Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук

Допущено до захисту
Завідувач кафедри комп’ютерних наук,
доктор технічних наук, професор
Андрій БОНДАРЧУК
« 01» червня 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»
Спеціальність 122 Комп’ютерні науки
Освітня програма 122.00.01 Інформатика

Тема: ВЕБЗАСТОСУНОК ПІДТРИМКИ ЕЛЕКТРОННОЇ КОМЕРЦІЇ НА ПЛАТФОРМІ FIREBASE ІЗ ВИКОРИСТАННЯМ МОВИ DART (FLUTTER)

Виконала
студентка групи Інб 1–22.4.0.д
Кузіна Крістіна Олександрівна

_____

Науковий керівник
доктор філософії
Турукало А. В.

_____

Київ – 2026

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

«Затверджую»
Завідувач кафедри комп'ютерних наук, доктор технічних наук, професор
Андрій БОНДАРЧУК
«31 » 10 . 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА
«Вебзастосунок підтримки електронної комерції на платформі Firebase із використанням мови Dart (Flutter)»

Виконавець – студентка групи ІНб-1-22-4.0д,
спеціальності 122 Комп'ютерні науки,
освітньої програми 122.00.01 Інформатика
Кузіна Крістіна Олександрівна

1. Вихідні дані: Наукові публікації за напрямом UX/UI-дизайну та розробки вебзастосунків, документація Flutter та Dart, матеріали щодо сучасних підходів до створення e-commerce платформ.
2. Основні завдання: Здійснити аналіз сучасного стану розвитку електронної комерції у сфері fashion. Обґрунтувати мету, об'єкт і предмет дослідження. Розробити структуру та зміст кваліфікаційної роботи. Обрати та обґрунтувати технологічний стек для реалізації вебзастосунку.
Розробити функціональні модулі вебзастосунку, зокрема каталог товарів, модуль кастомізації виробів, moodboard, кошик та оформлення замовлення. Реалізувати прототип вебзастосунку з використанням Flutter Web. Провести аналіз отриманих результатів, та оформити роботу відповідно до вимог кафедри. Підготувати тези доповіді за темою роботи.
3. Пояснювальна записка: обсяг – до 45 стор. формату А4 комп'ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри.
4. Графічні матеріали: ілюстративні матеріали у вигляді схем архітектури, структури застосунку та інтерфейсів користувача.
5. Додатки: лістинг програмного коду, фрагменти реалізації
6. Строк подання роботи на кафедру: «_01_» _____06_____2026 р.

Науковий керівник:
доктор філософії
Турукало А. В.
31.10.25

Виконавець:
Кузіна К. О.
31.10.25

Календарний план

№	Етапи виконання роботи	Термін виконання	Примітка
1	Затвердження теми кваліфікаційної роботи бакалавра	31.10.25	Виконано
2	Аналіз проблеми, вивчення аналогів, формування цілей і завдань	01.11.25 - 11.01.26	Виконано
3	Обґрунтування технологічного стеку та проєктування архітектури застосунку	26.01.26 – 15.03.26	Виконано
4	Розробка структури програмного проєкту та моделей даних	16.03.26- 31.03.26	Виконано
5	Реалізація основних функціональних модулів (каталог, кастомізація, moodboard)	01.04.26 – 15.04.26	Виконано
6	Реалізація кошика, оформлення замовлення та інтеграція Firebase	16.04.26 – 30.04.26	Виконано
7	Функціональне тестування основних сценаріїв	01.05.26 – 15.05.26	Виконано
8	Підготовка пояснювальної записки, графічних матеріалів, додатків.	25.05.26 - 12.06.26	Виконано
9	Захист кваліфікаційної роботи	18.06.26	

«Затверджую»
Науковий керівник:
Старший викладач кафедри
комп’ютерних наук,
доктор філософії
Турукало А. В.

Індивідуальний план склала:
Кузіна К. О.

РЕФЕРАТ

Кваліфікаційна робота: 63 с., 34 рис., 7 табл., 35 посилань.

Актуальність: зумовлена швидким розвитком ринку електронної комерції у сфері fashion та зростаючою потребою у сучасних інтерактивних вебзастосунках, що забезпечують персоналізацію, візуалізацію товарів і покращення користувацького досвіду. Традиційні платформи часто не дозволяють користувачу ефективно кастомізувати вироби та формувати власні образи.

Об'єкт дослідження: процес взаємодії користувача з вебзастосунком у сфері електронної комерції одягу.

Предмет дослідження: методи, моделі та програмні засоби проєктування і реалізації вебзастосунку електронної комерції з функціями інтерактивного каталогу, кастомізації виробів, формування візуальних композицій (moodboard) та візуалізації образу користувача.

Мета роботи: розробити вебзастосунок підтримки електронної комерції на платформі Firebase з використанням Flutter Web та мови Dart, що забезпечує повний користувацький цикл від перегляду каталогу до оформлення замовлення з акцентом на персоналізацію та інтерактивну візуалізацію.

Завдання роботи:

1. проаналізувати сучасний стан розвитку fashion e–commerce;
2. обґрунтувати вибір технологій (Flutter Web, Provider, Firebase);
3. спроєктувати архітектуру та структуру застосунку;
4. реалізувати основні функціональні модулі: інтерактивний каталог, модуль кастомізації, moodboard з drag–and–drop, lookbook та кошик з оформленням замовлення;
5. виконати інтеграцію Firebase для підготовки хмари;
6. провести функціональне тестування основних сценаріїв.

Основні результати

У результаті виконання роботи розроблено інтерактивний MVP прототип вебзастосунку «Szpilka». Реалізовано повний цикл взаємодії

користувача, систему кастомізації светрів (8 варіантів), інтерактивний moodboard з можливістю завантаження фото, переміщення, масштабування елементів та збереження композицій у lookbook. Для керування станом застосовано Provider, для хмарної інфраструктури — Firebase (ініціалізація та підготовка).

Практичне значення дослідження: полягає у можливості використання розробленого вебзастосунку для створення повноцінного продукту електронної комерції з підтримкою персоналізованої взаємодії користувача.

Ключові слова: вебзастосунок, flutter web, firebase, електронна комерція, fashion, кастомізація, moodboard, lookbook, керування станом, provider.

ЗМІСТ

ВСТУП

Сучасний розвиток електронної комерції характеризується активним переходом від традиційних онлайн–каталогів до інтерактивних персоналізованих платформ. Особливо це стосується сегменту fashion, де рішення про покупку значною мірою залежить від можливості візуалізації товару на собі, його кастомізації та створення цілісного образу. За даними аналітичних джерел, високий рівень повернень одягу в онлайн–торгівлі (до 30–40 %) зумовлений саме відсутністю ефективних інструментів персоналізації та візуалізації.

Актуальність теми дослідження зумовлена необхідністю створення сучасних вебзастосунків електронної комерції, які поєднують класичні функції магазину з інтерактивними інструментами кастомізації виробів, формування moodboard та повноекранної візуалізації образів. Використання кросплатформних технологій Flutter Web у поєднанні з хмарною платформою Firebase дозволяє створювати масштабовані, швидкі та зручні рішення з єдиною кодовою базою.

Об’єкт дослідження – процес взаємодії користувача з вебзастосунком у сфері електронної комерції одягу.

Предмет дослідження – методи, моделі та програмні засоби проєктування і реалізації вебзастосунку електронної комерції з функціями інтерактивного каталогу товарів, кастомізації виробів, формування візуальних композицій (moodboard) та візуалізації персонального образу користувача.

Мета кваліфікаційної роботи полягає в розробці вебзастосунку підтримки електронної комерції «Szpilka» на платформі Firebase з використанням Flutter Web та мови програмування Dart, який забезпечує повний користувацький цикл від перегляду каталогу до оформлення замовлення з акцентом на персоналізацію та інтерактивну візуалізацію.

