід, заснований на нечіткій логіці. На основі введених користувачем параметрів — ширини плечей, талії та стегон — система обчислює ступінь належності до одного з чотирьох основних типів фігури: «пісочний годинник», «прямокутник», «груша» та «перевернутий трикутник».

Вибір саме лінійних показників, а не обхватів, базується на сучасному підході до визначення силуету, де ключову роль відіграє співвідношення горизонтальних параметрів на рівні плечей, талії та стегон. Згідно з результатами дослідження [7], для формування достовірної класифікації тіла достатньо аналізувати пропорції між цими трьома точками, без урахування повного об'єму. Автори підкреслюють, що саме ширина (лінії), а не окружність, точніше відображає візуальне сприйняття типу фігури в контексті fashion-рекомендацій.

Алгоритм використовує методи нечіткої інтерпретації, зокрема функції приналежності calculateEqual, calculateMore, calculateLess, які оцінюють різницю між плечима та стегнами. Додатково розраховується ступінь вираженості талії через функції calculateStrongWaist, calculateMediumWaist і calculateWeakWaist. Ці значення використовуються для формування нечітких правил, які оцінюють кожен можливий тип фігури та обчислюють його "рівень відповідності".

Система формує відповідну карту оцінок (мапу BodyType → Double), де кожному типу фігури відповідає його ймовірнісна оцінка. Для остаточного вибору використовується стратегія max — обирається той тип, який має найвищий показник приналежності. У випадку однакових значень реалізовано пріоритетність: тип "груша" має перевагу, далі — "пісочний годинник", оскільки вони є найбільш специфічними в контексті рекомендацій одягу.

Важливою особливістю цієї реалізації є те, що вона дозволяє не просто жорстко класифікувати користувача за одним типом, а формувати більш гнучке, наближене до реальності рішення, що базується на порогових значеннях і відносних пропорціях. Це робить алгоритм стійким до незначних коливань у введених даних і покращує якість персоналізації в системі.

На рисунку 3.5 наведено фрагменти класу BodyTypeDetectorService, а саме методи оцінки ступеня вираженості талії.

```
private double calculateStrongWaist(double diff) {  1 usage
    return diff >= 10 ? 1.0 : (diff < 8 ? 0.0 : (diff - 8) / 2.0);
}

private double calculateMediumWaist(double diff) {  1 usage
    if (diff >= 6 && diff <= 10) return 1.0;
    if (diff > 10 && diff <= 12) return 1.0 - (diff - 10) / 2.0;
    if (diff >= 4 && diff < 6) return (diff - 4) / 2.0;
    return 0.0;
}

private double calculateWeakWaist(double diff) {  1 usage
    return diff < 6 ? 1.0 - Math.min(diff / 6.0, 1.0) : 0.0;
}
```

Рис. 3.5 Методи оцінки ступеня вираженості талії

Опис функцій:

– calculateStrongWaist(diff) — реалізує функцію належності для сильно вираженої талії. Якщо різниця між плечима/стегнами та талією ≥ 10 см — повертається максимальне значення приналежності 1. Якщо різниця < 8 см — значення 0. У проміжку 8–10 см використовується лінійна інтерполяція.

– calculateMediumWaist(diff) — визначає середній рівень вираженості талії. Якщо різниця 6–10 см — повертається 1. Якщо 10–12 см — значення поступово зменшується до 0. Якщо 4–6 см — значення плавно зростає від 0 до 1. Це дозволяє гнучко обробляти граничні випадки.

– calculateWeakWaist(diff) — описує слабко виражену талію. Якщо різниця < 6 см — значення обернено пропорційне до неї. При перевищенні 6 см функція повертає 0.

Таблиця 3.1 містить нечіткі правила, що використовуються для класифікації типів фігури на основі співвідношення між плечима, талією та стегнами.

Реалізований підхід забезпечує високу точність класифікації користувачів за типом фігури, що є основою для формування релевантних рекомендацій суконь у межах функціоналу системи. Завдяки нечіткій логіці вдалося досягти адаптивності

алгоритму до індивідуальних відмінностей користувачів, що суттєво покращує якість персоналізації [7].

Таблиця 3.1

Таблиця нечітких правил для класифікації фігури

Умова	**Визначений тип фігури**
Плечі приблизно дорівнюють стегнам (різниця не більше 5 см), талія виражена	Пісочний годинник
Плечі приблизно дорівнюють стегнам(різниця не більше 5 см), талія слабо виражена	Прямокутник
Стегна суттєво ширші за плечі (різниця більше 5 см)	Груша
Плечі значно ширші за стегна (різниця більше 5 см)	Перевернутий трикутник

3.6 Визначення кольоротипу на основі зовнішності користувач

У системі персоналізованого підбору товарів важливим етапом є визначення кольоротипу користувача — сезонного типу зовнішності, який ґрунтується на поєднанні тону шкіри, кольору очей і волосся. Це дозволяє підвищити точність рекомендацій, забезпечивши кращу відповідність кольорової гами товарів індивідуальним характеристикам користувача.

Для реалізації цього функціоналу в межах web-застосунку створено сервіс ColorTypeDetectorService, який аналізує параметри, введені користувачем у формі — тон шкіри (SkinTone), колір очей (EyeColor) та волосся (HairColor). На основі цих даних застосовується набір логічних умов, які дозволяють класифікувати зовнішність користувача до одного з 12 підтипів класичної сезонної системи: WINTER_DEEP, WINTER_CLEAR, WINTER_COOL, SPRING_WARM,

SPRING_LIGHT, SPRING_CLEAR, SUMMER_LIGHT, SUMMER_SOFT, SUMMER_COOL, AUTUMN_WARM, AUTUMN_SOFT, AUTUMN_DEEP.

Рішення засноване на принципах класичної теорії персонального кольоротипу, адаптованої для автоматизованої обробки. Сервіс враховує часткові збіги (наприклад, NEUTRAL як перехідне значення тону шкіри) і пріоритети відповідності, що забезпечує гнучкість і точність результатів.

Схожий підхід описано у дослідженні [24], де запропоновано систему персонального кольорового аналізу на основі параметрів зовнішності користувача. Автори підкреслюють, що такі системи можуть бути ефективно інтегровані у fashion-рекомендаційні платформи та покращують якість персоналізованих порад.

Такий підхід дозволяє враховувати індивідуальні особливості зовнішності кожного користувача, що покращує естетичну відповідність товарів, підвищує задоволеність клієнтів та зменшує ймовірність повернень.

Зображення логіки визначення кольоротипу наведено на рисунку 3.6.

```
public ColorType detectColorType(ColorProfileInput input) {  7 usages
    SkinTone tone = input.getSkinTone();
    EyeColor eyes = input.getEyeColor();
    HairColor hair = input.getHairColor();

    if ((tone == SkinTone.COOL || tone == SkinTone.NEUTRAL) &&
            (hair == HairColor.BRUNETTE || hair == HairColor.BROWN)) {
        if (eyes == EyeColor.BROWN || eyes == EyeColor.HAZEL) return ColorType.WINTER_DEEP;
        if (eyes == EyeColor.GREEN) return ColorType.WINTER_CLEAR;
        if (eyes == EyeColor.BLUE) return ColorType.WINTER_COOL;
    }

    if ((tone == SkinTone.WARM || tone == SkinTone.NEUTRAL) &&
            (hair == HairColor.BLOND || hair == HairColor.RED)) {
        if (eyes == EyeColor.BROWN || eyes == EyeColor.HAZEL) return ColorType.SPRING_WARM;
        if (eyes == EyeColor.GREEN) return ColorType.SPRING_LIGHT;
        if (eyes == EyeColor.BLUE) return ColorType.SPRING_CLEAR;
    }

    if ((tone == SkinTone.COOL || tone == SkinTone.NEUTRAL) &&
            (hair == HairColor.BLOND || hair == HairColor.RED)) {
        if (eyes == EyeColor.GREEN) return ColorType.SUMMER_LIGHT;
        if (eyes == EyeColor.HAZEL || eyes == EyeColor.BROWN) return ColorType.SUMMER_SOFT;
        if (eyes == EyeColor.BLUE) return ColorType.SUMMER_COOL;
    }

    if ((tone == SkinTone.WARM || tone == SkinTone.NEUTRAL) &&
            (hair == HairColor.BROWN || hair == HairColor.RED || hair == HairColor.BRUNETTE)) {
        if (eyes == EyeColor.HAZEL || eyes == EyeColor.BROWN) return ColorType.AUTUMN_WARM;
        if (eyes == EyeColor.GREEN) return ColorType.AUTUMN_SOFT;
        if (eyes == EyeColor.BLUE) return ColorType.AUTUMN_DEEP;
    }
    return null;
}
```

Рис. 3.6 Логіка визначення кольоротипу

3.7 Структура бази даних

Структура бази даних web-застосунку побудована на реляційній моделі та охоплює основні сутності, пов’язані з користувачами, товарами, замовленнями та підбором за індивідуальними параметрами.

Центральною таблицею є user_registration, яка зберігає дані про користувача, включаючи параметри тіла (плечі, талія, стегна), ознаки зовнішності (тон шкіри, колір очей і волосся), роль, логін та email. Таблиця dress містить інформацію про сукні: назву, опис, категорію, зображення, ціну, колір тощо.

Типи фігури та кольоротипи, що підходять для конкретних товарів, зберігаються в таблицях dress_body_types і dress_color_types. Підтримка кількох зображень реалізована через dress_image, а залишки товарів по розмірах — через dress_stock.

Функціонал кошика реалізовано через таблиці user_cart, user_cart_item, а замовлень — через orders і order_item. Для історії переглядів і пошуків використовуються search_history і view_history. Таблиці user_wishlist та user_wishlist_item відповідають за збереження вподобаних товарів.

Уся структура підтримує зв’язки між сутностями та адаптована для роботи з ORM (Hibernate), що забезпечує ефективне управління даними та масштабованість системи.

Структура бази даних наведена на рисунку 3.7.

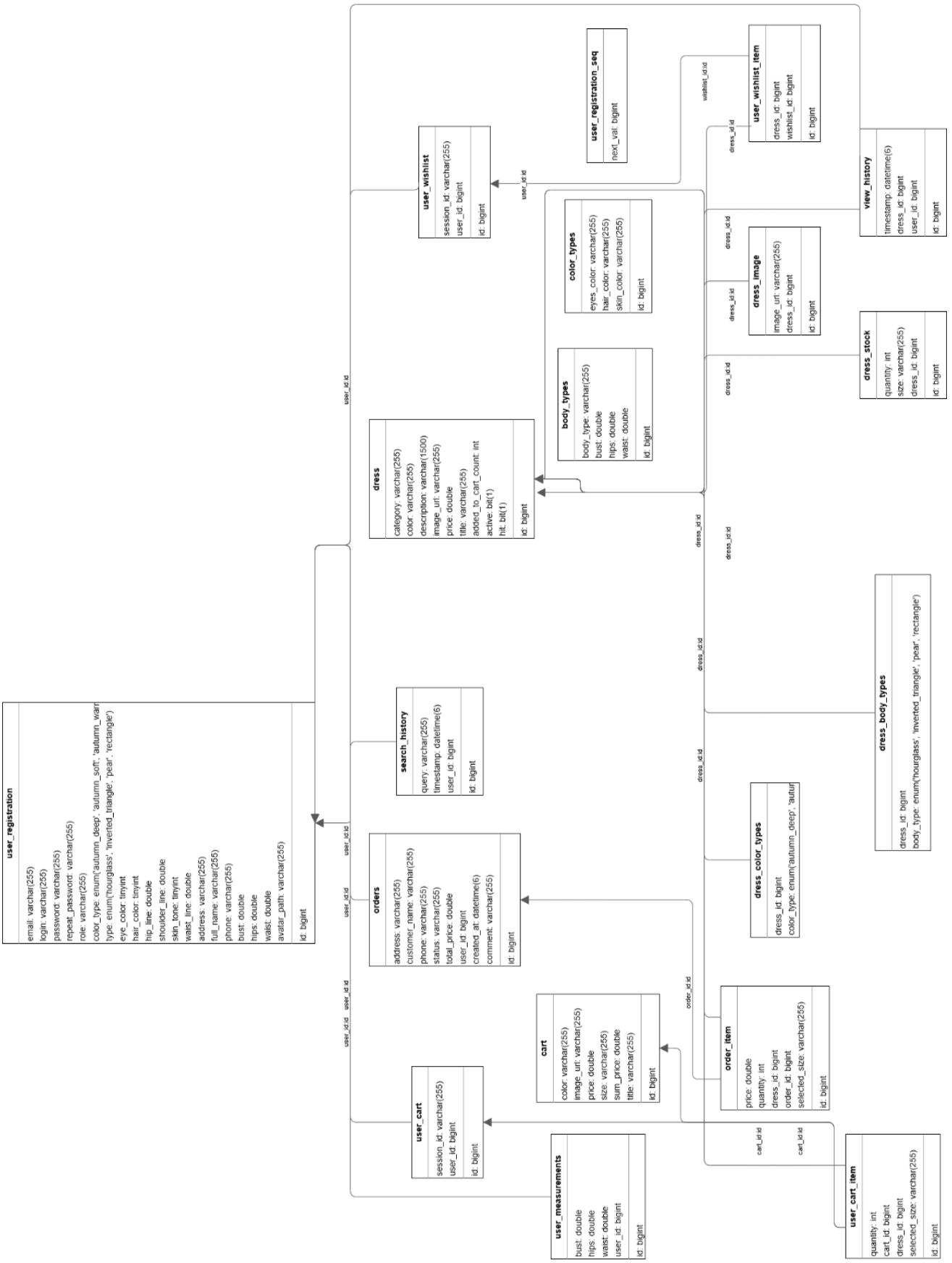


Рис. 3.7 Структура бази даних

4 РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ ТА ТЕСТУВАННЯ СИСТЕМИ

4.1 Структура навігації web-застосунку

У процесі розробки web-застосунку для автоматизованого підбору вечірніх суконь важливим етапом є побудова логічної та зручної структури навігації. Користувач має швидко знаходити потрібні товари, легко переходити до персоналізованих рекомендацій, керувати замовленнями та редагувати власні дані. З іншого боку, адміністратор повинен мати змогу оперативно працювати з каталогом товарів і замовленнями.

Для ілюстрації побудови маршрутизації в системі створено мапу сайту, що охоплює ключові функціональні області: головну сторінку користувача, особистий кабінет і панель адміністратора. На основі цих схем визначено всі можливі переходи між сторінками та сформовано логіку взаємодії з основними модулями застосунку. Кожен елемент навігації має чітке функціональне призначення, що забезпечує структуровану, послідовну й доступну роботу з інтерфейсом.

На першій діаграмі відображено навігацію на головній сторінці. Центральною точкою є «Головна сторінка», з якої користувач може перейти до каталогу товарів із фільтрацією, до індивідуального підбору суконь, або до інформаційного блоку «Про нас». Індивідуальний підбір, своєю чергою, включає два етапи: визначення типу фігури та визначення кольоротипу. Кожен товар у каталозі або рекомендаціях має сторінку деталей товару з вибором розміру, після чого можна додати товар до кошика або списку бажань. У фіналі — оформлення замовлення.

Діаграма навігації головної сторінки зображено на рисунку 4.1.

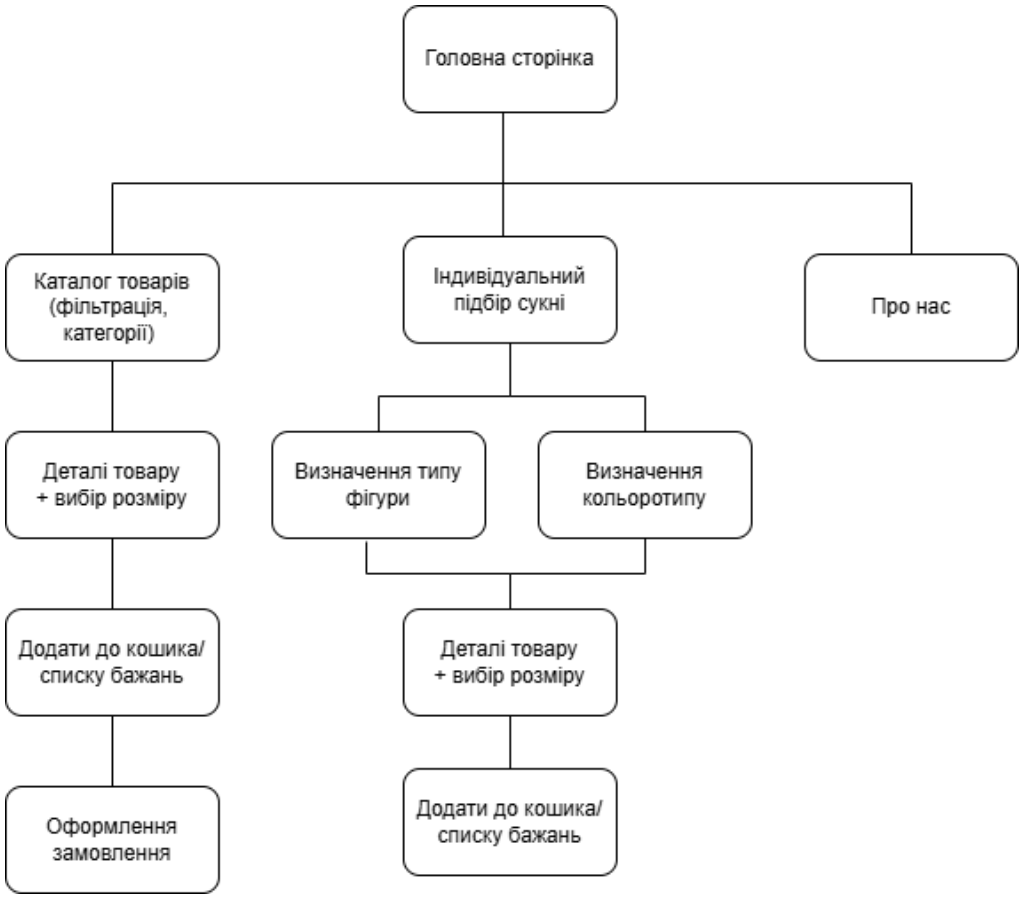


Рис. 4.1 Діаграма навігації головної сторінки

Друга діаграма відображає структуру особистого кабінету користувача. Тут доступні переходи до кошика, списку бажань, переглянутих товарів, історії замовлень, а також до розділу редагування профілю, де користувач може змінювати персональні дані, пароль або аватар.