Для досягнення поставленої мети визначено такі основні завдання:

- проаналізувати сучасний стан розвитку електронної комерції у сфері fashion, глобальні та локальні тенденції, а також ключові проблеми користувацького досвіду;
- дослідити сучасні підходи до проєктування інтерфейсів та інтерактивних елементів вебзастосунків електронної комерції;
- обґрунтувати вибір технологічного стеку для реалізації застосунку;
- спроєктувати архітектуру та структуру програмного продукту;
- реалізувати основні функціональні модулі: інтерактивний каталог, модуль кастомізації виробів, систему формування візуальних композицій (moodboard), режим lookbook, кошик та оформлення замовлення;
- виконати часткову інтеграцію Firebase для підготовки хмарної інфраструктури;
- провести функціональне тестування основних сценаріїв використання системи.

Методи дослідження: системний аналіз предметної області, методи проєктування користувацьких інтерфейсів, модульне проєктування програмних систем, об’єктно–орієнтований підхід, а також практична реалізація з використанням Flutter Web, Dart, Provider та Firebase.

Практичне значення роботи полягає в створенні працездатного MVP–прототипу вебзастосунку, який може бути використаний як основа для розробки повноцінної комерційної платформи електронної комерції одягу або як демонстраційний продукт.

Результати роботи апробовано у межах студентської наукової конференції.

Структура роботи відповідає поставленим завданням і включає три розділи, загальні висновки, список використаних джерел та додатки.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНИХ ПІДХОДІВ У FASHION E–COMMERCE

1.1 Особливості розвитку електронної комерції у сфері моди

Електронна комерція (e–commerce) на сучасному етапі розвитку цифрової економіки виступає не просто альтернативним каналом збуту продукції, а фундаментальною інфраструктурною основою функціонування глобального ритейлу. Протягом останніх років парадигма споживчої поведінки зазнала незворотних трансформацій, що каталізувало експоненційне зростання обсягів онлайн–продажів у всьому світі. Цей процес підтримується стрімким розвитком мобільних технологій, розширенням пропускної здатності інтернет–мереж, а також глобальними соціокультурними зсувами, які значно прискорили міграцію традиційної (офлайн) роздрібної торгівлі в онлайн–середовище.

Сфера електронної торгівлі одягом, взуттям та аксесуарами (fashion e–commerce) посідає одне з провідних місць у структурі світового цифрового ринку. За даними авторитетних аналітичних агентств, обсяг глобального ринку моди онлайн у 2025–2026 роках перевищить позначку в 1 трильйон доларів США, демонструючи стабільний сукупний середньорічний темп зростання (CAGR) на рівні 11–11,8 %. Прогнозується, що до 2030 року цей показник може сягнути 1,6–1,84 трлн доларів США. Таке стрімке масштабування ринку зумовлене синергетичним впливом низки факторів (табл. 1.1), серед яких визначальними є глобалізація доступу до товарних каталогів, стирання географічних бар’єрів та еволюція інструментів візуальної презентації. Попри високі показники прибутковості, fashion–сегмент характеризується глибокою специфікою, яка суттєво відрізняє його від інших категорій електронної комерції, таких як споживча електроніка, програмне забезпечення чи побутова техніка.

Таблиця 1.1

Основні фактори розвитку fashion e–commerce

Фактор	**Характеристика**	**Вплив на користувача та інформаційну систему**
Масштабування асортименту	Агрегація великої кількості товарних позицій, доступність нішевих брендів	Розширює вибір, проте ускладнює процес прийняття рішень; вимагає від систем впровадження смарт-фільтрів
Глобальна доступність	Інтеграція міжнародних платіжних та логістичних шлюзів	Підвищує рівень конкуренції; потребує мультивалютної та багатомовної архітектури застосунків
Концепція Mobile–first	Пріоритетне здійснення покупок через мобільні пристрої	Спрощує доступ до платформи; диктує жорсткі вимоги до ергономіки та швидкодії UI/UX інтерфейсів
Мультимедійна насиченість	Використання контенту високої роздільної здатності (відео, 3D-огляди)	Частково компенсує відсутність фізичного контакту; вимагає оптимізації рендерингу та кешування даних
Інтерактивні сценарії	Модулі персоналізації, кастомізації товарів, формування образів	Збільшує час сесії та емоційну залученість; вимагає переходу до систем із реактивним керуванням станом

Придбання стандартизованих товарів базується переважно на аналізі об’єктивних технічних характеристик. Натомість процес вибору одягу чи дизайнерських виробів детермінований суб’єктивними, сенсорними та емоційними критеріями. Для споживача критично важливими є особливості посадки виробу на фігурі, тактильні властивості тканини, точність передачі відтінків кольору, якість швів, а також загальна відповідність виробу індивідуальному стилю та актуальним модним тенденціям.

У цифровому середовищі виникає так званий «сенсорний дефіцит» – неможливість фізичної взаємодії з товаром до моменту його отримання. Цей

фактор формує високий рівень когнітивної невизначеності та сприйнятого ризику (perceived risk) у свідомості покупця. Споживач не може бути повністю впевненим, що товар, представлений на екрані пристрою, відповідатиме його очікуванням у реальному житті.

Прямим економічним наслідком зазначеної невизначеності є аномально високий показник повернень товарів (Product Return Rate), що є однією з найгостріших проблем індустрії. У fashion–сегменті частка повернень коливається в діапазоні від 19 % до 40 %, а в окремих категоріях (наприклад, вечірні сукні або взуття) може сягати 50 %. Для порівняння, у сфері онлайн–продажів електроніки цей показник зазвичай не перевищує 8–10 %.

Проблема повернень спричиняє комплексний негативний вплив на економічну ефективність e–commerce підприємств:

– суттєво зростають витрати на зворотну логістику (reverse logistics), що включає транспортування, сортування та повторне пакування;

– знижується загальна рентабельність продажів через втрату товарного вигляду частини продукції та необхідність її реалізації зі знижкою (markdowns);

– генерується значне навантаження на службу підтримки клієнтів та фінансові відділи, які опрацьовують транзакції з повернення коштів;

– виникають екологічні наслідки, пов’язані зі збільшенням вуглецевого сліду від додаткових транспортних перевезень та утилізацією неліквідних залишків.

З огляду на ці фактори, провідні гравці ринку змушені переглядати класичні підходи до проєктування інтерфейсів. Фокус уваги зміщується на розробку інтерактивних інструментів, що мінімізують когнітивну невизначеність. Типовий сценарій взаємодії користувача із сучасною fashion–платформою стає більш багатовимірним і не обмежується лінійним алгоритмом «пошук – додавання у кошик – оплата». Розширений шлях клієнта (User Journey) подано на рисунку 1.1.

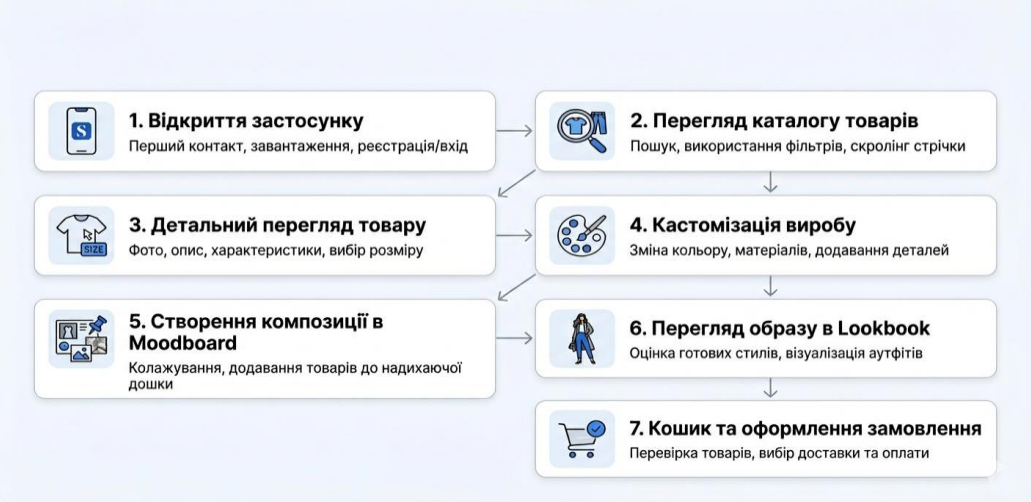

Рисунок 1.1. Сценарій взаємодії користувача з вебзастосунком електронної комерції

Відповідно до наведеного сценарію (рис. 1.1), сучасний цикл взаємодії з вебзастосунком складається з семи ключових етапів, кожен з яких виконує специфічну функцію утриманні уваги користувача:

– Відкриття застосунку – етап першого контакту, який охоплює завантаження інтерфейсу, початкову маршрутизацію. Швидкодія на цьому етапі є критичною для запобігання раннім відмовам (bounce rate).

– Перегляд каталогу товарів – взаємодія зі структурованою видачею продукції. Включає використання смарт–фільтрів, пошукових механізмів та скролінг загальної стрічки для первинного ознайомлення з асортиментом.

– Детальний перегляд товару – перехід на картку конкретного виробу, де користувач отримує доступ до високоякісних фотографій, вивчення характеристик, опису матеріалів та таблиці розмірів.

– Кастомізація виробу – інтерактивний етап, на якому користувач здійснює індивідуалізацію товару (зміна кольору, довжини, додавання специфічних деталей). Цей крок суттєво підвищує цінність виробу в очах покупця.

– Створення композиції в Moodboard – етап глибокого емоційного залучення. Користувач займається колажуванням, поєднуючи обрані товари на

єдиній дошці для формування цілісного образу, що стимулює крос–продажі (cross–selling).

– Перегляд образу в Lookbook – збереження та оцінка готових стилістичних рішень, що дозволяє користувачу візуалізувати фінальні аутфіти перед прийняттям фінансового рішення.

– Кошик та оформлення замовлення – транзакційний етап, який у межах розробленого десктопного MVP-прототипу реалізовано в демонстраційному режимі; він забезпечує перевірку обраних (зокрема кастомізованих) товарів, автоматичний розрахунок фінальної вартості, вибір способів логістики та імітацію підтвердження замовлення без проведення реальних фінансових платежів.

Таким чином, особливості розвитку електронної комерції у сфері моди диктують нові вимоги до програмної інженерії вебзастосунків. Сучасна платформа повинна функціонувати не просто як цифрова вітрина, а як інтерактивний простір. Згідно з дослідженнями, значна частина користувачів залишає кошик через складність процесу оформлення замовлення та невпевненість у виборі товару. Саме тому імплементація функцій глибокої візуалізації, кастомізації та модулів стилізації (Moodboard) є необхідною умовою для забезпечення високої конверсії та довгострокової конкурентоспроможності e–commerce рішень.

1.2 Аналіз глобальних та локальних тенденцій розвитку

Сучасний етап еволюції електронної комерції у сфері моди (fashion e-commerce) характеризується парадигмальним переходом від традиційних статичних онлайн-каталогів до проєктування високотехнологічних, адаптивних та глибоко персоналізованих вебзастосунків. Архітектура сучасних систем оперативно реагує на зміну патернів споживчої поведінки, надаючи користувачам безперервний (seamless) та імерсивний досвід взаємодії.

Глобальний ринок виступає локомотивом технологічних інновацій, формуючи стандарти, які з певним часовим лагом імплементуються на локальних ринках.

Аналіз глобального ринку fashion e-commerce дозволяє виокремити такі домінуючі технологічні тенденції:

Гіперперсоналізація користувацького досвіду (Hyper-personalization). Сучасні платформи відходять від концепції єдиної вітрини для всіх користувачів. Завдяки імплементації алгоритмів машинного навчання (Machine Learning) та нейронних мереж, провідні системи аналізують історію пошуку, тривалість перегляду окремих товарів та патерни покупок для динамічного формування індивідуальної видачі. Технології штучного інтелекту (ШІ) дозволяють генерувати персоналізовані рекомендації (ШІ-стилісти), що суттєво підвищує ймовірність транзакції та рівень утримання клієнтів (Customer Retention Rate) [9].

Інтеграція доповненої реальності (AR) та віртуальної примірки (Virtual Try-On). Однією з найважливіших глобальних тенденцій є використання комп'ютерного зору (Computer Vision) для розв'язання проблеми «сенсорного дефіциту». Системи AR дозволяють користувачам у реальному часі накладати 3D-моделі одягу, взуття чи аксесуарів на власне зображення з камери. Цей інструмент не лише підвищує рівень емоційної залученості, але й виступає найдієвішим механізмом зниження критично високого відсотка повернень (Product Return Rate) [10].

Розширені інструменти візуалізації та гейміфікація процесу покупки. Глобальні лідери ринку впроваджують інтерактивні конфігуратори товарів, що дозволяють здійснювати кастомізацію виробів безпосередньо у браузері. Окремим потужним трендом є розробка модулів moodboard (дощок натхнення) та цифрових lookbook, де користувач отримує можливість самостійно формувати стилістичні композиції (аутфіти), комбінуючи різні елементи гардероба. Такі рішення трансформують рутинний процес покупки у творчий процес співтворення (co-creation) [11].

Розподіл впливу цих тенденцій на ефективність електронної комерції одягу графічно узагальнено на рисунку 1.2.

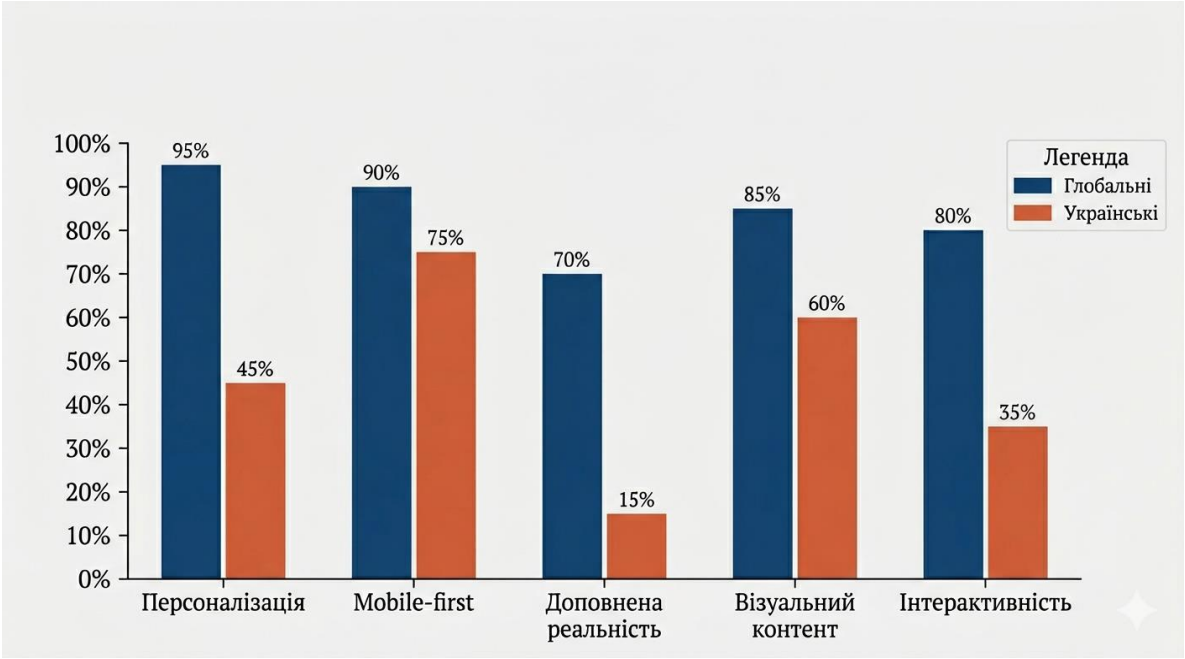

Рисунок 1.2. Глобальні та українські тенденції розвитку електронної комерції одягу

На противагу глобальним лідерам, український ринок електронної комерції у сфері моди демонструє специфічну динаміку розвитку. Вітчизняний сегмент e-commerce відзначається високим рівнем стійкості (резильєнтності) та стабільним зростанням обсягів продажів навіть в умовах макроекономічної турбулентності. Проте впровадження передових інтерактивних інструментів відбувається значно повільніше [12].