Діаграма навігації особистого кабінету користувача зображена на рисунку 4.2.

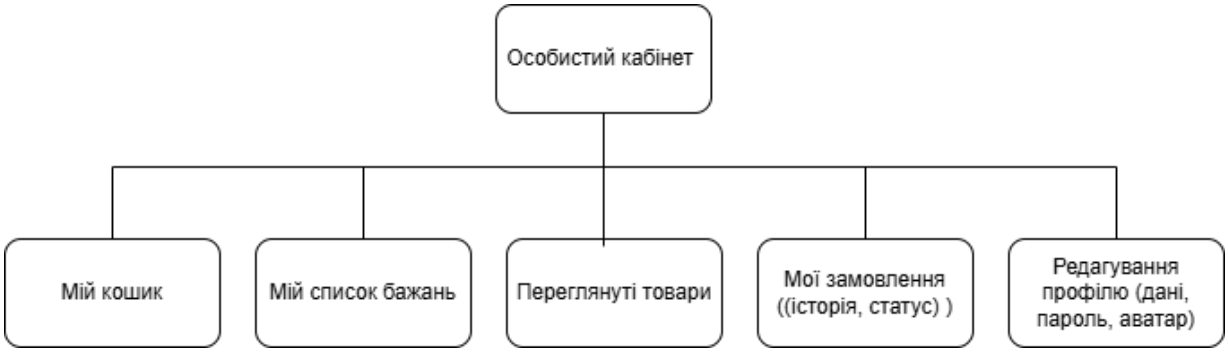


Рис. 4.2 Діаграма навігації особистого кабінету користувача

Третя діаграма показує панель адміністратора, яка має дві головні гілки — керування товарами (додавання, редагування, видалення, категоризація) та керування замовленнями (зміна статусу, видалення замовлень). Така структура дозволяє адміністратору ефективно працювати з даними магазину та контролювати активність.

Діаграма навігації панелі адміністратора зображена на рисунку 4.3.

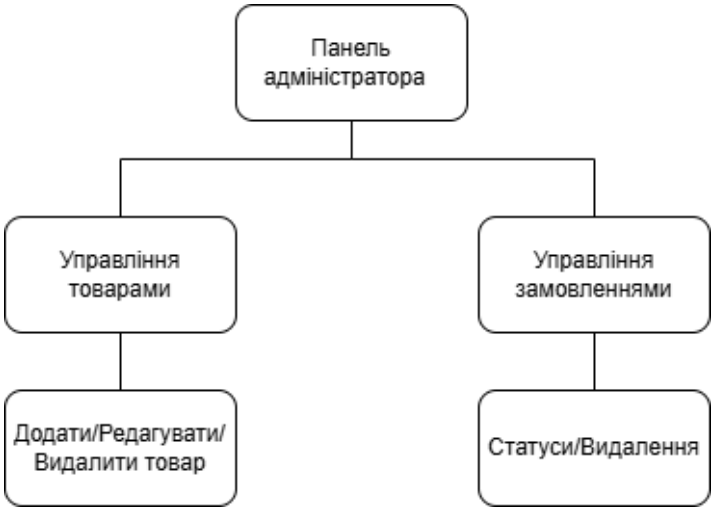


Рис. 4.3 Діаграма навігації панелі адміністратора

Структура навігації реалізована у відповідності до принципів юзабіліті: інтерфейс логічно розділений за ролями, користувачі легко орієнтуються між сторінками, а адміністратор має доступ до необхідних функцій з одного централізованого інтерфейсу.

4.2 Реалізація інтерфейсу системи

Ефективний користувацький інтерфейс (UI) є критично важливим елементом будь-якого web-застосунку, особливо в e-commerce системах, де зручність взаємодії напряму впливає на поведінку клієнта та загальний рівень задоволеності сервісом. Важливо, щоб інтерфейс був естетично привабливим і функціонально повним. Як показують сучасні дослідження, зручність навігації, швидкий доступ

до ключових функцій та персоналізація взаємодії суттєво підвищують конверсію користувачів у покупців [19].

Під час розробки інтерфейсу для системи автоматизованого підбору вечірніх суконь було використано адаптивну верстку з підтримкою різних розширень екранів. Головними критеріями стали: простота, логічна структура, візуальна привабливість і чіткий поділ між користувацькою та адміністративною зонами. Для побудови HTML-розмітки застосовано Thymeleaf, а для стилізації — Bootstrap 5, що дозволяє створювати сучасні та зручні інтерфейси з урахуванням UI/UX практик [20].

На наступних рисунках (4.4-4.17) наведено ключові інтерфейсні компоненти, реалізовані у проєкті.

На рисунку 4.4 зображено головну сторінку web-застосунку, яка виступає центральним вхідним пунктом для користувача. Інтерфейс забезпечує швидкий доступ до каталогу товарів, персоналізованого підбору суконь за типом фігури та кольоротипом, а також містить інформаційний блок із рекомендаціями та новинами магазину.

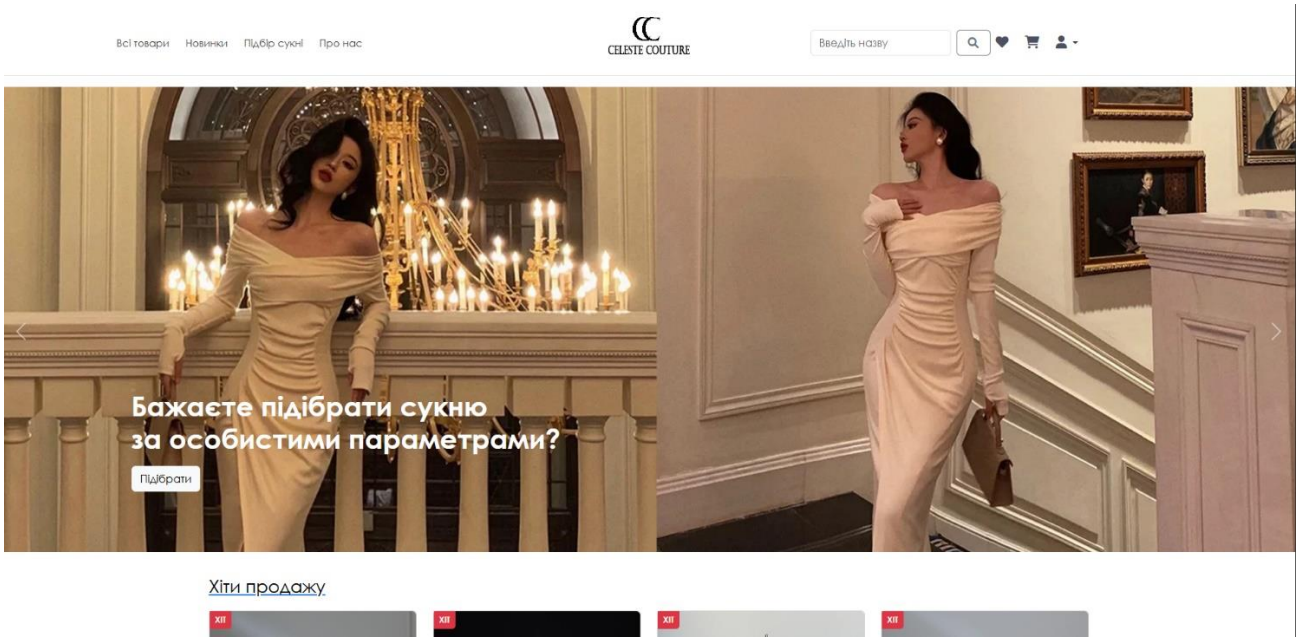


Рис. 4.4 Головна сторінка web-застосунку

На рисунку 4.5 представлено форму «Увійти», яка слугує сторінкою авторизації користувача. Вона реалізує перевірку введених даних на коректність, зокрема правильність електронної пошти та відповідність пароля, а у випадку помилки — інформує про це користувача.

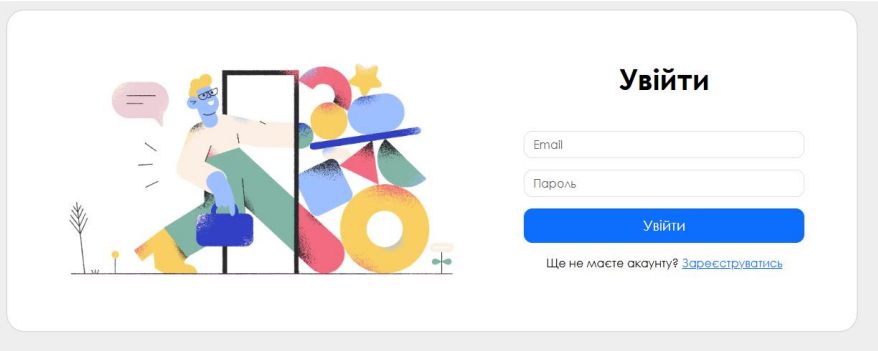


Рис. 4.5 Форма «Увійти»

Рисунок 4.6 ілюструє форму «Зареєструватись», яка дозволяє створити новий обліковий запис. Для завершення реєстрації обов’язково потрібно вказати електронну пошту, пароль, його підтвердження.

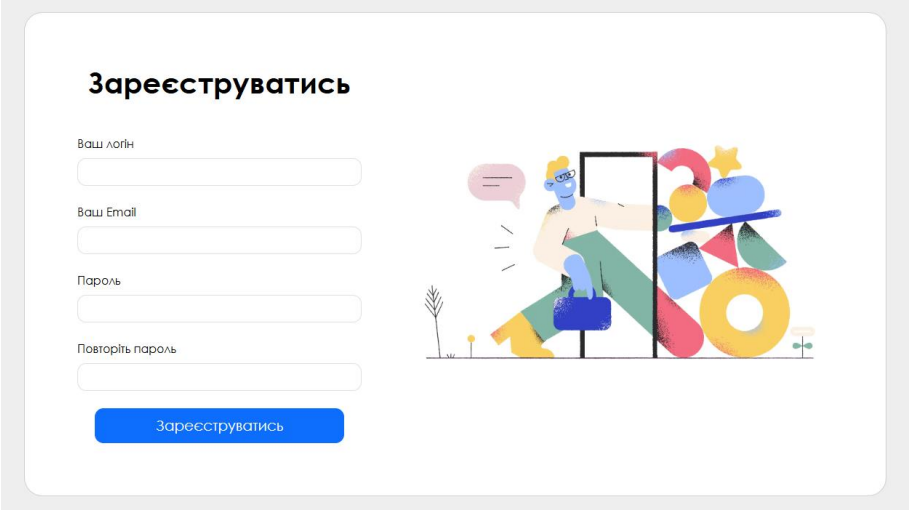


Рис. 4.6 Форма «Зареєструватись»

Рисунок 4.7 демонструє каталог усіх товарів, де реалізовано зручну фільтрацію за ключовими параметрами: категорією, кольором, розміром і ціновим діапазоном. Такий підхід дозволяє користувачеві швидко знайти потрібну сукню серед широкого асортименту.

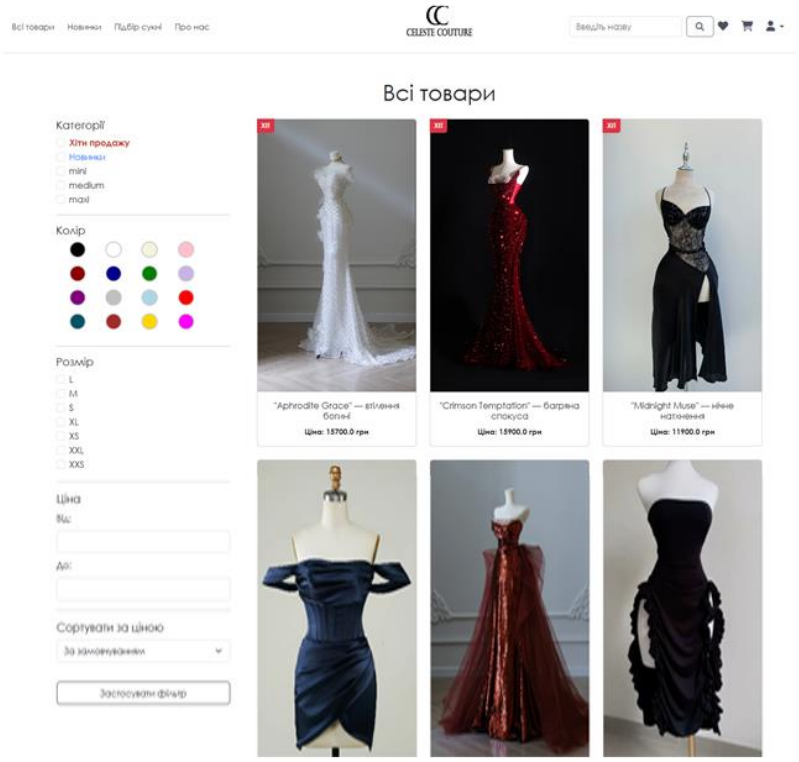


Рис. 4.7 Каталог усіх товарів з фільтром

На рисунку 4.8 зображено сторінку деталей товару, що містить розширену інформацію про вибрану модель: декілька фотографій, опис, ціну та список доступних розмірів. Кнопка «додати в кошик» залишається неактивною, доки користувач не вибере конкретний розмір, що запобігає помилковому додаванню товару.

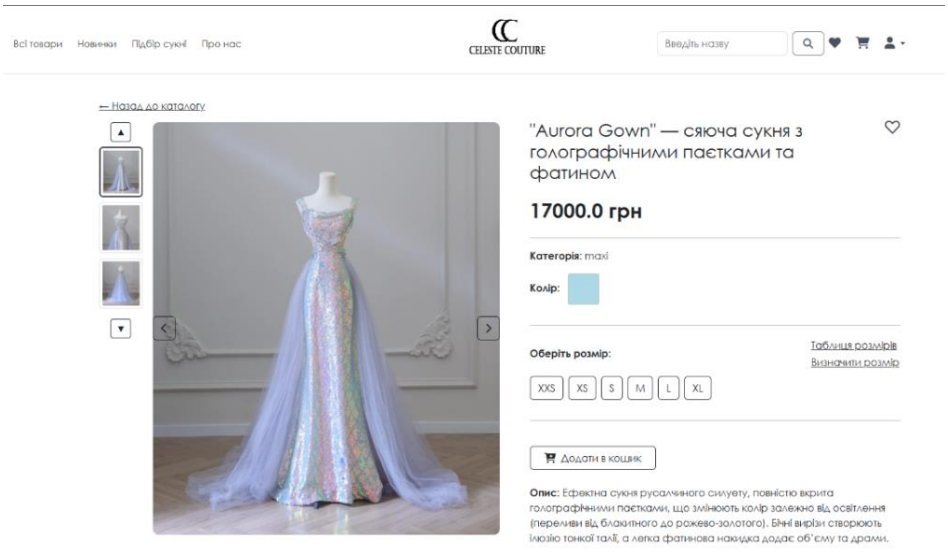


Рис. 4.8 Сторінка деталей товару

Рисунки 4.9 і 4.10 демонструють роботу модальних вікон для точнішого вибору розміру. Вікно «Таблиця розмірів» (рис. 4.9) містить сітку параметрів для кожного розміру, а «Визначити розмір» (рис. 4.10) дозволяє ввести власні дані (ог, от, ос) і отримати рекомендований розмір, який підсвічується червоною рамкою.

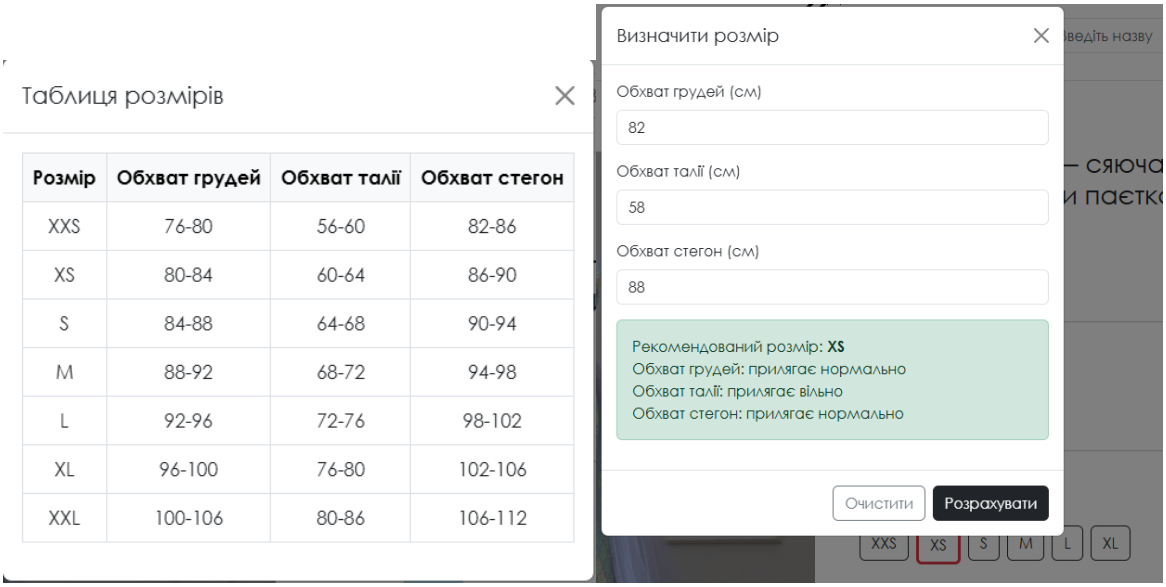


Рис. 4.9 - 4.10 Модальне вікно «Таблиця розмірів» та Модальне вікно «Визначити розмір»

Рисунок 4.11 демонструє модальне вікно початку персонального підбору, яке відкривається з головної сторінки. Користувачеві пропонується зробити вибір між двома напрямами — визначенням типу фігури або кольоротипу.

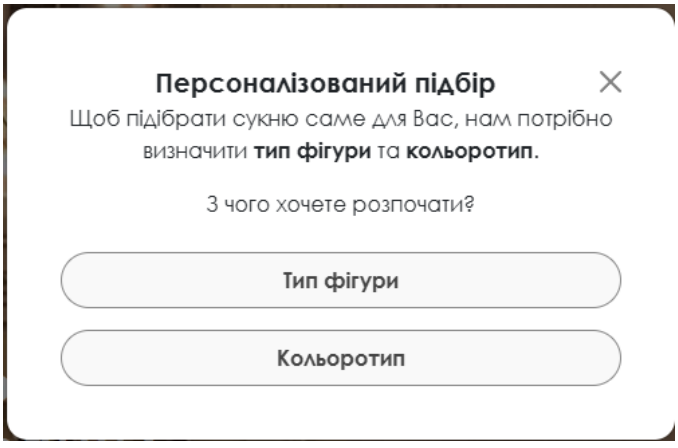


Рис. 4.11 Модальне вікно початку персонального підбору

Рисунок 4.12 ілюструє сторінку з описом основних типів фігури, а також порадами щодо правильного самостійного вимірювання ширини плечей, талії та стегон. Нижче розташована форма введення параметрів, яка дозволяє користувачеві здійснити розрахунок типу фігури для подальшого персоналізованого підбору товарів.

Рис. 4.12 Сторінка з інформацією про типи фігури та форма введення параметрів

Рисунок 4.13 відображає сторінку з описом кольоротипів та підказками щодо визначення тону шкіри за кольором вен. Нижче розміщено форму, де користувач обирає тон шкіри, колір очей і волосся для визначення кольоротипу.

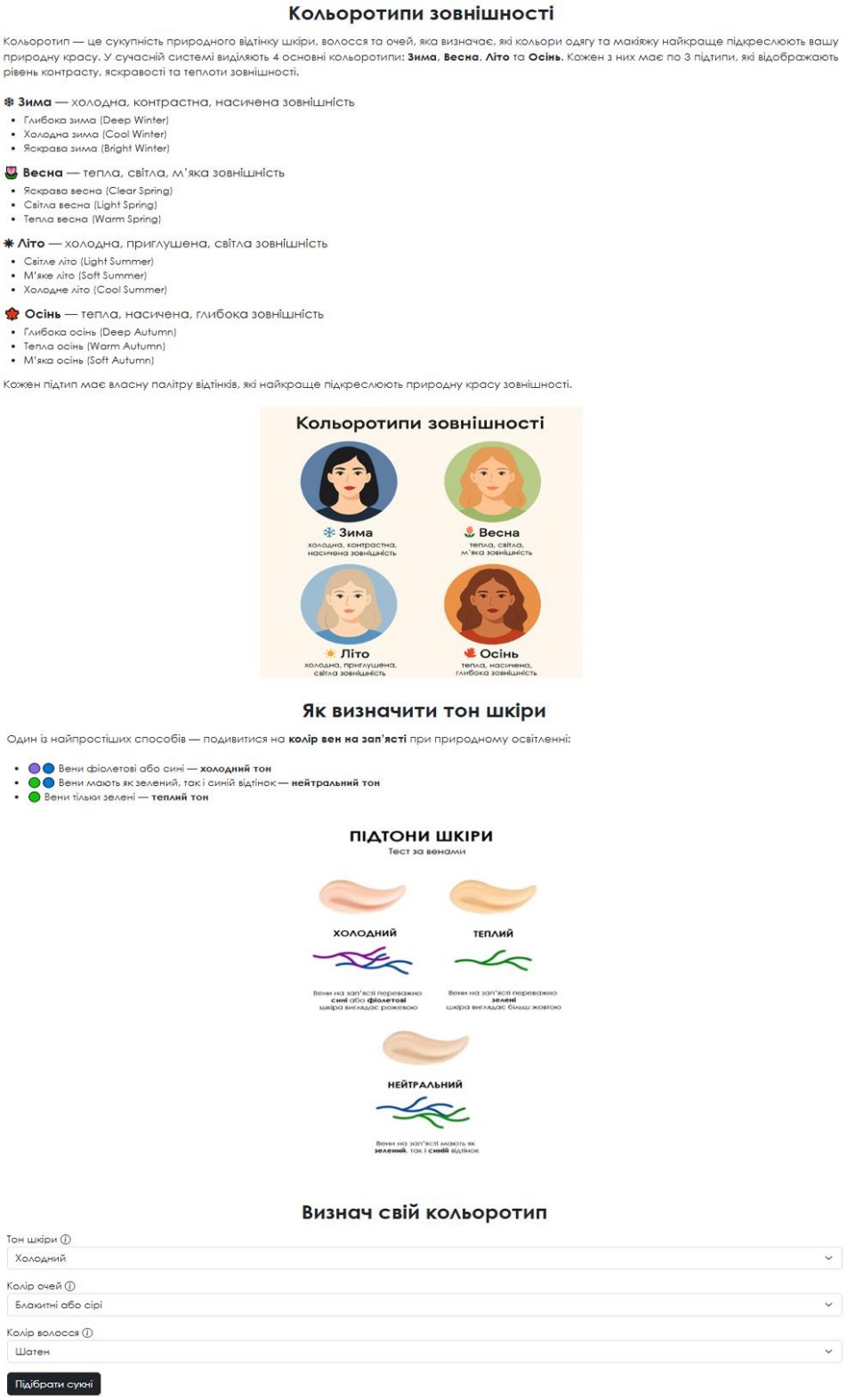


Рис. 4.13 Сторінка з інформацією про кольоротипи та форма введення

Рисунок 4.14 демонструє сторінку з результатами індивідуального підбору — відображається список рекомендованих суконь, сформований відповідно до типу фігури та кольоротипу користувача.

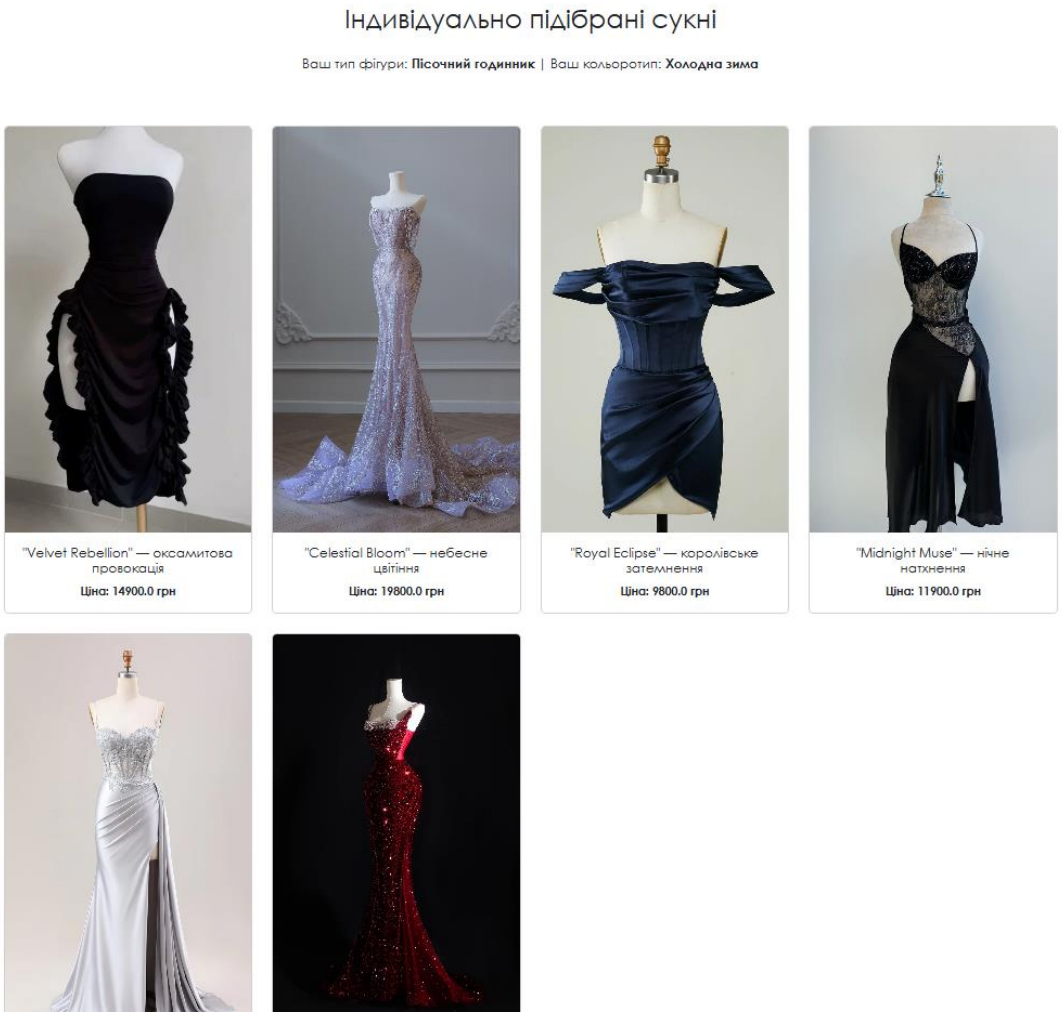


Рис. 4.14 Результати індивідуального підбору

Рисунок 4.15 ілюструє особистий кабінет користувача, який надає доступ до кошика, історії замовлень, переглянутих товарів і можливості редагування персональних даних.

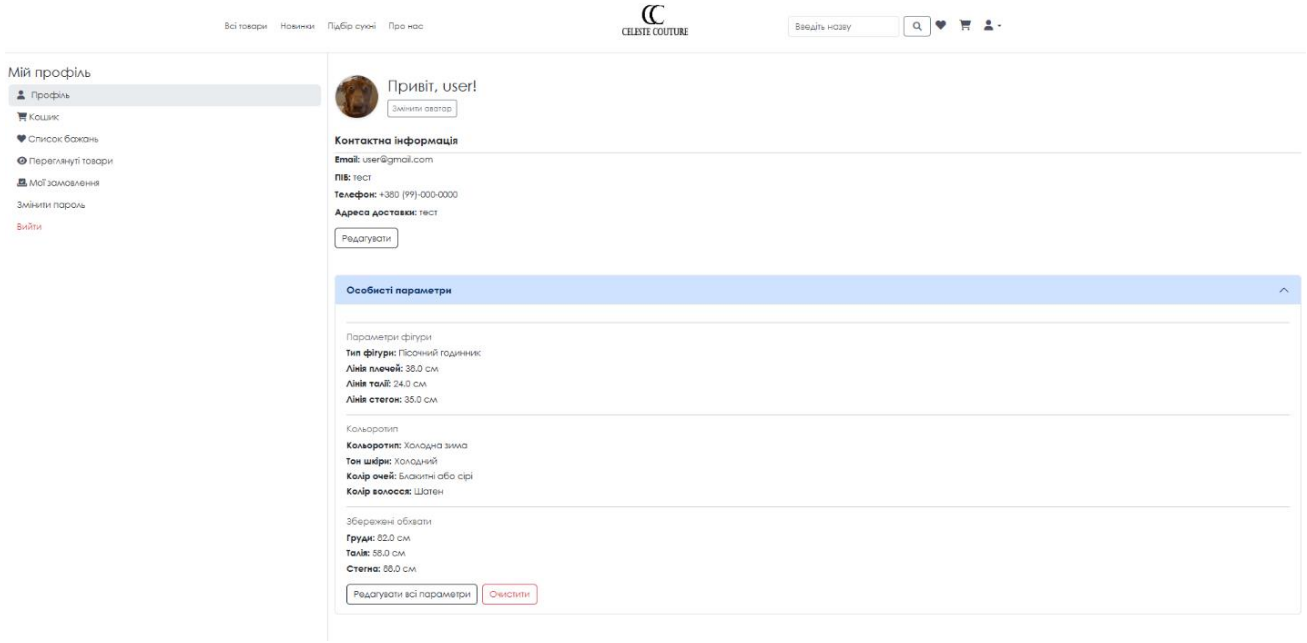


Рис. 4.15 Особистий кабінет користувача

Рисунок 4.16 демонструє панель адміністратора — спеціальний інтерфейс для користувачів з правами «ADMIN», який надає можливість керувати товарами: додавати нові, редагувати наявні та видаляти їх.

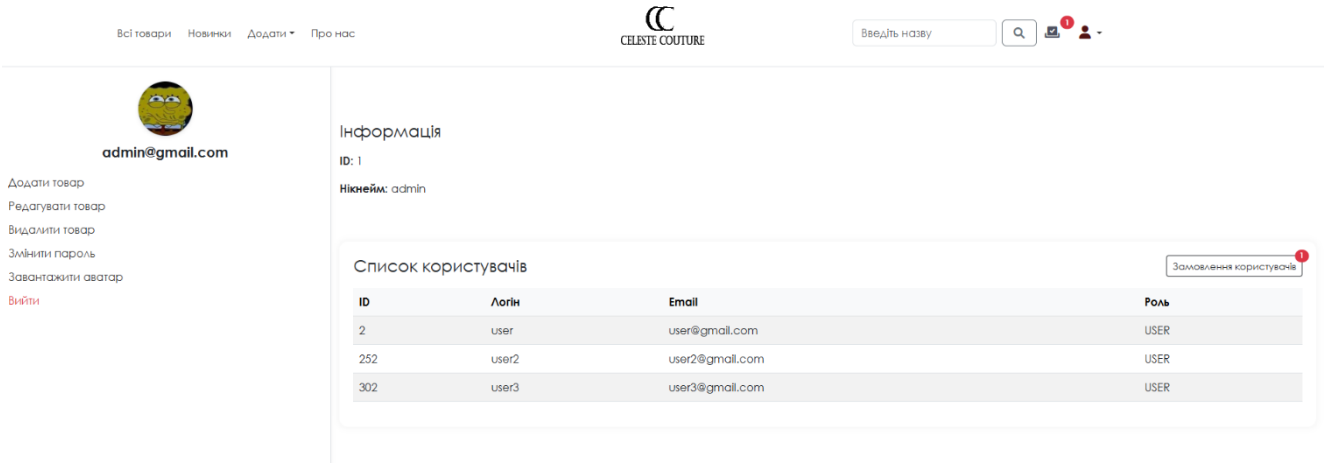


Рис. 4.16 Панель адміністратора

Рисунок 4.17 ілюструє сторінку всіх замовлень адміністратора — інтерфейс для перегляду та керування усіма замовленнями. Адміністратор може змінювати статус замовлень, видаляти їх, а також сортувати за статусом, причому замовлення зі статусом «Очікує» автоматично виводяться першими.