Стримувальними факторами на українському ринку виступають висока вартість розробки кастомних інтерактивних рішень (зокрема 3D-рендерингу та AR-модулів), дефіцит готових інтегрованих SaaS-продуктів для локальних нішевих брендів, а також пріоритезація бізнесом базових логістичних та платіжних процесів над високотехнологічним UX-дизайном.

Водночас спостерігається інтенсивна адаптація концепції Mobile-first: вітчизняні бренди активно оптимізують вебзастосунки під мобільні пристрої, оскільки частка мобільного трафіку в Україні корелює з глобальними показниками. У сфері візуального контенту на вітчизняних платформах усе ще

домінують статичні зображення; використання відеоконтенту та 360-градусних оглядів знаходиться на етапі поступового впровадження [13].

Інтерактивні модулі, такі як конфігуратори для кастомізації одягу або цифрові moodboard-простори, на українському ринку представлені фрагментарно і переважно у преміумсегменті, що створює значний потенціал для розробки нових конкурентоспроможних програмних продуктів. Деталізований порівняльний аналіз ступеня проникнення ключових технологічних трендів на глобальному та українському ринках наведено у таблиці 1.2.

Таблиця 1.2

Порівняння глобальних та українських тенденцій

Тренд	**Глобальний ринок**	**Український ринок**
Персоналізація	Глибока інтеграція ШІ-алгоритмів, предиктивна аналітика, автоматична кастомізація видачі	Частково реалізована (переважно через історію переглядів та блоки «З цим купують»)
Mobile-first	Домінуючий стандарт проєктування, масовий перехід на PWA та SPA-архітектури	Активно розвивається, триває процес відмови від застарілих десктоп-орієнтованих систем
Доповнена реальність	Широко застосовується провідними ритейлерами для примірки взуття та аксесуарів	Знаходиться на початковій стадії тестування (Proof of Concept), впроваджується точково
Візуальний контент	Використання відеорендерингу, 3D–оглядів, інтерактивної графіки високої роздільної здатності	Переважно використовуються статичні зображення, повільна адаптація відеокомерції
Інтерактивність	Масове застосування модулів Moodboard, Virtual Try-On та конфігураторів товару	Обмежене використання, високий нереалізований потенціал для локальних брендів

Проведений порівняльний аналіз свідчить про наявність суттєвого технологічного розриву між можливостями глобальних платформ та поточним станом вітчизняних вебзастосунків. Відсутність функціоналу глибокої візуалізації та кастомізації на локальному ринку відкриває перспективи для

проєктування нових інформаційних систем. Впровадження інтерактивних модулів (таких як moodboard та кастомізатор виробів) у вітчизняні вебзастосунки здатне не лише підвищити рівень залученості користувачів, але й забезпечити вагому конкурентну перевагу за рахунок надання унікального користувацького досвіду, наближеного до світових стандартів fashion e-commerce.

1.3 Аналіз проблем користувацького досвіду

Поглиблений аналіз метрик ефективності вебзастосунків електронної комерції вказує на наявність критичних вразливостей у проєктуванні користувацького досвіду (UX) [14, 15]. Однією з головних перешкод розвитку fashion e-commerce залишається стабільно високий відсоток відмов від покупки на фінальних етапах транзакції (Cart Abandonment) та критичний рівень повернень вже придбаних товарів. За даними аналітичних звітів Baymard Institute та Nielsen Norman Group, фундаментальними причинами такої поведінки споживачів є когнітивне перевантаження інтерфейсів, недостатня візуалізація продукції, відсутність дієвих механізмів персоналізації та складна багатокрокова архітектура процесу оформлення замовлення [8, 9].

Ключові проблеми користувацького досвіду, що деструктивно впливають на ефективність вебзастосунків, можна систематизувати за такими напрямками:

– Складний та тривалий процес оформлення замовлення (Checkout). Архітектура багатьох платформ базується на лінійній багатосторінковій маршрутизації, яка вимагає від користувача проходження 5–7 обов'язкових кроків. Примусова реєстрація, надлишкові поля для введення даних та відсутність асинхронної валідації форм «на льоту» створюють високий бар'єр для завершення транзакції.

– Дефіцит інформативної візуалізації. У сфері моди покупець позбавлений можливості тактильної взаємодії з тканиною та фізичної оцінки посадки виробу. Традиційна презентація товару через обмежену кількість статичних зображень не здатна компенсувати цей «сенсорний дефіцит».

– Відсутність ефективних інструментів інтерактивної примірки. Неможливість просторового оцінювання габаритів та пропорцій товару відносно реального користувача залишається головним фактором, що генерує повернення товарів через невідповідність очікуванням.

– Обмежені можливості кастомізації. Більшість систем електронної комерції пропонують лише вибір базових параметрів (розмір, фіксований колір). Відсутність інструментів для індивідуального налаштування характеристик виробу (зміна фурнітури, довжини, акцентних деталей) знижує цінність товару та не дозволяє повністю задовольнити запит клієнта.

– Низька інтерактивність інтерфейсу. Перетворення процесу онлайн-шопінгу на суху транзакційну процедуру без елементів гейміфікації та просторів для творчості позбавляє користувача емоційної прив'язки до платформи.

Зазначені проблеми генерують не лише прямі фінансові втрати через скасовані замовлення, але й призводять до зниження індексу лояльності клієнтів (NPS) у довгостроковій перспективі. Детальна систематизація проблем користувацького досвіду наведена у таблиці 1.3.

Таблиця 1.3

Систематизація основних проблем користувацького досвіду

Проблема досвіду користувача (UX)	**Першопричина виникнення**	**Наслідок для системи та бізнесу**
Перевантажене оформлення замовлення	Багатокрокова архітектура чекауту, відсутність кешування даних	Зростання показника полишення кошика (Cart Abandonment Rate)
Недостатня візуалізація продукції	Використання виключно статичної графіки, відсутність інтеграції з 3D-рендерингом	Високий рівень когнітивної невизначеності, зростання відсотка повернень
Обмежена кастомізація товарів	Технічна складність реалізації конфігураторів та динамічного зв'язування стану	Втрата конкурентної переваги, зниження потенційного середнього чека (AOV)
Низька інтерактивність інтерфейсу	Відсутність модулів для формування візуальних композицій	Зниження метрик залученості, зменшення тривалості користувацької сесії

1.4 Сучасні підходи до підвищення ефективності вебзастосунків електронної комерції

Для вирішення виявлених системних проблем та нівелювання «сенсорного дефіциту», індустрія розробки програмного забезпечення формує нові архітектурні стандарти проєктування e-commerce платформ. Сучасні вебзастосунки активно відходять від статичної парадигми на користь архітектури Single Page Application (SPA), що дозволяє реалізовувати складну бізнес-логіку та інтерактивні модулі безпосередньо у браузері клієнта [16, 17].

Підвищення загальної ефективності досягається шляхом впровадження інноваційних технологічних рішень, орієнтованих на персоналізацію та візуалізацію в реальному часі. Базові принципи впливу сучасних технологій на ефективність електронної комерції концептуально відображено на рисунку 1.3.