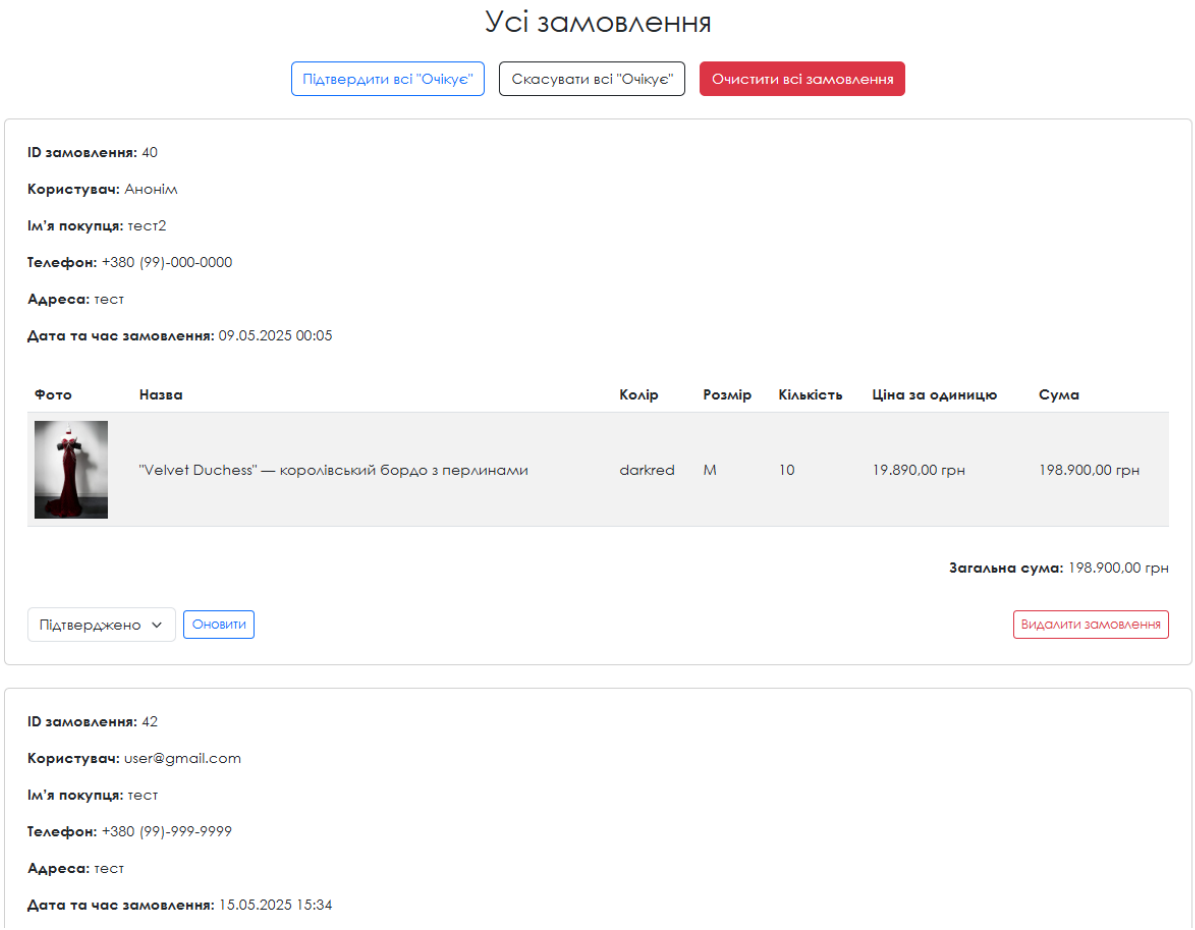


Рис. 4.17 Сторінка всіх замовлень (для адміністратора)

4.3 Кросбраузерне тестування

Кросбраузерне тестування є критичним етапом забезпечення якості web-застосунку, особливо в сфері електронної комерції, де користувачі використовують різноманітні браузери та пристрої. Цей процес гарантує, що web-застосунок функціонує коректно та відображається однаково у різних браузерах, забезпечуючи стабільний користувацький досвід [21].

Під час розробки системи автоматизованого підбору вечірніх суконь було проведено кросбраузерне тестування для перевірки стабільності інтерфейсу, коректності стилів, адаптивної верстки та взаємодії з функціональними модулями. Тестування здійснювалося вручну за допомогою інструментів Chrome DevTools, Firefox Developer Tools та вбудованих засобів інших браузерів [22].

Тестування охоплювало такі браузери:

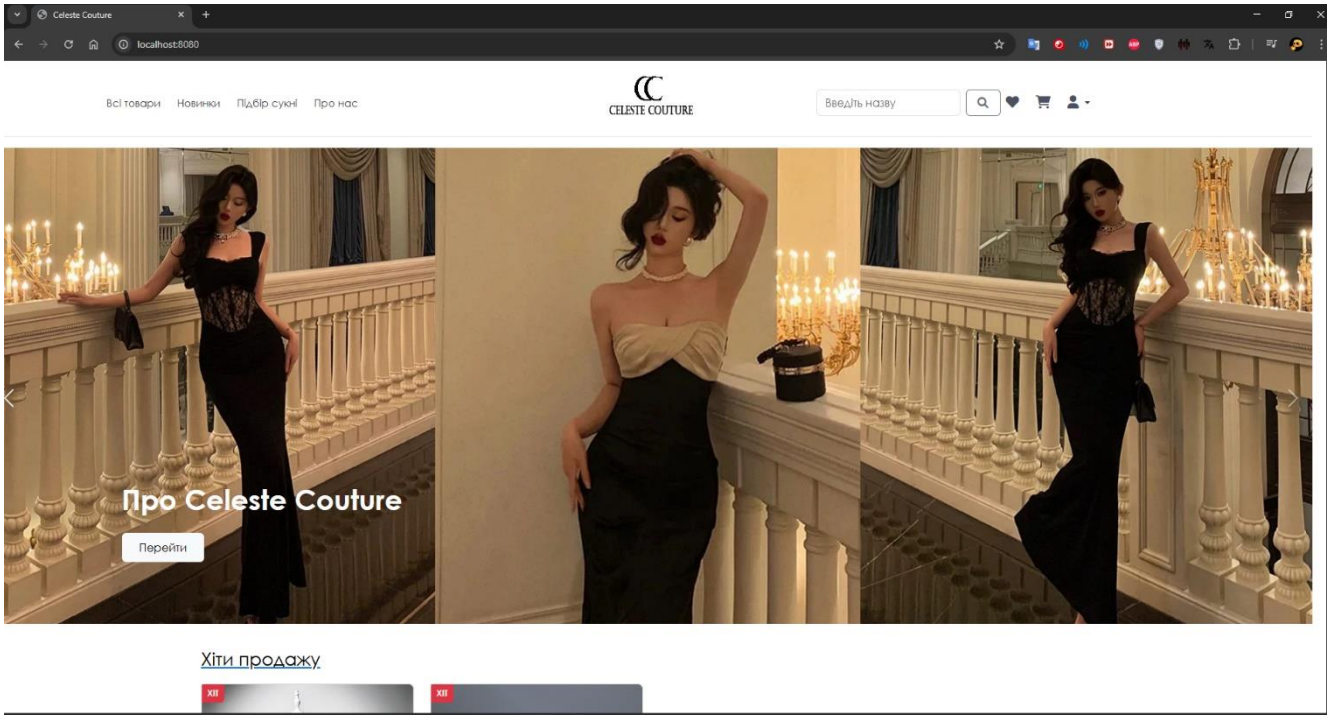


Рис. 4.18 Google Chrome

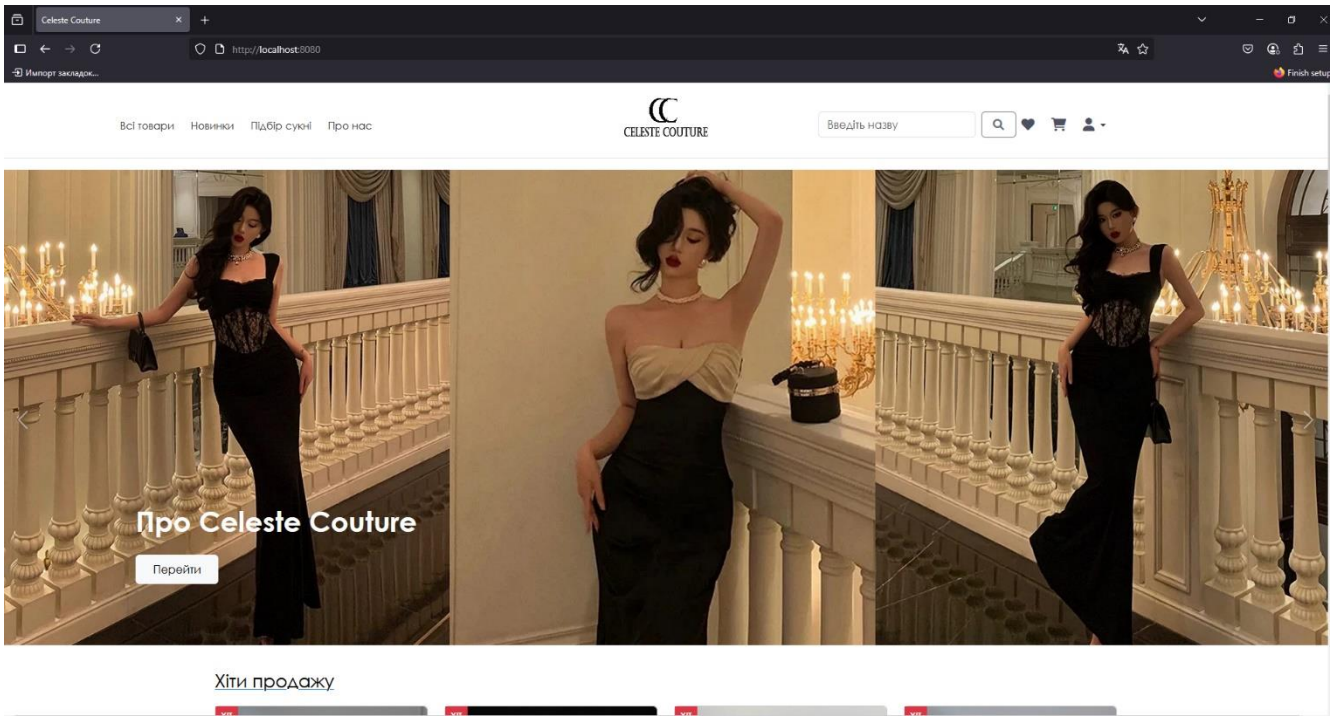


Рис. 4.19 Mozilla Firefox

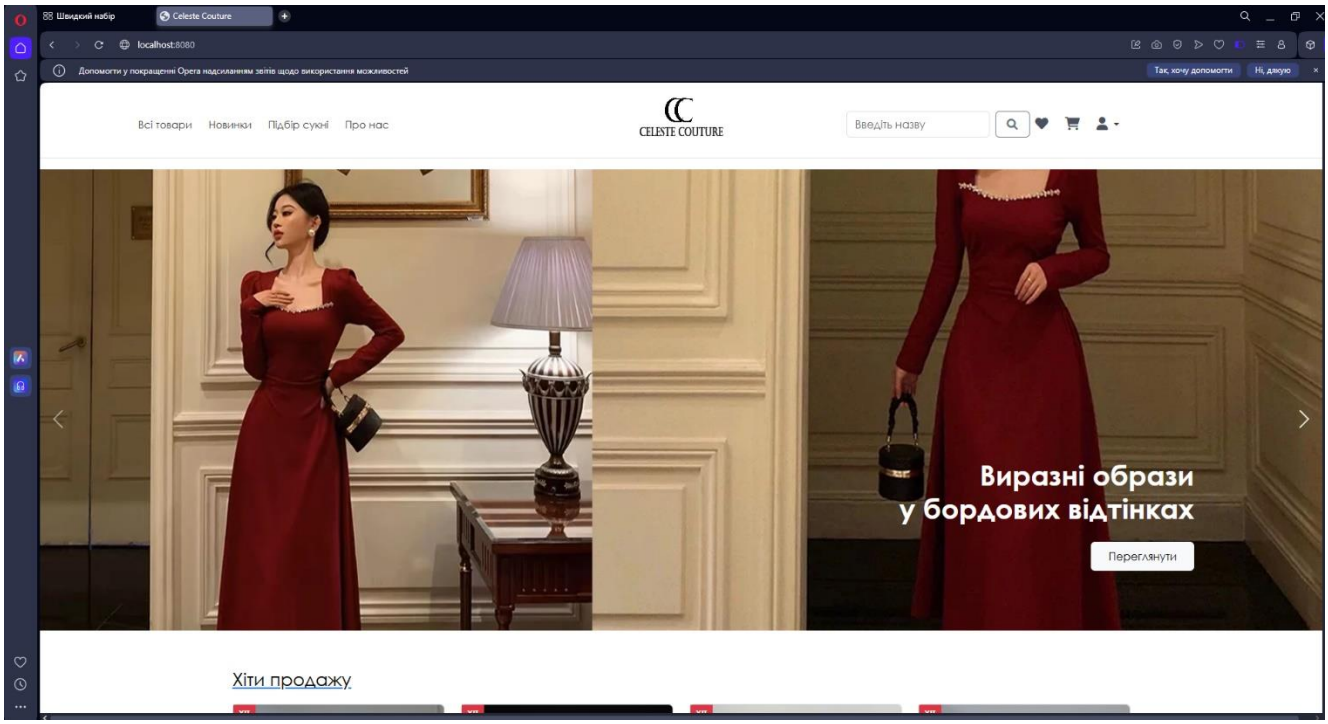


Рис. 4.20 Opera

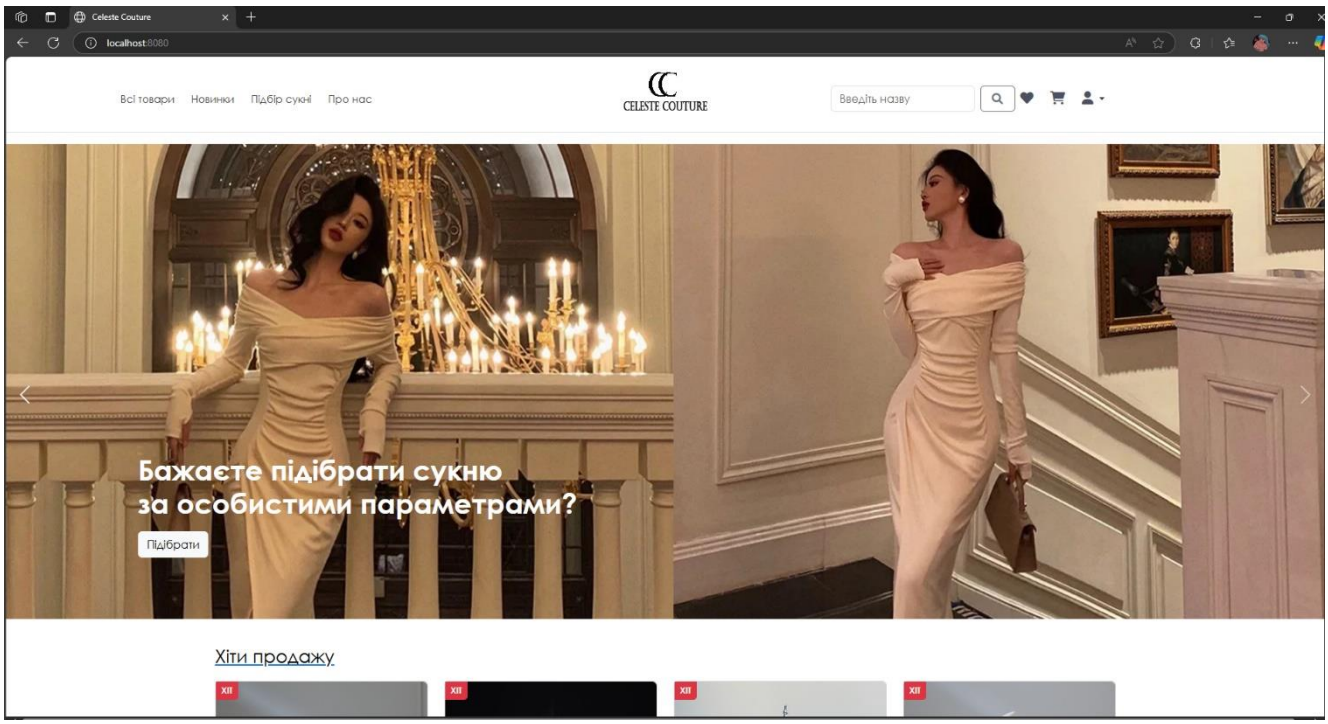


Рис. 4.21 Microsoft Edge

У ході перевірки було протестовано такі ключові елементи:

- головну сторінку;
- каталог товарів з фільтрами;
- вікна визначення типу фігури та кольоротипу;

- процес додавання товару у кошик та вішліст;
- оформлення замовлення;
- особистий кабінет користувача та адміністратора.

За підсумками тестування встановлено, що web-застосунок відповідає стандартам кросбраузерної сумісності, зберігаючи однакову функціональність і дизайн на всіх перевірених платформах. Це підвищує довіру до ресурсу та зменшує ризик втрати клієнтів через технічні обмеження [23].

4.4 Тестування функціональної логіки за допомогою JUnit

Для забезпечення надійності ключових елементів бізнес-логіки web-застосунку було проведено модульне тестування (unit testing) за допомогою фреймворку JUnit 5. Основною метою стало підтвердження коректної роботи найважливіших сервісів і контролерів, зокрема модулів реєстрації користувача, визначення типу фігури та кольоротипу.

4.4.1 Тестування RegisterController

У класі RegisterControllerTest перевірено поведінку методу реєстрації за декількох сценаріїв:

- успішна реєстрація нового користувача (з перевіркою збереження в базі);
- обробка ситуації, коли email уже зайнятий;
- випадок занадто короткого пароля;
- помилка, коли введені паролі не співпадають.

Ці тести перевіряють логіку валідації введених даних, взаємодію з репозиторієм та відповідну реакцію інтерфейсу.

Результати тестування наведені на рисунку 4.22.