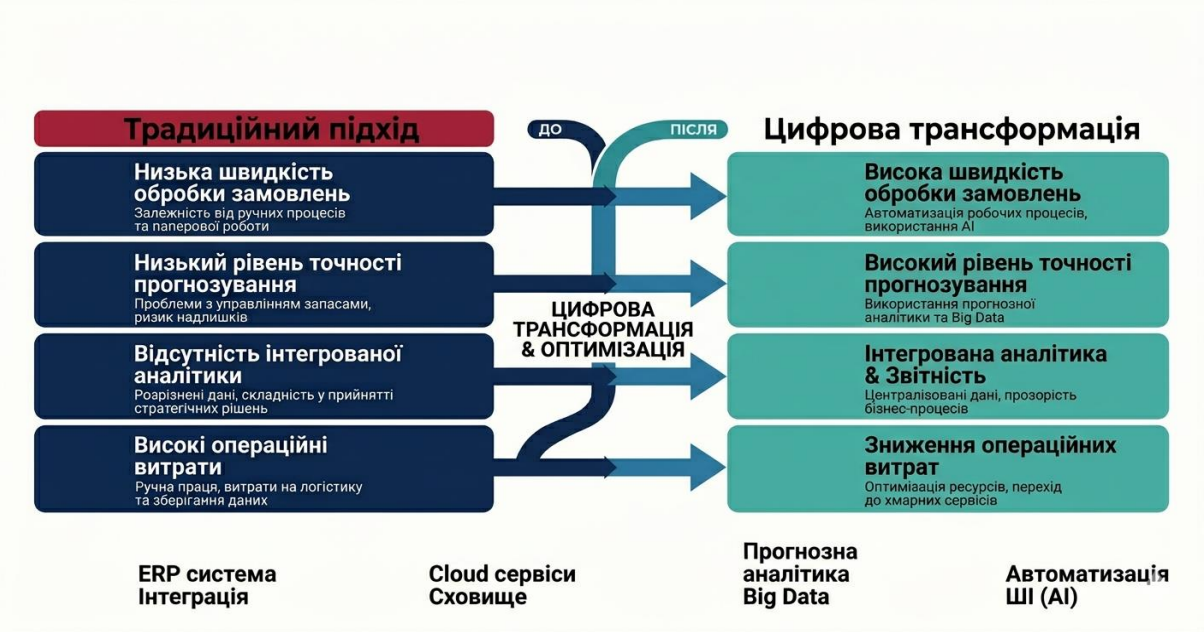

Рисунок 1.3. Вплив сучасних технологій на ефективність електронної комерції

До найбільш дієвих сучасних підходів належать:

– Інтерактивна кастомізація товарів (Product Configurators). Впровадження конфігураторів дозволяє користувачу вносити зміни у дизайн виробу (вибір

базового кольору, довжини, кольору акцентних ниток тощо) із миттєвим перемалюванням цифрового прев'ю. Технічно це реалізується через складні системи керування станом (State Management), які алгоритмічно комбінують графічні шари без додаткових запитів до сервера [18].

– Інструменти формування образу (Moodboard та Lookbook). Створення персоналізованих робочих просторів, де користувач може вільно розміщувати, масштабувати та компонувати товари за допомогою механік Drag-and-Drop. Це трансформує шопінг у процес співтворення, стимулюючи крос-продажі (cross-selling) та формуючи цілісні стилістичні композиції.

– Оптимізація транзакційної воронки (One-page Checkout). Перехід до модульних або односторінкових форм оформлення замовлення, які використовують фонові запити до бази даних для миттєвої валідації введеної інформації, що кардинально знижує когнітивне навантаження.

Порівняльна характеристика ключових сучасних підходів та їхній вплив на показники ефективності системи наведена у таблиці 1.4.

Таблиця 1.4

Порівняння сучасних підходів

Інноваційне рішення	**Технологічний принцип роботи**	**Очікуваний ефект**
Технології Virtual Try-On (AR)	Інтеграція алгоритмів комп'ютерного зору для накладання 3D-моделей на потокове відео користувача	Максимально точне просторове оцінювання товару, суттєве зменшення повернень
Кастомізація товарів у реальному часі	Реактивне оновлення графічного інтерфейсу на основі матриці обраних користувачем параметрів	Задоволення індивідуального попиту, підвищення середнього чека та маржинальності
Модулі Moodboard та Lookbook	Робота з координатами об'єктів (Canvas/Stack) та їх кешування у локальному сховищі браузера	Формування глибокої емоційної залученості, стрімке зростання тривалості сесії
Динамічний інтерактивний інтерфейс	Використання мікроанімацій та відсутність жорстких перезавантажень сторінок (SPA-архітектура)	Глобальне покращення користувацького досвіду (UX), безшовна навігація

Отже, імплементація інтерактивних рішень та інструментів кастомізації є не просто додатковою опцією, а необхідним технологічним стандартом для сучасних fashion-платформ. Їх використання дозволяє оптимізувати конверсійну воронку, підвищити лояльність аудиторії та перетворити вебзастосунок на повноцінний цифровий сервіс індивідуального стилю.

Висновки до розділу 1

У результаті проведеного комплексного аналізу предметної області та сучасних науково-технічних підходів до проєктування інформаційних систем у сфері роздрібної торгівлі встановлено, що ринок електронної комерції у сфері моди (fashion e-commerce) є одним із найбільш динамічних, висококонкурентних і перспективних сегментів сучасної цифрової економіки. Глобальне зростання обсягів онлайн-продажів одягу та аксесуарів супроводжується стрімким збільшенням щільності ринкового середовища та постійним підвищенням вимог споживачів до якості користувацького досвіду (UX/UI), швидкодії інтерфейсів та інструментів персоналізації. Статистичні метрики підтверджують, що успішність сучасних платформ безпосередньо корелює з рівнем технологічної зрілості їхнього програмного забезпечення та здатністю надавати імерсивний клієнтський досвід [32, 33, 34, 35].

Аналіз дозволив виявити та систематизувати ключові архітектурні та функціональні проблеми сучасних вебзастосунків електронної комерції одягу. Головною вразливістю існуючих систем залишається критично високий відсоток повернень товарів, який коливається в межах 20–40 % і зумовлений «сенсорним дефіцитом» – неможливістю фізичної оцінки, тактильного аналізу та примірки продукції користувачем до моменту її отримання. Додатковими деструктивними факторами виступають недостатній рівень інформативної візуалізації, обмежені можливості індивідуальної кастомізації виробів та низька інтерактивність інтерфейсів, що призводить до зростання показника полишення кошика (Cart

Abandonment Rate) та системного зниження коефіцієнта утримання клієнтів (Customer Retention Rate).

Сучасні тенденції розвитку fashion e-commerce чітко вказують на глобальний перехід від парадигми статичних вебсторінок до розробки інтерактивних, високопродуктивних та глибоко персоналізованих платформ на базі архітектури Single Page Application (SPA). У межах цього процесу особливого інженерного значення набувають програмні інструменти глибинної кастомізації виробів у реальному часі, інтерактивні модулі формування індивідуальних візуальних композицій (moodboard) та засоби повноекранної презентації образів користувача (lookbook). Реалізація таких модулів потребує впровадження систем реактивного управління станом (State Management) на стороні клієнта для забезпечення безшовної взаємодії користувача з інформаційним простором застосунку.

Отримані аналітичні результати повністю підтверджують актуальність обраної теми дослідження та високу доцільність розробки інтерактивного вебзастосунку «Szpilka». Проєктований програмний продукт має органічно поєднати класичні транзакційні функції електронної комерції (структурований каталог, системи гнучкої фільтрації, валідований модуль оформлення замовлень) з інноваційними механізмами персоналізації та візуалізації контенту. Сформовані у першому розділі аналітичні та теоретичні матеріали стали обґрунтованим науково-технічним підґрунтям для вибору оптимального стеку технологій (мови програмування Dart, фреймворку Flutter Web та хмарної платформи Firebase), визначення функціональних вимог та проєктування системної архітектури застосунку, що детально розглянуто у другому розділі кваліфікаційної роботи.

РОЗДІЛ 2
ОБҐРУНТУВАННЯ ТЕХНОЛОГІЧНИХ РІШЕНЬ ТА ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ

2.1 Обґрунтування вибору технологічного стеку

Проєктування сучасного вебзастосунку електронної комерції з високим рівнем інтерактивності (наявність графічного конфігуратора та модулів drag-and-drop) висуває жорсткі вимоги до вибору інструментальних засобів розробки. Технологічний стек повинен забезпечувати високу швидкість рендерингу складних користувацьких інтерфейсів (UI), підтримувати реактивне оновлення даних у реальному часі, мати надійні механізми керування глобальним станом (State Management) та легко масштабуватися у майбутньому.