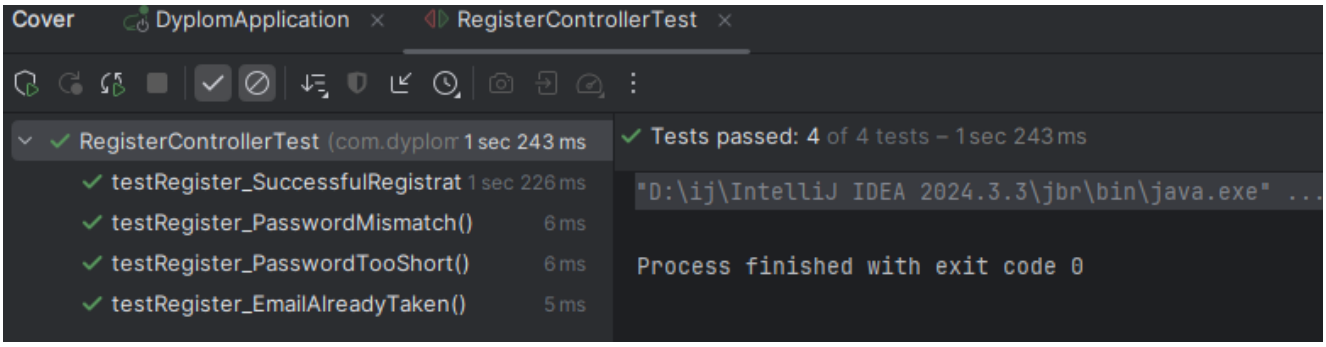


Рис. 4.22 Успішне тестування RegisterControllerTest

4.4.2 Тестування алгоритму визначення типу фігури

У класі BodyTypeDetectorServiceTest реалізовано тестування методу detectBodyType(), що класифікує фігуру користувача як одну з чотирьох категорій:

- «пісочний годинник» (hourglass);
- «груша» (pear);
- «прямокутник» (rectangle);
- «перевернутий трикутник» (inverted triangle).

Кожен тестовий кейс перевіряє конкретне співвідношення параметрів плечей, талії та стегон, що дозволяє впевнено стверджувати про стабільність алгоритму при роботі з різними вхідними даними.

Результати тестування зображено на рисунку 4.23.

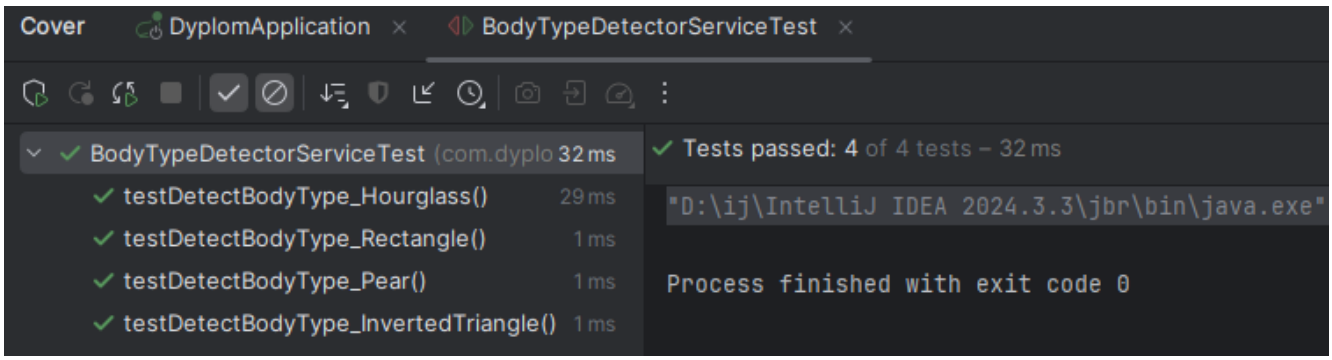


Рис. 4.23 Успішне тестування BodyTypeDetectorServiceTest

4.4.3 Тестування визначення кольоротипу

Сервіс ColorTypeDetectorService також піддавався тестуванню через клас ColorTypeDetectorServiceTest. Було перевірено відповідність повернутого результату введеним параметрам тону шкіри, кольору очей та волосся. Окремі кейси покривають усі основні сезонні кольоротипи: WINTER_DEEP, SPRING_CLEAR, SUMMER_SOFT, AUTUMN_WARM.

Ці тести підтверджують правильність логіки відповідності користувача певному кольоротипу за комбінацією його зовнішніх характеристик.

Результати тестування наведено на рисунку 4.24.

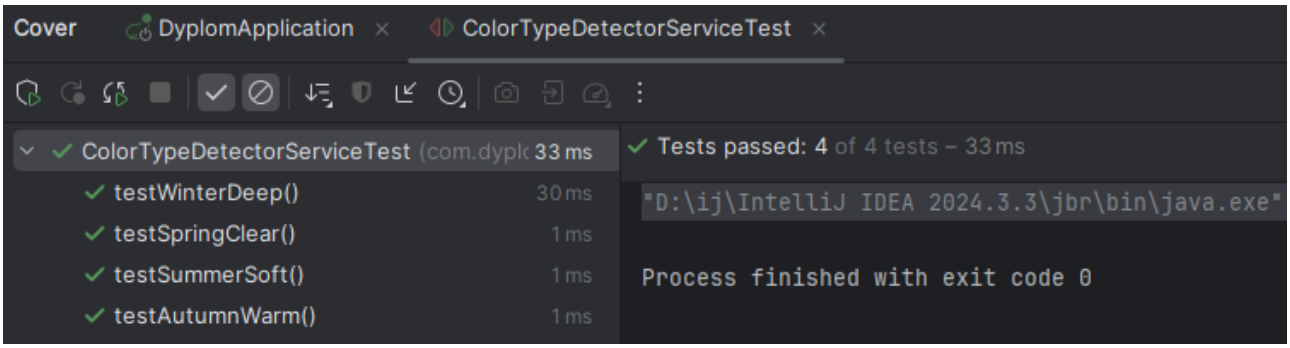


Рис. 4.24 Успішне тестування ColorTypeDetectorServiceTest

ВИСНОВКИ

У результаті виконаної дипломної роботи було створено web-застосунок для автоматизованого підбору вечірніх суконь на основі індивідуальних параметрів фігури та кольоротипу користувача. Це дозволяє підвищити персоналізацію процесу онлайн-шопінгу, зменшити кількість повернень і підвищити задоволеність клієнтів.

У процесі роботи було виконано такі задачі:

1. Проведено аналіз існуючих онлайн-магазинів одягу — виявлено відсутність інструментів персоналізованого підбору за фігурою та кольоротипом. У межах роботи здійснено детальний огляд найпопулярніших платформ з продажу одягу в Україні (Rozetka, Kasta, Oksana Mukha), в ході якого встановлено, що жодна з них не забезпечує можливість підбору товару відповідно до індивідуальних параметрів зовнішності. Це створює бар'єри для користувачів, які прагнуть обрати одяг, що гармонійно пасує саме їм.

2. Визначено ключові параметри фігури (плечі, талія, стегна) та їхній вплив на вибір фасонів. На основі аналізу джерел та логіки підбору одягу виявлено, що найбільший вплив на визначення типу фігури мають лінії плечей, талії та стегон. Врахування саме цих параметрів дає змогу створити класифікацію користувачів за фігурою, що дозволяє персоналізувати рекомендації фасонів вечірніх суконь.

3. Розроблено алгоритм підбору суконь за типом фігури та кольоротипом. Застосовано нечітку логіку для оцінки співвідношення параметрів фігури, що забезпечує гнучке визначення типу тіла — пісочний годинник, груша, прямокутник або перевернутий трикутник. Окрім цього, реалізовано алгоритм визначення кольоротипу на основі комбінації тону шкіри, кольору очей і волосся, що дозволяє враховувати сезонний тип зовнішності при формуванні рекомендацій.

4. Сформовано базу даних користувачів, товарів та параметрів підбору. Базу розроблено з урахуванням принципів нормалізації. Вона включає таблиці

користувачів, суконь, категорій, типів фігури, кольоротипів, замовлень, обраного розміру, кошика, списку бажань тощо. Це забезпечує збереження повного набору інформації, необхідної для роботи системи персоналізованого підбору.

5. Реалізовано web-застосунок з каталогом, фільтрами, підбором, кошиком, вішлістом і особистим кабінетом. Розроблено повноцінний web-застосунок на базі Spring Boot, із підтримкою багаторівневої авторизації, фільтрації товарів, персоналізованого підбору суконь, збереження обраних товарів у кошику чи списку бажань, а також особистим кабінетом із функціями перегляду замовлень та редагування профілю. Для адміністратора реалізовано окрему панель керування товарами та користувачами.

6. Проведено тестування — підтверджено точність підбору, зручність використання та ефективність рішення. Система пройшла функціональне, модульне та кросбраузерне тестування. Оцінено стабільність роботи, коректність логіки підбору типу фігури та кольоротипу, швидкість відгуку й інтуїтивність інтерфейсу. Результати підтверджують доцільність реалізованого рішення та його практичну цінність для користувачів.

Отриманий результат підтверджує практичну ефективність запропонованого рішення. Розроблений web-застосунок дозволяє реалізувати персоналізований підхід до онлайн-продажу суконь, орієнтований на індивідуальні характеристики кожної користувачки.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Рудська Т.А., Золотухіна О.А. Розробка веб-застосунку для магазину вечірніх суконь з автоматизованим підбором товару. // Матеріали VI Міжнародної науково-технічної конференції «Сучасний стан та перспективи розвитку IoT». 15.04.2025, ДУІКТ, Київ, України, (подано до друку).

2. Рудська Т.А., Золотухіна О.А. Функціонал веб-додатків орієнтованих на потреби користувача. // Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». Збірник тез. – К.: ДУІКТ, 2025, С. 67-69.

ПЕРЕЛІК ПОСИЛАНЬ

1. Olawoyin R., Adebiyi A. A., Adekoya A. F. A Review of Modern Fashion Recommender Systems. ResearchGate. 2023. URL: https://www.researchgate.net/publication/374048560_A_Review_of_Modern_Fashion_Recommender_Systems (дата звернення: 15.05.2025).

2. Sheikh A.-S., Guigoures R., Koriagin E., Ho Y. K., Shirvany R., Vollgraf R., Bergmann U. A Deep Learning System for Predicting Size and Fit in Fashion E-Commerce. arXiv preprint. 2020. URL: https://arxiv.org/abs/1907.09844 (дата звернення: 15.05.2025).

3. Raji M. A., Olodo H. B., Oke T. T., Addy W. A., Ofodile O. C., Oyewole A. T. E-commerce and consumer behavior: A review of AI-powered personalization and market trends. *GSC Advanced Research and Reviews*. 2024. Vol. 18(3). P. 66–77. URL: https://gsconlinepress.com/journals/gscarr/sites/default/files/GSCARR-2024-0090.pdf (дата звернення: 15.05.2025).

4. Mei S., et al. Personalized clothing recommendation algorithm based on body features and preference mining. *Proceedings of the 2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*. 2020. URL: https://ieeexplore.ieee.org/document/8959711 (дата звернення: 15.05.2025).

5. Hsiao W.-L., Grauman K. ViBE: Dressing for Diverse Body Shapes. Meta AI Research. 2020. URL: https://ai.meta.com/research/publications/vibe-dressing-for-diverse-body-shapes/ (дата звернення: 15.05.2025).

6. Zong C. Dress Style Recommendation Based on Female Body Shapes. Cornell University. 2021. URL: https://ecommons.cornell.edu/items/57486bfc-4491-4bd7-b883-87a796672ed3 (дата звернення: 15.05.2025).

7. Wazarkar S., Patil S., Gupta P. S., Singh K., Khandelwal M., Vaishnavi C.V.S., Kotecha K. Advanced Fashion Recommendation System for Different Body Types using Deep Learning Models. *International Journal of Creative Research*

Thoughts. 2022. URL: https://www.researchgate.net/publication/362086487 (дата звернення: 15.05.2025).

8. Suvarna S., Balakrishna K. Enhanced Content-Based Fashion Recommendation System through Deep Ensemble Classifier. *Fashion and Textiles*. 2024. URL: https://fashionandtextiles.springeropen.com/articles/10.1186/s40691-024-00382-y (дата звернення: 15.05.2025).

9. Oksana Mukha – офіційний сайт. URL: https://oksana-mukha.com/ua (дата звернення: 15.05.2025).

10. Rozetka – офіційний сайт. URL: https://rozetka.com.ua/ (дата звернення: 15.05.2025).

11. Kasta – офіційний сайт. URL: https://kasta.ua/ (дата звернення: 15.05.2025).

12. JetBrains. IntelliJ IDEA – The Java IDE for Professional Developers. URL: https://www.jetbrains.com/idea (дата звернення: 16.05.2025).

13. Spring Boot Documentation. URL: https://spring.io/projects/spring-boot (дата звернення: 16.05.2025).

14. Spring Security Reference. URL: https://docs.spring.io/spring-security/ (дата звернення: 16.05.2025).

15. Thymeleaf Documentation. URL: https://www.thymeleaf.org/documentation.html (дата звернення: 16.05.2025).

16. MySQL Developer Guide. URL: https://dev.mysql.com/doc/ (дата звернення: 16.05.2025).

17. Open Server. URL: https://ospanel.io/en/ (дата звернення: 16.05.2025).

18. phpMyAdmin. URL: https://docs.phpmyadmin.net/uk/latest/intro.html (дата звернення: 16.05.2025).

19. Wazarkar S., et al. Optimal Website Design Strategy in E-Commerce. *ResearchGate*. 2023. URL: https://www.researchgate.net/publication/387992372_Optimal_Website_Design_Strategy_in_E-Commerce (дата звернення: 20.05.2025).

20. Responsive Web Design for E-Commerce Platforms. *BrowserStack*. 2023. URL: https://www.browserstack.com/guide/responsive-web-design-for-ecommerce-platforms (дата звернення: 20.05.2025).

21. Testlio. Cross Browser Testing for Web & Mobile Applications. 2025. URL: https://testlio.com/blog/comprehensive-guide-to-cross-browser-testing/ (дата звернення: 20.05.2025).

22. Google Developers. Chrome DevTools Overview. URL: https://developer.chrome.com/docs/devtools/ (дата звернення: 20.05.2025).

23. LambdaTest. Cross Browser Compatibility – Web Apps That Work Universally. 2024. URL: https://www.lambdatest.com/learning-hub/cross-browser-compatibility (дата звернення: 20.05.2025).

24. Seasonal color type as a guideline for selecting best color outfit. 2020. URL: https://digitalcommons.aaru.edu.jo/cgi/viewcontent.cgi?article=1488&context=faa-design (дата звернення: 21.05.2025).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка web-застосунку для магазину вечірніх суконь з автоматизованим підбором товару

Виконала студентка 4 курсу
групи ПД-41
Рудська Таіра Анатоліївна
Керівник роботи
К.т.н, доц, доцент кафедри ІПЗ Золотухіна Оксана Анатоліївна

Київ – 2025

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – автоматизація процесу підбору моделей вечірніх суконь в онлайн-магазині з урахуванням індивідуальних параметрів користувача.
- **Об’єкт дослідження** – процес підбору моделей вечірніх суконь для клієнтів магазину.
- **Предмет дослідження** – програмне забезпечення для автоматизованого підбору моделей вечірніх суконь з урахуванням індивідуальних параметрів користувача.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Провести аналіз існуючих рішень у сфері онлайн-продажу одягу, зокрема систем підбору товарів за параметрами фігури та кольоротипом користувача.
2. Визначити ключові антропометричні параметри для класифікації типів фігури та дослідити їхній вплив на вибір фасонів вечірніх суконь.
3. Розробити алгоритм автоматизованого підбору моделей відповідно до типу фігури та сезонного кольоротипу.
4. Сформувати базу даних для збереження користувачів, товарів і параметрів підбору.
5. Реалізувати веб-застосунок для онлайн-магазину вечірніх суконь із функціоналом каталогу, фільтрації товарів, особистого кабінету та адміністративної панелі.
6. Провести тестування системи, перевірити точність підбору, зручність використання та загальну ефективність розробленого рішення.