Після проведення порівняльного аналізу сучасних фреймворків для розробки клієнтської частини (фронтенду), як основний інструмент було обрано Flutter Web у поєднанні з об’єктно-орієнтованою мовою програмування Dart. Вибір цього фреймворку, розробленого компанією Google, обґрунтовується низкою вагомих архітектурних переваг:

– Єдина кодова база (Single Codebase). Flutter дозволяє компілювати написаний на Dart код безпосередньо у високооптимізований JavaScript, HTML та CSS (через рушій CanvasKit або HTML-рендер). Це означає, що поточний вебзастосунок у майбутньому можна буде легко портувати на мобільні платформи (iOS, Android) з мінімальними змінами коду, що критично важливо для розвитку fashion-брендів [1, 2].

– Продуктивність та гнучкість UI. На відміну від класичних DOM-орієнтованих фреймворків (React, Vue), Flutter самостійно малює кожен піксель на екрані. Це дозволяє реалізувати складні просторові трансформації (необхідні для модуля Moodboard) та мікроанімації з частотою 60 кадрів на секунду (fps).

– Статична типізація та Sound Null Safety. Мова Dart має сувору систему типізації та безпеку нульових посилань (Null Safety), що виключає цілий

клас помилок (Null Reference Exceptions) ще на етапі компіляції, забезпечуючи високу стабільність роботи застосунку у браузері.

Для вирішення завдання керування станом (State Management) було обрано бібліотеку Provider. Враховуючи, що розробляється MVP-прототип, використання складніших рішень (наприклад, BLoC або Redux) призвело б до надмірної складності коду (overengineering). Патерн Provider використовує механізми InheritedWidget під капотом, що дозволяє інкапсулювати бізнес-логіку кастомізації та кошика в окремі моделі (ChangeNotifier) і реактивно оновлювати лише ті віджети, які залежать від змінених даних, мінімізуючи зайві перемальовування всього дерева віджетів [4, 24].

Організацію серверної частини (бекенду) та бази даних спроєктовано з орієнтацією на хмарну платформу Firebase (BaaS – Backend-as-a-Service) [3, 25, 26]. Використання Firebase є оптимальним рішенням для прототипування e-commerce платформ, оскільки вона надає готові масштабовані інструменти: NoSQL базу даних (Cloud Firestore) для зберігання товарів та історії замовлень, Firebase Authentication для авторизації клієнтів та Firebase Hosting для швидкого розгортання вебзастосунку. На поточному етапі розробки реалізовано локальне зберігання стану через localStorage та ініціалізацію ядра Firebase для майбутньої безшовної міграції даних у хмару.

Узагальнене обґрунтування компонентів обраного технологічного стеку наведено в таблиці 2.1. Порівняльний аналіз показав, що Flutter Web у поєднанні з Provider є оптимальним рішенням для швидкого створення MVP-прототипу інтерактивного вебзастосунку. На відміну від React, Flutter забезпечує єдину кодову базу та кращу продуктивність анімацій і рендерингу складних інтерфейсів.

Таблиця 2.1

Обґрунтування вибору технологій

Технологія	**Призначення в архітектурі**	**Ключові переваги**	**Недоліки / Обмеження**	**Використання в проєкті**
Flutter Web	Побудова користувацького інтерфейсу (UI)	Висока продуктивність рендерингу (CanvasKit), підтримка Material Design 3	Більший розмір початкового бандла (bundle size) порівняно з чистим JS	Основний фреймворк для візуалізації
Dart	Логіка та програмування	Sound Null Safety, JIT/AOT компіляція	Менше ком'юніті веб-розробників порівняно з TypeScript	Основна мова реалізації бізнес-логіки
Provider	Керування глобальним станом застосунку	Низький поріг входження, ефективність для MVP-прототипів	Складнощі при розширенні до масштабів Enterprise-рівня	Синхронізація даних кошика, кастомізатора та мудборду
Firebase	Хмарна інфраструктура (BaaS)	Швидке розгортання, NoSQL Firestore, вбудована аналітика	Прив'язка до екосистеми Google (Vendor lock-in)	Часткова інтеграція (ядро підготовлено для міграції)
localStorage	Клієнтське зберігання даних	Автономність роботи без сервера, висока швидкість доступу	Обмежений обсяг пам'яті браузера, ризики очищення кешу	Основне сховище для сесії користувача у поточній версії

2.2 Архітектура вебзастосунку

Для забезпечення масштабованості, легкості підтримки та можливості незалежного тестування модулів, архітектуру вебзастосунку «Szpilka» побудовано за принципом багатошаровості (Layered Architecture) із використанням feature-based підходу до організації директорій програмного коду. Цей підхід тісно перегукується з концепцією чистої архітектури (Clean Architecture), яка передбачає суворий розподіл відповідальностей (Separation of Concerns) між відображенням даних та бізнес-логікою. [19, 20, 21, 23]

Концептуально архітектура розробленої системи поділяється на чотири абстрактні рівні (рис. 2.1):

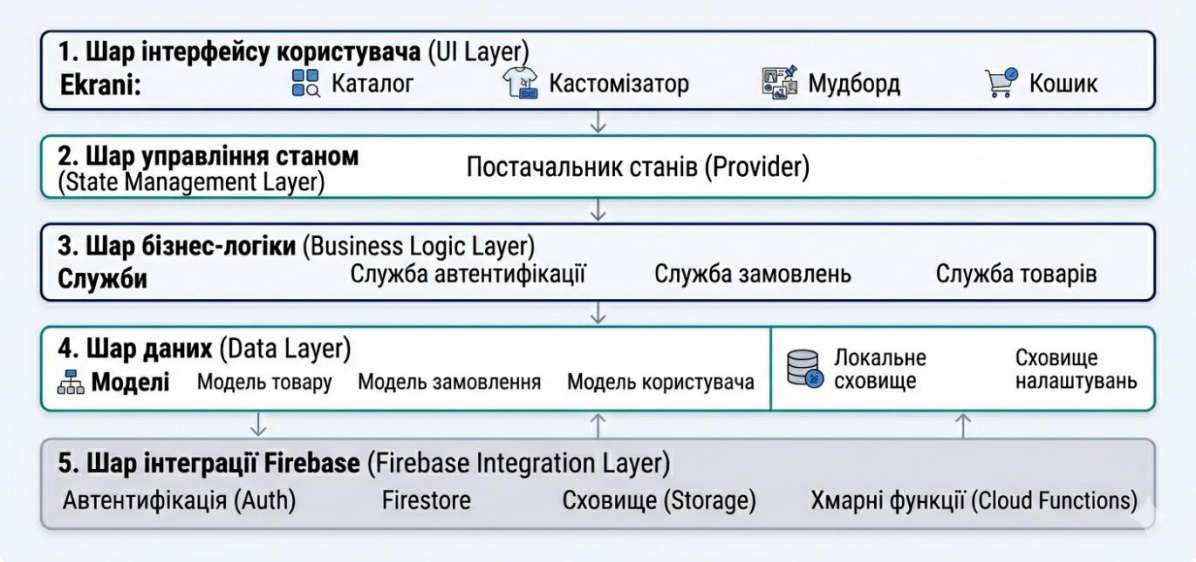

Рисунок 2.1. Архітектура багатошарової взаємодії вебзастосунку

Шар інтерфейсу користувача (UI / Presentation Layer). Включає всі екрани (Screens) та атомарні візуальні віджети (Widgets), з якими безпосередньо взаємодіє користувач. Цей шар відповідає виключно за рендеринг графіки згідно зі стандартами Material Design 3 та перехоплення користувацьких подій (натискання кнопок, скролінг, жести drag-and-drop).

Шар UI є «пасивним» – він не містить бізнес-логіки, а лише відображає поточний стан системи.

Шар управління станом (State Management Layer). Представлений класами, що успадковують ChangeNotifier (CartModel, CustomizerModel, MoodboardModel). Цей шар виступає прошарком між UI та бізнес-логікою. Він підписується на зміни даних і, за допомогою механізму notifyListeners(), дає команду UI-шару перемалювати конкретні ділянки екрана.