АНАЛІЗ АНАЛОГІВ

Функціональність / Показник	**Oksana Mukha**	**Rozetka**	**Kasta**	**Celeste Couture**
Наявність веб-платформи	+	+	+	+
Каталог товарів з фільтрацією (за кольором, категорією)	+	+	+	+
Підбір товару за типом фігури (автоматичний аналіз)	-	-	-	+
Підбір товару за кольоротипом (тон шкіри, очі, волосся)	-	-	-	+
Рекомендований розмір на основі введених параметрів	-	+	-	+
Підсвічування рекомендованого розміру на сторінці товару	-	+	-	+
Підрахунок залишку товару за розміром та блокування кнопки	-	+	+	+
Особистий кабінет користувача з історією перегляду товарів	-	+	-	+

ВИМОГИ ДО WEB-ЗАСТОСУНКУ

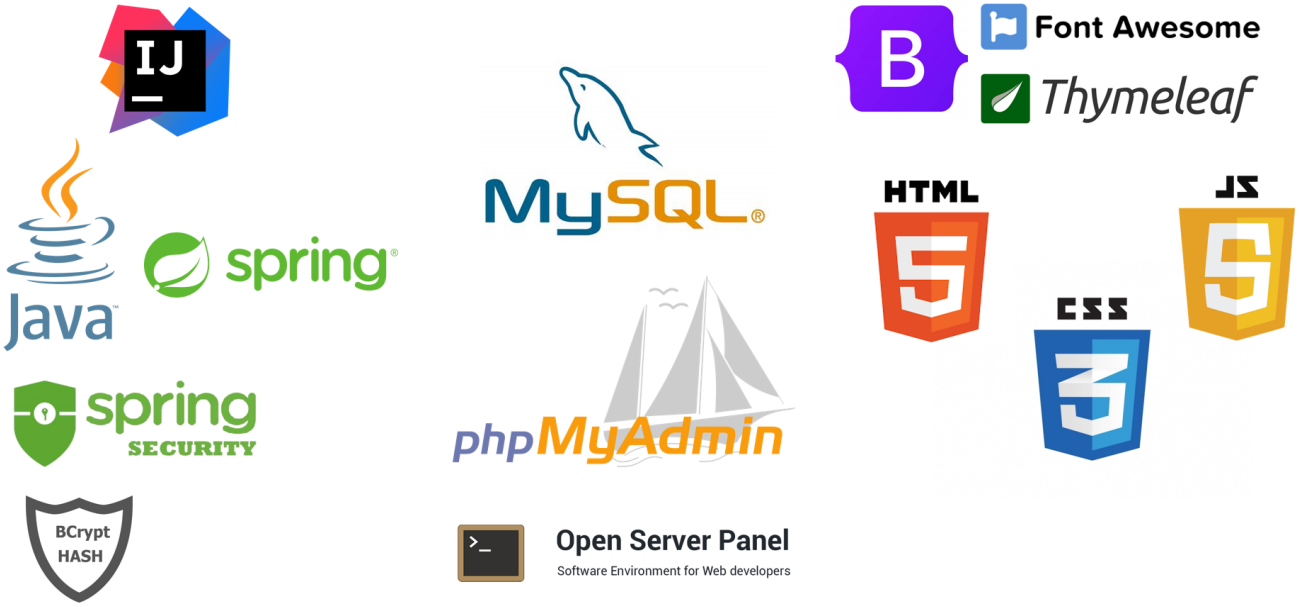
Функціональні вимоги:

- Забезпечити реєстрацію та авторизацію користувачів із розподілом ролей.
- Надати можливість користувачам вводити параметри тіла та зовнішності для визначення типу фігури та кольоротипу.
- Реалізувати автоматичне визначення типу фігури та кольоротипу на основі введених даних.
- Забезпечити автоматичне формування персоналізованого списку рекомендованих товарів.
- Надати доступ до каталогу товарів з фільтрацією за категоріями, кольором, ціною та розміром.
- Надати користувачеві доступ до особистого кабінету з можливістю перегляду історії замовлень.
- Реалізувати адміністративну панель для додавання, редагування та видалення товарів, перегляду користувачів та перегляду замовлень.

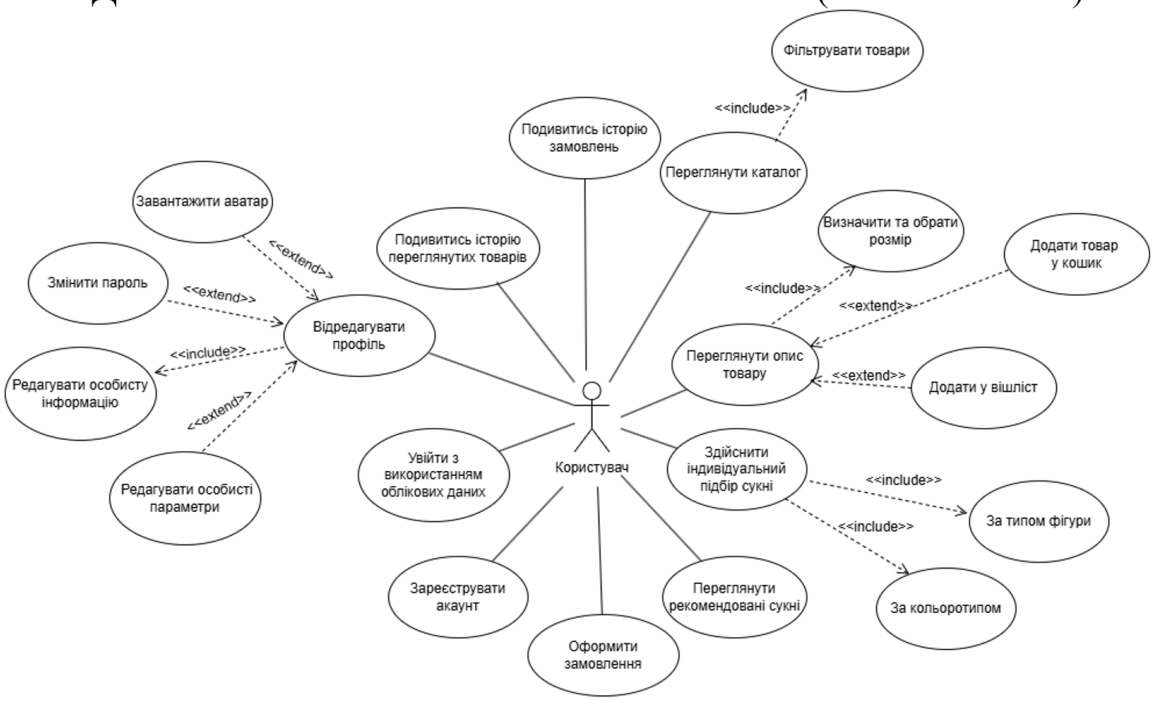
Нефункціональні вимоги:

- Обмежити доступ до адміністративного розділу.
- Забезпечити шифрування паролів з використанням bcrypt-алгоритму.
- Забезпечити стабільну роботу сайту при навантаженні та швидке завантаження сторінок (до 2 секунд).
- Забезпечити кросбраузерну сумісність (підтримка сучасних версій Chrome, Firefox, Opera, Microsoft Edge).

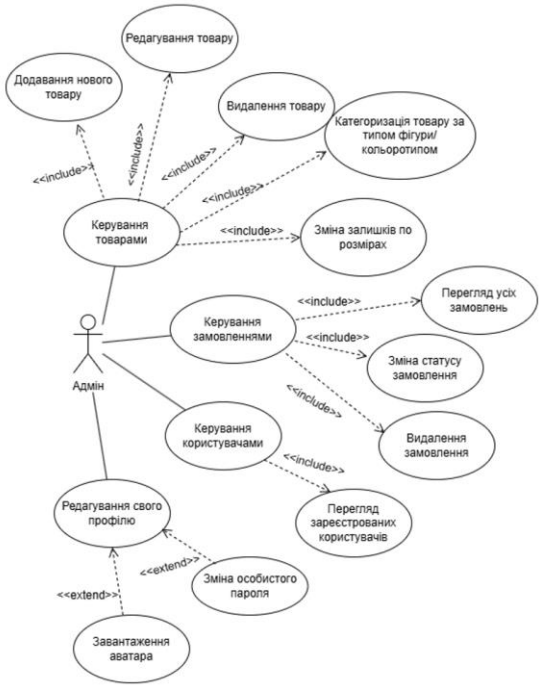
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



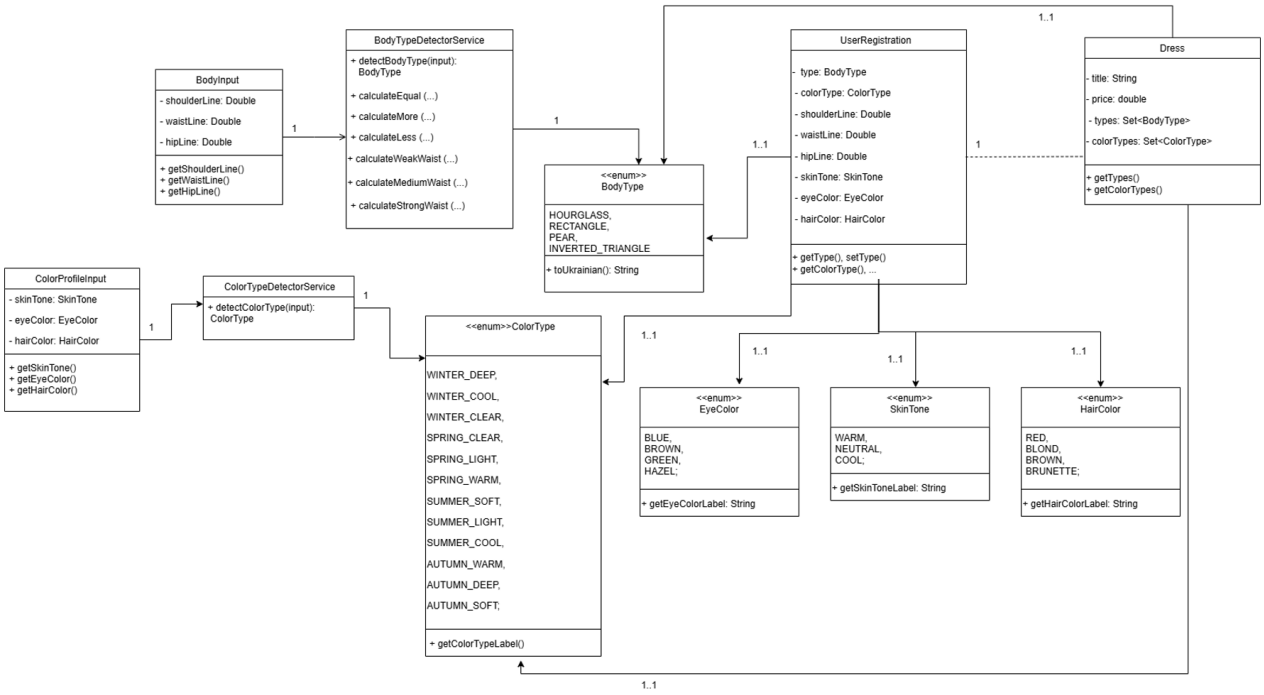
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ (КОРИСТУВАЧ)



ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ (АДМІН)

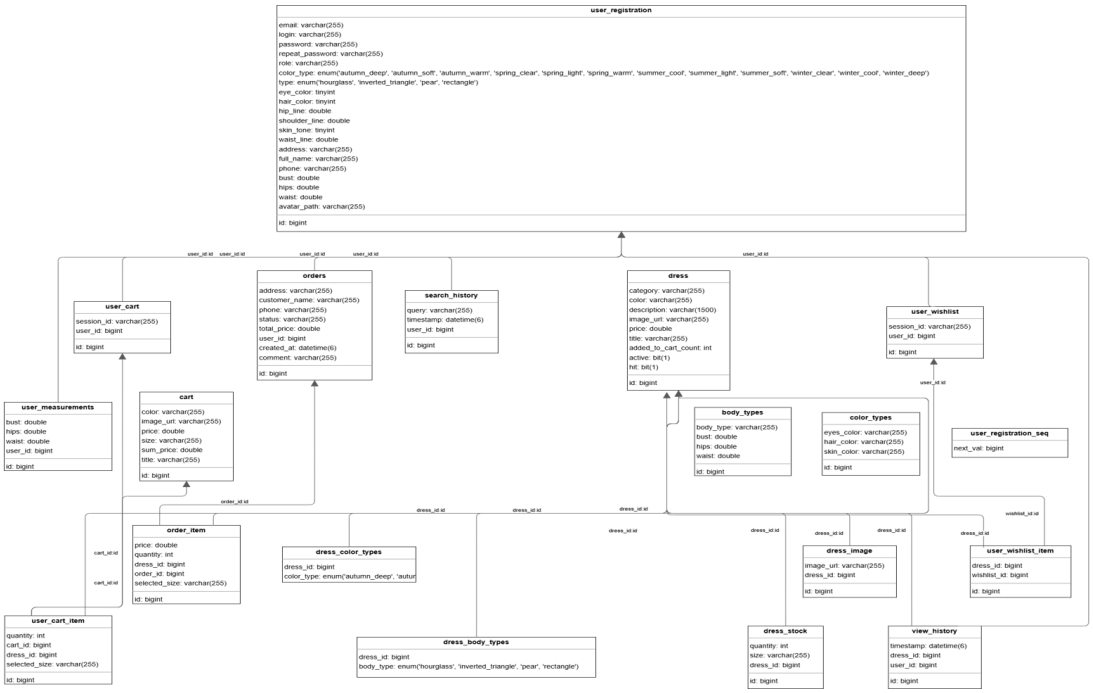


ДІАГРАМА КЛАСІВ ПІДБОРУ ТОВАРІВ ЗА ТИПОМ ФІГУРИ ТА КОЛЬОРОТИПОМ



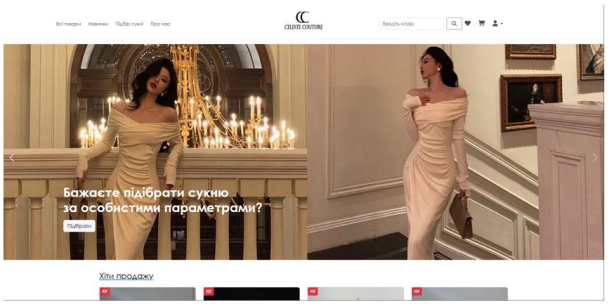
9

ДІАГРАМА СТРУКТУРИ БАЗИ ДАНИХ

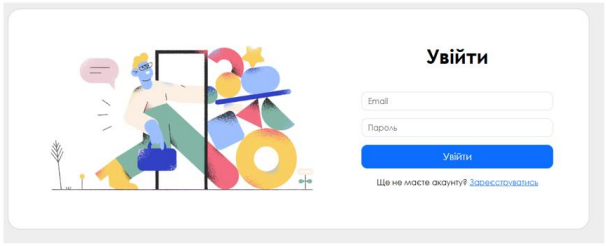


10

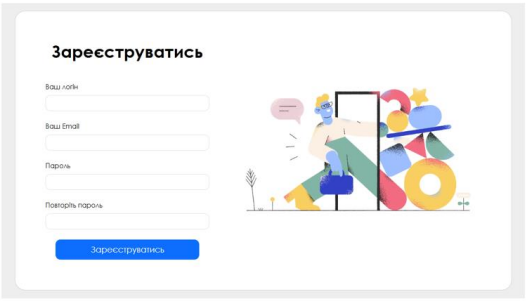
ЕКРАННІ ФОРМИ



Головна сторінка сайту

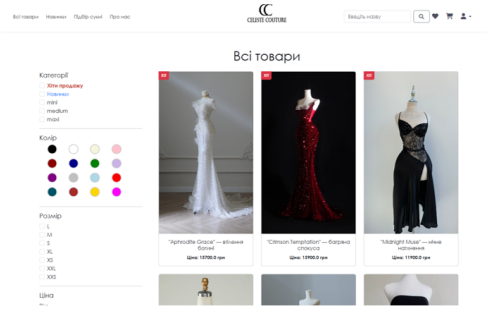


Вхід до акаунту

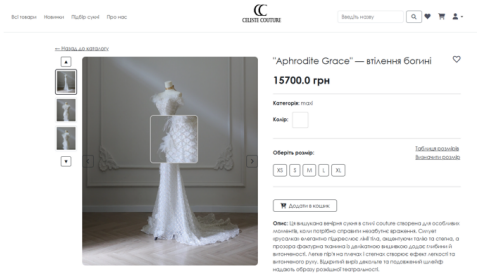


Реєстрація

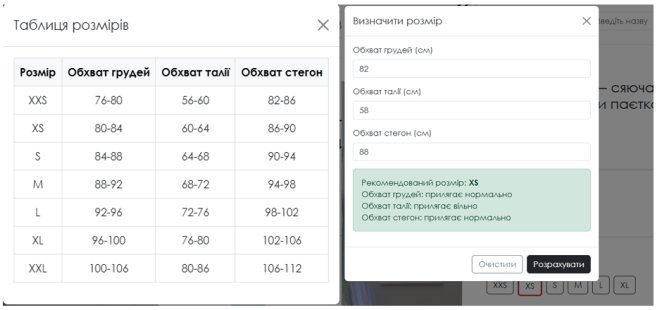
ЕКРАННІ ФОРМИ



Каталог товарів та фільтр

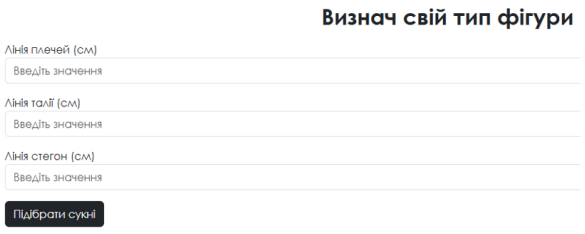


Сторінка опису товару

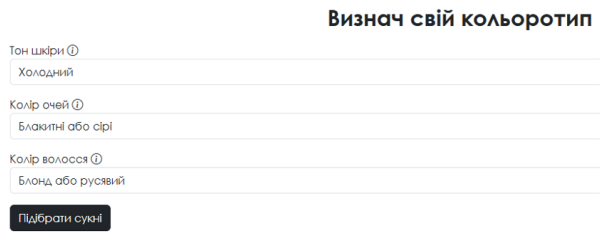


Модальне вікно «Таблиця розмірів» та Модальне вікно «Визначити розмір»

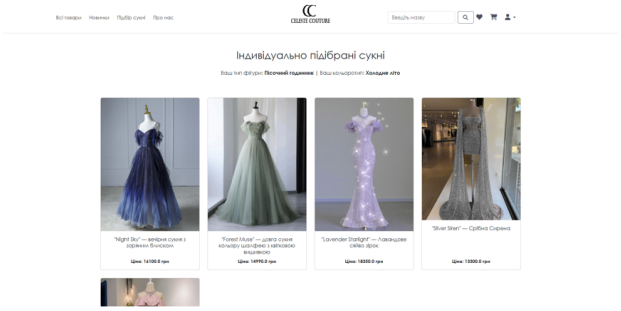
ЕКРАННІ ФОРМИ



Форма введення параметрів фігури

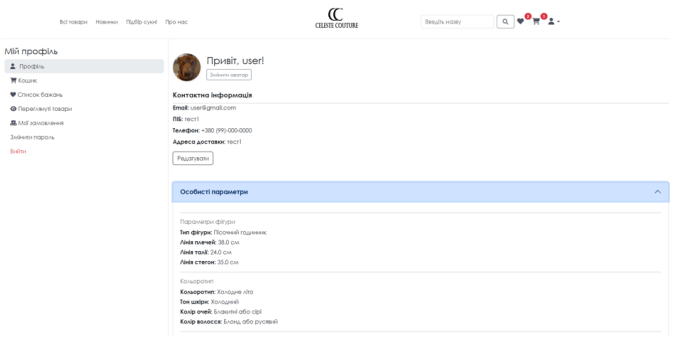


Форма введення параметрів зовнішності

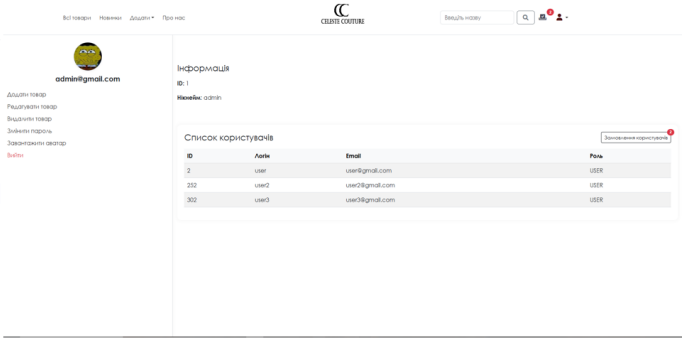


Сторінка з результатами індивідуального підбору сукні

ЕКРАННІ ФОРМИ



Особистий кабінет користувача



Панель адміністратора

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Рудська Т.А., Золотухіна О.А. Розробка веб-застосунку для магазину вечірніх суконь з автоматизованим підбором товару. // Матеріали VI Міжнародної науково-технічної конференції «Сучасний стан та перспективи розвитку IoT». Збірник тез. – К.: ДУІКТ, 2025, (подано до друку).
2. Рудська Т.А., Золотухіна О.А. Функціонал веб-додатків орієнтованих на потреби користувача. // Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». Збірник тез. – К.: ДУІКТ, 2025, С. 67-69

ВИСНОВКИ

1. Проведено аналіз існуючих онлайн-магазинів одягу — виявлено відсутність інструментів персоналізованого підбору за фігурою та кольоротипом.
2. Визначено ключові параметри фігури (плечі, талія, стегна) та їхній вплив на вибір фасонів.
3. Розроблено алгоритм підбору суконь за типом фігури та кольоротипом.
4. Сформовано базу даних користувачів, товарів та параметрів підбору.
5. Реалізовано веб-застосунок з каталогом, фільтрами, підбором, кошиком, вішлістом і особистим кабінетом.
6. Проведено тестування — підтверджено точність підбору, зручність використання та ефективність рішення.