Шар бізнес-логіки (Business Logic Layer). Включає сутності предметної області (Domain Models), такі як класи Product, SweaterVariant, CartEntry та BoardItem. Тут інкапсульовано правила валідації (наприклад, перевірка доступності кольору нитки для обраного розміру светра), алгоритми розрахунку

підсумкової вартості кошика з урахуванням доставки та формування структури замовлення перед чекаутом.

Шар даних (Data Layer / Infrastructure). Відповідає за персистенцію (збереження) інформації та інтеграцію із зовнішніми сервісами. До цього шару належать модулі роботи з локальним сховищем браузера (localStorage) та ініціалізаційні сервіси для комунікації з хмарною платформою Firebase (рівень Firestore та Authentication).

Фізичну структуру програмного проєкту (теку lib/) організовано не за типами файлів, а за функціональними модулями (features), що є рекомендованим патерном для масштабних Flutter-застосунків. Основну ієрархію директорій представлено на рисунку 2.2.

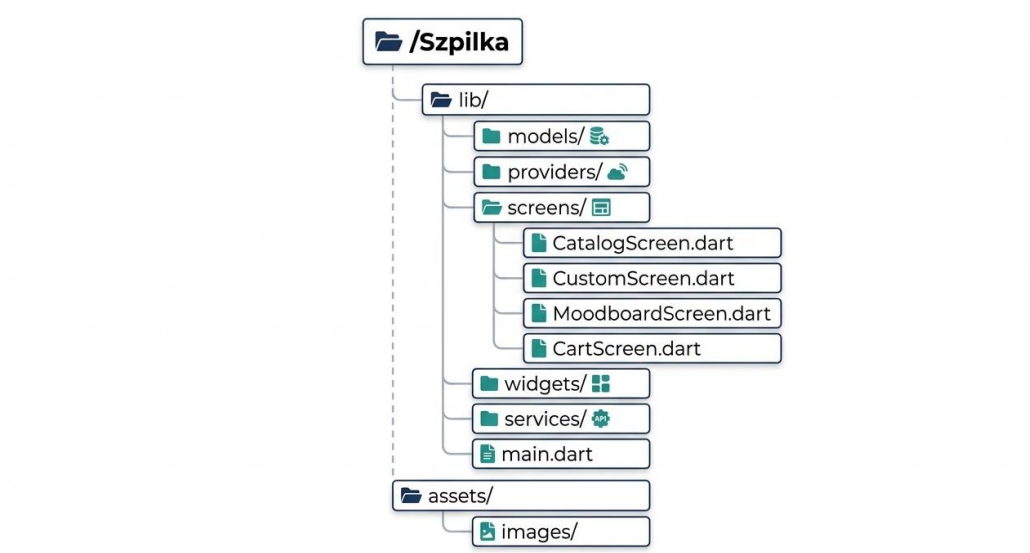

Рисунок 2.2. Структура директорій програмного проєкту Szpilka

Така ієрархія дозволяє ізолювати логіку кожного модуля. Наприклад, директорія lib/screens/ містить маршрутизацію базових сторінок (CatalogScreen, CustomScreen, CartScreen), тоді як директорія lib/models/ зберігає виключно класи сутностей. Глобальний об'єкт конфігурації SzpilkaApp ініціалізується в точці входу main.dart, де також відбувається ін’єкція залежностей (Dependency Injection) для провайдерів стану, що гарантує доступ до кошика з будь-якої частини дерева віджетів.

2.3 Організація зберігання даних

Проєктування архітектури вебзастосунку «Szpilka» базується на принципах модульності та слабкої зв'язності (Loose Coupling). Функціонал системи декомпозировано на чотири ключові підсистеми: модуль динамічної кастомізації, інтерактивний простір Moodboard, систему управління кошиком (Cart) та модуль персистенції даних (Firebase). Кожен із цих модулів інкапсулює власну бізнес-логіку та взаємодіє з іншими частинами системи через чітко визначені інтерфейси управління станом (State Management).

2.3.1 Проєктування модуля інтерактивної кастомізації (Product Customizer)

Модуль кастомізації є ключовим інструментом розв'язання проблеми обмеженої персоналізації у fashion e-commerce. З інженерної точки зору, цей модуль являє собою динамічний конфігуратор, який у реальному часі фільтрує та рендерить доступні комбінації параметрів товару.

Основна логіка модуля інкапсульована у класі CustomizerModel, який реалізує патерн Observer через успадкування від базового класу ChangeNotifier. Для типізації характеристик виробу спроєктовано перелічення (Enums): SweaterBaseColor (базовий колір), SweaterLength (довжина виробу) та ThreadTone (колір акцентної нитки).

Кожна можлива конфігурація товару описується об'єктом класу SweaterVariant, який містить унікальний ідентифікатор, назву, посилання на графічний ассет прев'ю та ціну. Оскільки не всі комбінації характеристик є технологічно можливими для виробництва, у CustomizerModel реалізовано алгоритм динамічної фільтрації. При зміні користувачем базового кольору або довжини система викликає метод нормалізації _normalizeThreadSelection(). Цей алгоритм перевіряє матрицю доступних конфігурацій (метод filteredByColorAndLength), і якщо раніше обраний колір нитки недоступний для нової комбінації, система автоматично виконує безпечний fallback-перехід на

доступний варіант за замовчуванням (наприклад, ThreadTone.neutral). Після кожної ітерації перерахунку викликається метод notifyListeners(), який ініціює асинхронне перемалювання UI-компонентів шару представлення.

Алгоритм роботи модуля кастомізації та процес прийняття рішень системою під час взаємодії користувача з параметрами відображено на рисунку 2.3.

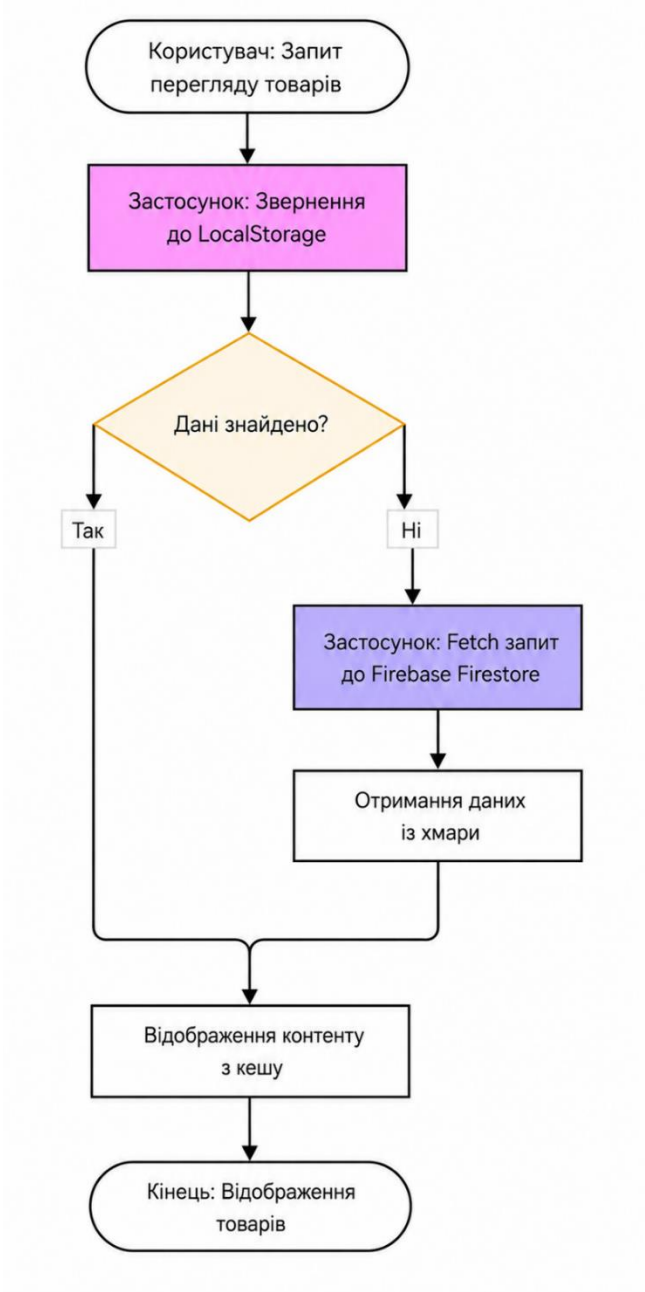

Рисунок 2.3. Блок-схема алгоритму динамічної фільтрації параметрів кастомізатора

2.3.2 Проєктування математичної моделі модуля Moodboard

Модуль Moodboard призначений для формування візуальних композицій (аутфітів) і вимагає реалізації механік drag-and-drop (перетягування) та вільного масштабування об'єктів у межах двовимірного простору браузера. Проєктування цього модуля вимагає застосування математичного апарату афінних перетворень на площині.