ДОДАТОК Б. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

```
package com.dyplom.dyplom.service;
import com.dyplom.dyplom.dto.BodyInput;
import com.dyplom.dyplom.models.BodyType;
import org.springframework.stereotype.Service;
import java.util.HashMap;
import java.util.Map;
@Service
public class BodyTypeDetectorService {
public BodyType detectBodyType(BodyInput input) {
double shoulder = input.getShoulderLine();
double waist = input.getWaistLine();
double hip = input.getHipLine();
double shoulderHipDiff = shoulder - hip;
double minSide = Math.min(shoulder, hip);
double waistDiff = minSide - waist;
double equalShoulderHip = calculateEqual(shoulderHipDiff);
double widerShoulders = calculateMore(shoulderHipDiff);
double widerHips = calculateLess(shoulderHipDiff);
double waistStrong = calculateStrongWaist(waistDiff);
double waistMedium = calculateMediumWaist(waistDiff);
double waistWeak = calculateWeakWaist(waistDiff);
// fuzzy inference rules
double hourglassScore = Math.min(equalShoulderHip,
waistStrong);
double rectangleScore = Math.min(equalShoulderHip,
Math.max(waistMedium, waistWeak));
double pearScore = 0.0;
double invertedTriangleScore = Math.min(widerShoulders,
Math.max(waistMedium, waistWeak));
if (shoulderHipDiff <= -5 && (waistStrong >= 0.8 ||
waistMedium >= 1.0)) {
pearScore = 1.0;
hourglassScore = 0.0;
rectangleScore = 0.0;
invertedTriangleScore = 0.0;
}
if (Math.abs(shoulderHipDiff) > 5) {
hourglassScore = 0.0;
rectangleScore = 0.0;
}
if (waistStrong >= 1.0) {
rectangleScore = 0.0;
}
Map<BodyType, Double> scores = new HashMap<>();
scores.put(BodyType.HOURGLASS, hourglassScore);
scores.put(BodyType.RECTANGLE, rectangleScore);
scores.put(BodyType.PEAR, pearScore);
scores.put(BodyType.INVERTED_TRIANGLE,
invertedTriangleScore);
return scores.entrySet()
.stream()
.max((a, b) -> {
int cmp = Double.compare(a.getValue(), b.getValue());
if (cmp != 0) return cmp;
if (a.getKey() == BodyType.PEAR) return 1;
if (b.getKey() == BodyType.PEAR) return -1;
if (a.getKey() == BodyType.HOURGLASS) return 1;
if (b.getKey() == BodyType.HOURGLASS) return -1;
return 0;
})
.orElseThrow()
.getKey();
}
private double calculateEqual(double diff) {
return 1.0 - Math.min(Math.abs(diff) / 5.0, 1.0);
}
private double calculateMore(double diff) {
return diff <= 5 ? 0.0 : Math.min((diff - 5) / 10.0, 1.0);
}
private double calculateLess(double diff) {
return diff >= -5 ? 0.0 : Math.min((-5 - diff) / 10.0, 1.0);
}
private double calculateStrongWaist(double diff) {
return diff >= 10 ? 1.0 : (diff < 8 ? 0.0 : (diff - 8) / 2.0);
}
private double calculateMediumWaist(double diff) {
if (diff >= 6 && diff <= 10) return 1.0;
if (diff > 10 && diff <= 12) return 1.0 - (diff - 10) / 2.0;
if (diff >= 4 && diff < 6) return (diff - 4) / 2.0;
return 0.0;
}
private double calculateWeakWaist(double diff) {
return diff < 6 ? 1.0 - Math.min(diff / 6.0, 1.0) : 0.0; }
}
```

```
package com.dyplom.dyplom.service;
import com.dyplom.dyplom.dto.ColorProfileInput;
import com.dyplom.dyplom.models.ColorType;
import com.dyplom.dyplom.models.EyeColor;
import com.dyplom.dyplom.models.HairColor;
import com.dyplom.dyplom.models.SkinTone;
import org.springframework.stereotype.Service;
@Service
public class ColorTypeDetectorService {
public ColorType detectColorType(ColorProfileInput input) {
SkinTone tone = input.getSkinTone();
EyeColor eyes = input.getEyeColor();
HairColor hair = input.getHairColor();
if ((tone == SkinTone.COOL || tone == SkinTone.NEUTRAL)
&&
(hair == HairColor.BRUNETTE || hair ==
HairColor.BROWN)) {
if (eyes == EyeColor.BROWN || eyes == EyeColor.HAZEL)
return ColorType.WINTER_DEEP;
if (eyes == EyeColor.GREEN) return
ColorType.WINTER_CLEAR;
if (eyes == EyeColor.BLUE) return
ColorType.WINTER_COOL;
}
if ((tone == SkinTone.WARM || tone ==
SkinTone.NEUTRAL) &&
(hair == HairColor.BLOND || hair == HairColor.RED)) {
if (eyes == EyeColor.BROWN || eyes == EyeColor.HAZEL)
return ColorType.SPRING_WARM;
if (eyes == EyeColor.GREEN) return
ColorType.SPRING_LIGHT;
if (eyes == EyeColor.BLUE) return
ColorType.SPRING_CLEAR;
}
if ((tone == SkinTone.COOL || tone == SkinTone.NEUTRAL)
&&
(hair == HairColor.BLOND || hair == HairColor.RED)) {
if (eyes == EyeColor.GREEN) return
ColorType.SUMMER_LIGHT;
```

```
if (eyes == EyeColor.HAZEL || eyes == EyeColor.BROWN)
return ColorType.SUMMER_SOFT;
if (eyes == EyeColor.BLUE) return
ColorType.SUMMER_COOL;
}
if ((tone == SkinTone.WARM || tone ==
SkinTone.NEUTRAL) &&
(hair == HairColor.BROWN || hair == HairColor.RED || hair
== HairColor.BRUNETTE)) {
if (eyes == EyeColor.HAZEL || eyes == EyeColor.BROWN)
return ColorType.AUTUMN_WARM;
if (eyes == EyeColor.GREEN) return
ColorType.AUTUMN_SOFT;
if (eyes == EyeColor.BLUE) return
ColorType.AUTUMN_DEEP; }
return null; }
}
```

```
package com.dyplom.dyplom.controllers;
import com.dyplom.dyplom.models.BodyType;
import com.dyplom.dyplom.models.ColorType;
import com.dyplom.dyplom.models.Dress;
import com.dyplom.dyplom.models.UserRegistration;
import com.dyplom.dyplom.repository.DressRepository;
import
com.dyplom.dyplom.repository.UserRegistrationRepository;
import jakarta.servlet.http.HttpSession;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PostMapping;
import java.util.List;
import java.util.Optional;
@Controller
public class RecommendationController {
@Autowired
private DressRepository dressRepository;
@Autowired
private UserRegistrationRepository userRepository;
@GetMapping("/recommendation")
public String showRecommendations(HttpSession session,
Model model, Authentication authentication) {
BodyType bodyType = null;
ColorType colorType = null;
if (authentication != null && authentication.isAuthenticated()
&& !"anonymousUser".equals(authentication.getName())) {
String email = authentication.getName();
Optional<UserRegistration> userOpt =
userRepository.findByEmail(email);
if (userOpt.isPresent()) {
UserRegistration user = userOpt.get();
bodyType = user.getType();
colorType = user.getColorType();
}
} else {
bodyType = (BodyType) session.getAttribute("bodyType");
colorType = (ColorType) session.getAttribute("colorType");
}
if (bodyType == null || colorType == null) {
return "redirect:/";
}
List<Dress> dresses =
dressRepository.findByBodyTypeAndColorType(bodyType,
colorType);
model.addAttribute("dresses", dresses);
model.addAttribute("type", bodyType.toUkrainian());
model.addAttribute("colorTypeLabel",
ColorType.getColorTypeLabel(colorType));
return "recommendation_result";
}
@PostMapping("/reset-session-parameters")
public String resetSessionParameters(HttpSession session) {
session.removeAttribute("bodyType");
session.removeAttribute("colorType");
session.removeAttribute("bodyInput");
session.removeAttribute("colorInput");
return "redirect:/"; }
}
package com.dyplom.dyplom.controllers;
import com.dyplom.dyplom.models.UserRegistration;
import
com.dyplom.dyplom.repository.UserRegistrationRepository;
import jakarta.servlet.http.HttpSession;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.Authentication;
import org.springframework.web.bind.annotation.*;
import java.util.HashMap;
import java.util.Map;
import java.util.Optional;
@RestController
@RequestMapping("/api/user/measurements")
public class MeasurementsRestController {
@Autowired
private UserRegistrationRepository userRepository;
@PostMapping
public ResponseEntity<?> saveMeasurements(@RequestBody
Map<String, Object> body,
Authentication auth,
HttpSession session) {
Number bustNumber = (Number) body.get("bust");
Number waistNumber = (Number) body.get("waist");
Number hipsNumber = (Number) body.get("hips");
Double bust = bustNumber != null ?
bustNumber.doubleValue() : null;
Double waist = waistNumber != null ?
waistNumber.doubleValue() : null;
Double hips = hipsNumber != null ?
hipsNumber.doubleValue() : null;
String size = (String) body.get("recommendedSize");
if (auth != null && auth.isAuthenticated()) {
String email = auth.getName();
Optional<UserRegistration> userOpt =
userRepository.findByEmail(email);
userOpt.ifPresent(user -> {
user.setBust(bust);
user.setWaist(waist);
user.setHips(hips);
userRepository.save(user);
});
} else {
session.setAttribute("anon_bust", bust);
session.setAttribute("anon_waist", waist);
session.setAttribute("anon_hips", hips);
session.setAttribute("anon_recommended_size", size);
}
return ResponseEntity.ok().build();
}
@PostMapping("/clear")
public ResponseEntity<?> clearMeasurements(Authentication
auth, HttpSession session) {
if (auth != null && auth.isAuthenticated()) {
String email = auth.getName();
```

```
Optional<UserRegistration> userOpt =
userRepository.findByEmail(email);
if (userOpt.isPresent()) {
UserRegistration user = userOpt.get();
user.setBust(null);
user.setWaist(null);
user.setHips(null);
userRepository.save(user);
}
} else {
session.removeAttribute("anon_bust");
session.removeAttribute("anon_waist");
session.removeAttribute("anon_hips");
}
return ResponseEntity.ok().build();
}
@GetMapping
public ResponseEntity<Map<String, Object>>
getMeasurements(Authentication auth, HttpSession session) {
Map<String, Object> result = new HashMap<>();
if (auth != null && auth.isAuthenticated()) {
String email = auth.getName();
Optional<UserRegistration> userOpt =
userRepository.findByEmail(email);
userOpt.ifPresent(user -> {
result.put("bust", user.getBust());
result.put("waist", user.getWaist());
result.put("hips", user.getHips());
result.put("recommendedSize", null);
});
} else {
result.put("bust", session.getAttribute("anon_bust"));
result.put("waist", session.getAttribute("anon_waist"));
result.put("hips", session.getAttribute("anon_hips"));
result.put("recommendedSize",
session.getAttribute("anon_recommended_size")); }
return ResponseEntity.ok(result); }
}
```

```
<script>
const sizeTable = [
{ size: "XXS", bust: [76, 80], waist: [56, 60], hips: [82, 86] },
{ size: "XS", bust: [80, 84], waist: [60, 64], hips: [86, 90] },
{ size: "S", bust: [84, 88], waist: [64, 68], hips: [90, 94] },
{ size: "M", bust: [88, 92], waist: [68, 72], hips: [94, 98] },
{ size: "L", bust: [92, 96], waist: [72, 76], hips: [98, 102] },
{ size: "XL", bust: [96, 100], waist: [76, 80], hips: [102, 106] },
{ size: "XXL", bust: [100, 106], waist: [80, 86], hips: [106,
112] }
];
function getFit(value, [min, max]) {
if (value < min) return "вільно";
if (value > max) return "щільно";
return "нормально";
}
function calculateRecommendedSize() {
const bust =
parseFloat(document.getElementById("bust").value);
const waist =
parseFloat(document.getElementById("waist").value);
const hips =
parseFloat(document.getElementById("hips").value);
if (isNaN(bust) || isNaN(waist) || isNaN(hips)) {
document.getElementById("recommendationResult").innerHT
ML = '';
highlightSizeButton(null);
return;
}
let bestMatch = null;
let maxMatchCount = -1;
for (let i = 0; i < sizeTable.length; i++) {
const row = sizeTable[i];
let matchCount = 0;
if (bust >= row.bust[0] && bust <= row.bust[1])
matchCount++;
if (waist >= row.waist[0] && waist <= row.waist[1])
matchCount++;
if (hips >= row.hips[0] && hips <= row.hips[1])
matchCount++;
if (
matchCount > maxMatchCount ||
(matchCount === maxMatchCount && i >
sizeTable.findIndex(r => r.size === bestMatch))
) {
maxMatchCount = matchCount;
bestMatch = row.size;
}
}
const recommendedSize = bestMatch;
const resultHTML = recommendedSize ? `
<div class="alert alert-success">
Рекомендований розмір:
<strong>${recommendedSize}</strong><br>
Обхват грудей: прилягає ${getFit(bust, sizeTable.find(r =>
r.size === recommendedSize).bust)}<br>
Обхват талії: прилягає ${getFit(waist, sizeTable.find(r =>
r.size === recommendedSize).waist)}<br>
Обхват стегон: прилягає ${getFit(hips, sizeTable.find(r =>
r.size === recommendedSize).hips)}
</div>` :
`<div class="alert alert-warning">Не вдалося підібрати
розмір за введеними параметрами.</div>`;
document.getElementById("recommendationResult").innerHT
ML = resultHTML;
highlightSizeButton(recommendedSize);
localStorage.setItem("anon_bust", bust);
localStorage.setItem("anon_waist", waist);
localStorage.setItem("anon_hips", hips);
localStorage.setItem("anon_recommended_size",
recommendedSize);
fetch('/api/user/measurements', {
method: 'POST',
headers: { 'Content-Type': 'application/json' },
body: JSON.stringify({ bust, waist, hips, recommendedSize })
});
}
function clearMeasurements() {
sessionStorage.removeItem("bust");
sessionStorage.removeItem("waist");
sessionStorage.removeItem("hips");
sessionStorage.removeItem("recommendedSize");
document.getElementById("bust").value = '';
document.getElementById("waist").value = '';
document.getElementById("hips").value = '';
document.getElementById("recommendationResult").innerHT
ML = '';
fetch('/api/user/measurements/clear', { method: 'POST' });
document.querySelectorAll(".size-btn").forEach(btn => {
btn.classList.remove("border-danger", "border-3");
});
}
function highlightSizeButton(size) {
if (!size) return;
document.querySelectorAll(".size-btn").forEach(btn => {
if (btn.dataset.size === size) {
btn.classList.add("border-danger", "border-3");
```

```
} else {
btn.classList.remove("border-danger", "border-3");
}
});
}
function restoreMeasurements() {
fetch('/api/user/measurements')
.then(response => response.json())
.then(data => {
let bust = data.bust ?? sessionStorage.getItem("bust");
let waist = data.waist ?? sessionStorage.getItem("waist");
let hips = data.hips ?? sessionStorage.getItem("hips");
let recommendedSize = data.recommendedSize ??
sessionStorage.getItem("recommendedSize");
if (bust) document.getElementById("bust").value = bust;
if (waist) document.getElementById("waist").value = waist;
if (hips) document.getElementById("hips").value = hips;
if (bust && waist && hips) {
calculateRecommendedSize();
} else if (recommendedSize) {
highlightSizeButton(recommendedSize);
const fitBust = getFit(parseFloat(bust), sizeTable.find(r =>
r.size === recommendedSize).bust);
const fitWaist = getFit(parseFloat(waist), sizeTable.find(r =>
r.size === recommendedSize).waist);
const fitHips = getFit(parseFloat(hips), sizeTable.find(r =>
r.size === recommendedSize).hips);
document.getElementById("recommendationResult").innerHT
ML = `
<div class="alert alert-success">
Рекомендований розмір:
<strong>${recommendedSize}</strong><br>
Обхват грудей: прилягає ${fitBust}<br>
Обхват талії: прилягає ${fitWaist}<br>
package com.dyplom.dyplom.service;
import com.dyplom.dyplom.models.*;
import com.dyplom.dyplom.repository.*;
import jakarta.servlet.http.HttpSession;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Service;
import java.util.*;
@Service
public class CartService {
@Autowired
private UserCartRepository cartRepository;
@Autowired
private UserCartItemRepository itemRepository;
@Autowired
private UserRegistrationRepository userRepository;
@Autowired
private DressRepository dressRepository;
public UserCart getOrCreateCart(HttpSession session,
Authentication auth) {
if (auth != null && auth.isAuthenticated()) {
String email = auth.getName();
UserRegistration user =
userRepository.findByEmail(email).orElse(null);
if (user == null) return null;
return cartRepository.findByUser(user)
.orElseGet(() -> {
UserCart cart = new UserCart();
cart.setUser(user);
return cartRepository.save(cart);
});
} else {
Object sessionAttr =
session.getAttribute("ANONYMOUS_CART_ID");
Обхват стегон: прилягає ${fitHips}
</div>`; }
}); }
document.addEventListener("DOMContentLoaded", () => {
updateCartCount();
updateWishlistCount();
restoreMeasurements();
const bustInput = document.getElementById("bust");
const waistInput = document.getElementById("waist");
const hipsInput = document.getElementById("hips");
bustInput.addEventListener("input", () => {
const value = parseFloat(bustInput.value);
document.getElementById("error-bust").classList.toggle("d-
none", value >= 50 && value <= 232);
});
waistInput.addEventListener("input", () => {
const value = parseFloat(waistInput.value);
document.getElementById("error-waist").classList.toggle("d-
none", value >= 40 && value <= 200);
});
hipsInput.addEventListener("input", () => {
const value = parseFloat(hipsInput.value);
document.getElementById("error-hips").classList.toggle("d-
none", value >= 50 && value <= 232);
});
const sizeModalTrigger = document.querySelector('[data-bs-
target="#sizeRecommendModal"]');
if (sizeModalTrigger) {
sizeModalTrigger.addEventListener("click",
restoreMeasurements);
}
});
</script>

String sessionId = sessionAttr != null ? sessionAttr.toString() :
null;
if (sessionId == null) {
sessionId = UUID.randomUUID().toString();
session.setAttribute("ANONYMOUS_CART_ID", sessionId);
}
final String finalSessionId = sessionId;
return cartRepository.findBySessionId(finalSessionId)
.orElseGet(() -> {
UserCart cart = new UserCart();
cart.setSessionId(finalSessionId);
return cartRepository.save(cart);
});
}
}
public void addToCart(Long dressId, String size, HttpSession
session, Authentication auth) {
Dress dress = dressRepository.findById(dressId).orElse(null);
if (dress == null || size == null || size.isBlank()) return;
UserCart cart = getOrCreateCart(session, auth);
if (cart == null) return;
for (UserCartItem item : cart.getItems()) {
if (item.getDress().getId().equals(dressId) &&
size.equals(item.getSelectedSize())) {
item.setQuantity(item.getQuantity() + 1);
itemRepository.save(item);
return;
}
}
UserCartItem newItem = new UserCartItem();
newItem.setCart(cart);
newItem.setDress(dress);
newItem.setSelectedSize(size);
newItem.setQuantity(1);
itemRepository.save(newItem);
```

```
cart.getItems().add(newItem);
cartRepository.save(cart);
}
public void updateQuantity(Long dressId, String size, int
quantity, HttpSession session, Authentication auth) {
UserCart cart = getOrCreateCart(session, auth);
if (cart == null) return;
for (UserCartItem item : cart.getItems()) {
if (item.getDress().getId().equals(dressId) &&
size.equals(item.getSelectedSize())) {
item.setQuantity(quantity);
itemRepository.save(item);
break;
} }
cartRepository.save(cart);
}
public void removeFromCart(Long dressId, String size,
HttpSession session, Authentication auth) {
UserCart cart = getOrCreateCart(session, auth);
if (cart == null) return;
UserCartItem toRemove = null;
for (UserCartItem item : cart.getItems()) {
if (item.getDress().getId().equals(dressId) &&
size.equals(item.getSelectedSize())) {
toRemove = item;
break;
} }
if (toRemove != null) {
cart.removeItem(toRemove);
itemRepository.delete(toRemove);
cartRepository.save(cart);
}
}
public void mergeCarts(HttpSession session, Authentication
auth) {}
public void clearCart(HttpSession session, Authentication auth)
{
UserCart cart = getOrCreateCart(session, auth);
if (cart == null) return;
for (UserCartItem item : cart.getItems()) {
itemRepository.delete(item);
}
cart.getItems().clear();
cartRepository.save(cart);
}
public double calculateTotal(List<UserCartItem> items) {
double sum = 0.0;
for (UserCartItem item : items) {
if (item.getDress() != null) {
sum += item.getDress().getPrice() * item.getQuantity();
} }
return sum;
} }
package com.dyplom.dyplom.service;
import com.dyplom.dyplom.models.*;
import com.dyplom.dyplom.repository.*;
import jakarta.servlet.http.HttpSession;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Service;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
@Service
public class OrderService {
@Autowired
private OrderRepository orderRepository;
@Autowired
private DressStockRepository dressStockRepository;
@Autowired
private CartService cartService;
@Autowired
private DressRepository dressRepository;
public Order createOrder(UserRegistration user, String
fullName, String address, String phone,String comment,
HttpSession session, Authentication auth) {
UserCart cart = cartService.getOrCreateCart(session, auth);
if (cart == null || cart.getItems().isEmpty()) {
throw new IllegalStateException("Кошик порожній");
}
Order order = new Order();
order.setUser(user);
order.setCustomerName(fullName);
order.setAddress(address);
order.setPhone(phone);
order.setComment(comment);
order.setStatus("Очікує");
order.setCreatedAt(LocalDateTime.now());
List<OrderItem> orderItems = new ArrayList<>();
double totalPrice = 0;
for (UserCartItem cartItem : cart.getItems()) {
Dress dress = cartItem.getDress();
DressStock stock =
dressStockRepository.findByDressIdAndSize(dress.getId(),
cartItem.getSelectedSize())
.orElseThrow(() -> new IllegalStateException("Розмір
недоступний"));
if (stock.getQuantity() < cartItem.getQuantity()) {
throw new IllegalStateException("Недостатня кількість
товару для розміру " + cartItem.getSelectedSize());
}
stock.setQuantity(stock.getQuantity() - cartItem.getQuantity());
dressStockRepository.save(stock);
OrderItem orderItem = new OrderItem();
orderItem.setOrder(order);
orderItem.setDress(dress);
orderItem.setSelectedSize(cartItem.getSelectedSize());
orderItem.setQuantity(cartItem.getQuantity());
orderItem.setPrice(dress.getPrice() * cartItem.getQuantity());
orderItems.add(orderItem);
totalPrice += orderItem.getPrice();
}
order.setItems(orderItems);
order.setTotalPrice(totalPrice);
orderRepository.save(order);
cartService.clearCart(session, auth);// лишаємо auth
for (OrderItem item : orderItems) {
Dress dress = item.getDress();
long totalPurchases = orderRepository.findAll().stream()
.flatMap(o -> o.getItems().stream())
.filter(i -> i.getDress().getId().equals(dress.getId()))
.mapToInt(OrderItem::getQuantity)
.sum();
if (totalPurchases >= 10 && !dress.isHit()) {
dress.setHit(true);
dressRepository.save(dress);
}
}
return order;
}
public List<Order> getOrdersByUser(UserRegistration user) {
return orderRepository.findByUser(user);
}
public List<Order> getAllOrders() {
List<Order> all = orderRepository.findAll();
all.sort((o1, o2) -> {
```

```
List<String> priority = List.of("Очікує", "Підтверджено",
"Скасовано");
return Integer.compare(
priority.indexOf(o1.getStatus()),
priority.indexOf(o2.getStatus())
);
});
return all;
}
public void updateOrderStatus(Long orderId, String status) {
Order order =
orderRepository.findById(orderId).orElseThrow();
order.setStatus(status);
orderRepository.save(order);
}
public void deleteOrderById(Long id) {
orderRepository.deleteById(id);
}
public void deleteAllOrders() {
orderRepository.deleteAll();
}
public void updateStatusForAll(String fromStatus, String
toStatus) {
List<Order> orders =
orderRepository.findByStatus(fromStatus);
for (Order order : orders) {
order.setStatus(toStatus);
}
orderRepository.saveAll(orders);
}
}
```

```
package com.dyplom.dyplom.service;
import com.dyplom.dyplom.models.*;
import com.dyplom.dyplom.repository.*;
import jakarta.servlet.http.HttpSession;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Service;
import java.util.*;
@Service
public class WishlistService {
@Autowired
private UserWishlistRepository wishlistRepository;
@Autowired
private UserWishlistItemRepository wishlistItemRepository;
@Autowired
private UserRegistrationRepository userRepository;
@Autowired
private DressRepository dressRepository;
@Autowired
private CartService cartService;
public UserWishlist getOrCreateWishlist(HttpSession session,
Authentication auth) {
if (auth != null && auth.isAuthenticated()) {
String email = auth.getName();
UserRegistration user =
userRepository.findByEmail(email).orElse(null);
if (user == null) return null;
return wishlistRepository.findByUser(user)
.orElseGet(() -> {
UserWishlist wishlist = new UserWishlist();
wishlist.setUser(user);
return wishlistRepository.save(wishlist);
});
} else {
String sessionId = (String)
session.getAttribute("ANONYMOUS_WISHLIST_ID");
if (sessionId == null) {
sessionId = UUID.randomUUID().toString();
session.setAttribute("ANONYMOUS_WISHLIST_ID",
sessionId);
}
final String finalSessionId = sessionId;
return wishlistRepository.findBySessionId(finalSessionId)
.orElseGet(() -> {
UserWishlist wishlist = new UserWishlist();
wishlist.setSessionId(finalSessionId);
return wishlistRepository.save(wishlist);
});
}
}
public void addToWishlist(Long dressId, HttpSession session,
Authentication auth) {
Dress dress = dressRepository.findById(dressId).orElse(null);
if (dress == null) return;
UserWishlist wishlist = getOrCreateWishlist(session, auth);
if (wishlist == null) return;
boolean exists = wishlist.getItems().stream()
.anyMatch(item -> item.getDress().getId().equals(dressId));
if (!exists) {
UserWishlistItem item = new UserWishlistItem();
item.setDress(dress);
item.setWishlist(wishlist);
wishlist.addItem(item);
wishlistItemRepository.save(item);
wishlistRepository.save(wishlist);
}
}
public void removeFromWishlist(Long dressId, HttpSession
session, Authentication auth) {
UserWishlist wishlist = getOrCreateWishlist(session, auth);
if (wishlist == null) return;
wishlist.getItems().removeIf(item ->
item.getDress().getId().equals(dressId));
wishlistRepository.save(wishlist);
}
public void clearWishlist(HttpSession session, Authentication
auth) {
UserWishlist wishlist = getOrCreateWishlist(session, auth);
if (wishlist == null) return;
for (UserWishlistItem item : wishlist.getItems()) {
wishlistItemRepository.delete(item);
}
wishlist.getItems().clear();
wishlistRepository.save(wishlist);
}
public void moveItemToCart(Long dressId, String size,
HttpSession session, Authentication auth) {
if (size == null || size.isBlank()) {
throw new IllegalArgumentException("Розмір обов'язковий
для додавання в кошик.");
}
removeFromWishlist(dressId, session, auth);
cartService.addToCart(dressId, size, session, auth);
}
public void moveAllToCart(HttpSession session,
Authentication auth, String size) {
if (size == null || size.isBlank()) {
throw new IllegalArgumentException("Розмір обов'язковий
для додавання в кошик.");
}
UserWishlist wishlist = getOrCreateWishlist(session, auth);
if (wishlist == null) return;
for (UserWishlistItem item : wishlist.getItems()) {
cartService.addToCart(item.getDress().getId(), size, session,
auth);
```

```
}
clearWishlist(session, auth);
}
public int getWishlistCount(HttpSession session,
Authentication auth) {
UserWishlist wishlist = getOrCreateWishlist(session, auth);
return wishlist != null ? wishlist.getItems().size() : 0;
}
public int getItemCount(HttpSession session, Authentication
auth) {
return getWishlistCount(session, auth);
}
public boolean isInWishlist(Long dressId, HttpSession session,
Authentication auth) {
UserWishlist wishlist = getOrCreateWishlist(session, auth);
return wishlist.getItems().stream()
.anyMatch(item -> item.getDress().getId().equals(dressId));
}
}
package com.dyplom.dyplom.security;
import
com.dyplom.dyplom.service.CustomUserDetailsService;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.authentication.AuthenticationMa
nager;
import
org.springframework.security.config.annotation.web.builders.H
ttpSecurity;
import
org.springframework.security.config.annotation.authentication.
configuration.AuthenticationConfiguration;
import
org.springframework.security.config.annotation.web.configurat
ion.EnableWebSecurity;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordE
ncoder;
import
org.springframework.security.crypto.password.PasswordEncod
er;
import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.web.authentication.Authenticatio
nSuccessHandler;
import
org.springframework.security.config.annotation.web.configurer
s.AbstractHttpConfigurer;
@Configuration
@EnableWebSecurity
public class SecurityConfig {
private final CustomUserDetailsService userDetailsService;
private final AuthenticationSuccessHandler successHandler;
public SecurityConfig(CustomUserDetailsService
userDetailsService,
AuthenticationSuccessHandler successHandler) {
this.userDetailsService = userDetailsService;
this.successHandler = successHandler;
}
@Bean
public SecurityFilterChain filterChain(HttpSecurity http)
throws Exception {
http
.csrf(AbstractHttpConfigurer::disable)
.authorizeHttpRequests(auth -> auth
.requestMatchers(
"/", "/index", "/login", "/register", "/redirect", "/about",
"/body-type", "/body-type/**", "/color-type", "/color-type/**",
"/dresses/**", "/uploads/**", "/images/**", "/css/**", "/js/**",
"/recommendation", "/api/check-recommendation", "/reset-
session-parameters",
"/api/search/**",
"/dresses/search",
"/dresses/search/history",
"/dresses/search/history/**"
).permitAll()
.requestMatchers(
"/cart", "/add-to-cart", "/cart/remove", "/cart/clear",
"/cart/update", "/api/cart/**"
).permitAll()
.requestMatchers(
"/wishlist", "/wishlist/add", "/wishlist/remove",
"/wishlist/clear",
"/wishlist/move-to-cart", "/wishlist/move-all-to-cart",
"/api/wishlist/**"
).permitAll()
.requestMatchers(
"/order/checkout", "/order/success"
).permitAll()
.requestMatchers("/admin/**").hasRole("ADMIN")
.requestMatchers("/user/**").hasRole("USER")
.anyRequest().authenticated()
)
.sessionManagement(session -> session
.sessionFixation().none()
)
.formLogin(form -> form
.loginPage("/login")
.loginProcessingUrl("/login")
.successHandler(successHandler)
.permitAll()
)
.logout(logout -> logout
.logoutUrl("/logout")
.logoutSuccessUrl("/")
.invalidateHttpSession(false)
.clearAuthentication(true)
.permitAll()
)
.userDetailsService(userDetailsService);
return http.build();
}
@Bean
public PasswordEncoder passwordEncoder() {
return new BCryptPasswordEncoder();
}
@Bean
public AuthenticationManager
authenticationManager(AuthenticationConfiguration config)
throws Exception {
return config.getAuthenticationManager();
}
}
```