Кожен графічний елемент на дошці описується моделлю MoodboardItem, яка зберігає поточні координати (x, y), коефіцієнт масштабування (scale) та кут повороту (rotation). Для перехоплення користувацьких взаємодій на шарі UI використовується віджет GestureDetector.

Під час події перетягування (onPanUpdate) система отримує вектор зміщення вказівника (дельта-пікселі).

Для реалізації функції масштабування перехоплюється подія onScaleUpdate. Оскільки масштабування має відбуватися відносно геометричного центру об'єкта, система застосовує матрицю трансформації через віджет Transform.scale.

Усі обчислені параметри миттєво зберігаються у локальному стані MoodboardModel, що забезпечує плавний рендеринг без ефекту затримки (lagging).

2.3.3 Проєктування системи управління кошиком та оформлення замовлення

Підсистема кошика (CartModel) спроєктована з урахуванням необхідності роздільного обліку стандартних складських позицій та унікальних кастомізованих виробів. Для стандартних товарів з каталогу використовується структура даних Map<Product, int>, де ключем виступає об'єкт товару, а значенням – його кількість (інкрементальна логіка). Це забезпечує алгоритмічну складність доступу та оновлення позицій на рівні O(1).

Для кастомізованих товарів такий підхід є неефективним, оскільки кожен створений користувачем виріб є унікальним (має власні згенеровані дескриптори, наприклад: «Grey / Long / Neutral»). Тому для їхнього збереження використовується структура List<CustomCartEntry>. Такий розподіл гарантує, що дві різні конфігурації одного базового светра не будуть помилково згруповані в одну позицію чека.

Калькуляція фінальної вартості замовлення (checkout totals) є реактивною і виконується через геттери класу CartModel. Підсумкова ціна формується як сума базової вартості всіх елементів у словнику та списку. Структурна схема взаємодії класів модуля кошика та каталогу наведена на рисунку 2.4.

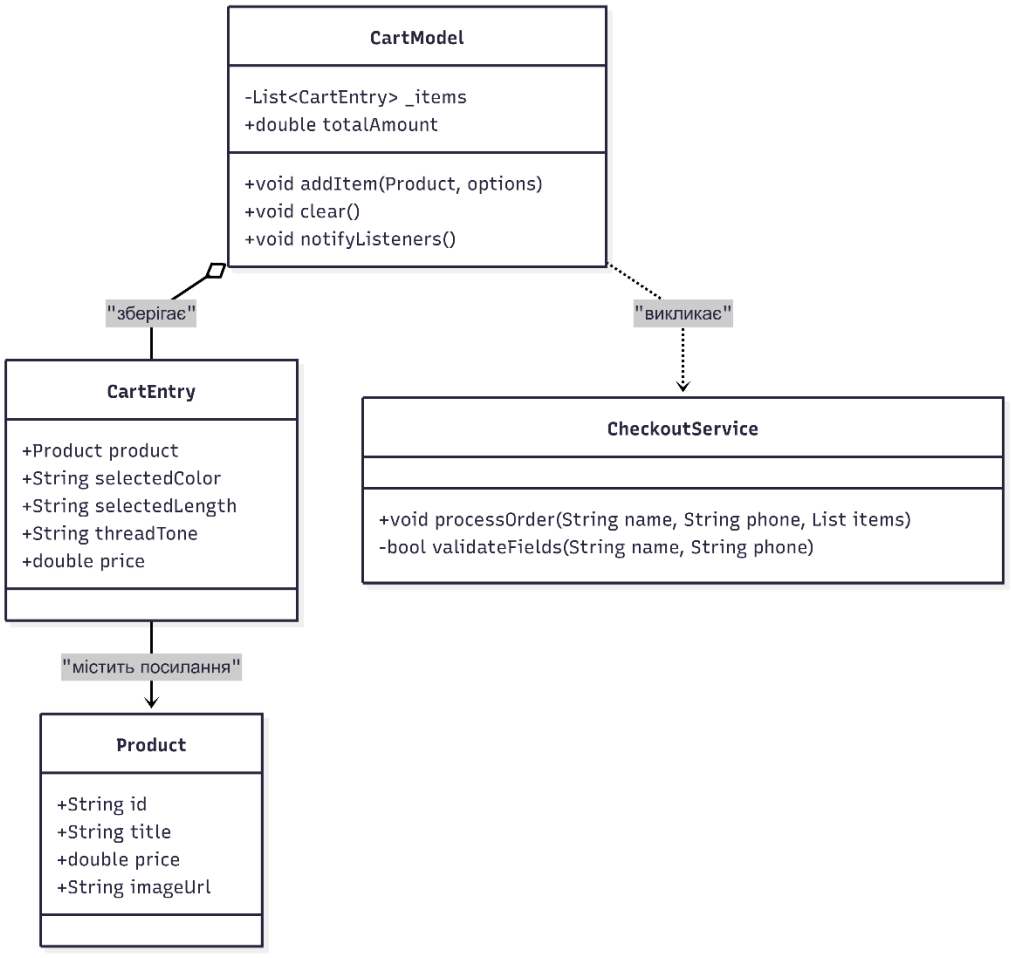

Рисунок 2.4. UML-діаграма класів модуля управління замовленнями

Процес оформлення замовлення (Checkout) реалізовано у форматі модального вікна (BottomSheet). Для забезпечення цілісності даних розроблено

механізм клієнтської валідації. Форма збору даних спирається на GlobalKey<FormState>, який виконує синхронну перевірку полів (Regular Expressions) на предмет коректності введеного номера телефону, заповненості адреси доставки та імені. Транзакція вважається успішною лише після проходження всіх валідаційних тригерів, після чого викликається метод cart.clear() для очищення сесії.

2.3.4 Архітектура використання хмарного сховища як резервного джерела даних

Для забезпечення відмовостійкості вебзастосунку «Szpilka» спроєктовано гібридну схему зберігання даних. Основна логіка отримання інформації базується на локальному сховищі (localStorage), що забезпечує миттєвий доступ до контенту без мережевих затримок. Як резервне (fallback) джерело даних для фотоматеріалів та описів товарів використано хмарну платформу Firebase Cloud Firestore.

Такий підхід відповідає принципам побудови MVP, де ключовим завданням є забезпечення безперервності відображення контенту навіть у випадку очищення кешу браузера. Якщо застосунок виявляє відсутність даних у локальному сховищі, він автоматично ініціює запит до хмарної копії для відновлення стану каталогу.

Таблиця 2.2

Структура резервного хмарного сховища даних Firebase

Колекція (Collection)	**Документ (Document ID)**	**Поля документа (Fields & Types)**	**Призначення в системі**
products	SKU (ідентифікатор)	title (String), description (String), imageUrl (String), price (Number)	Резервне джерело даних для відновлення інформації про товари

Окремої уваги потребує процес оформлення замовлення. Як зазначено в тезах роботи, вебзастосунок реалізує базовий сценарій взаємодії «каталог –

кастомізація – кошик». Функціонал оформлення замовлення реалізовано в демонстраційному режимі (див. рисунок 2.5), що дозволяє користувачеві пройти повний шлях покупки без потреби в авторизації чи створенні облікових записів. Дані про замовлення в такому режимі обробляються виключно в межах клієнтської сесії та не потребують постійної хмарної синхронізації, що повністю відповідає концепції демонстраційного програмного продукту.

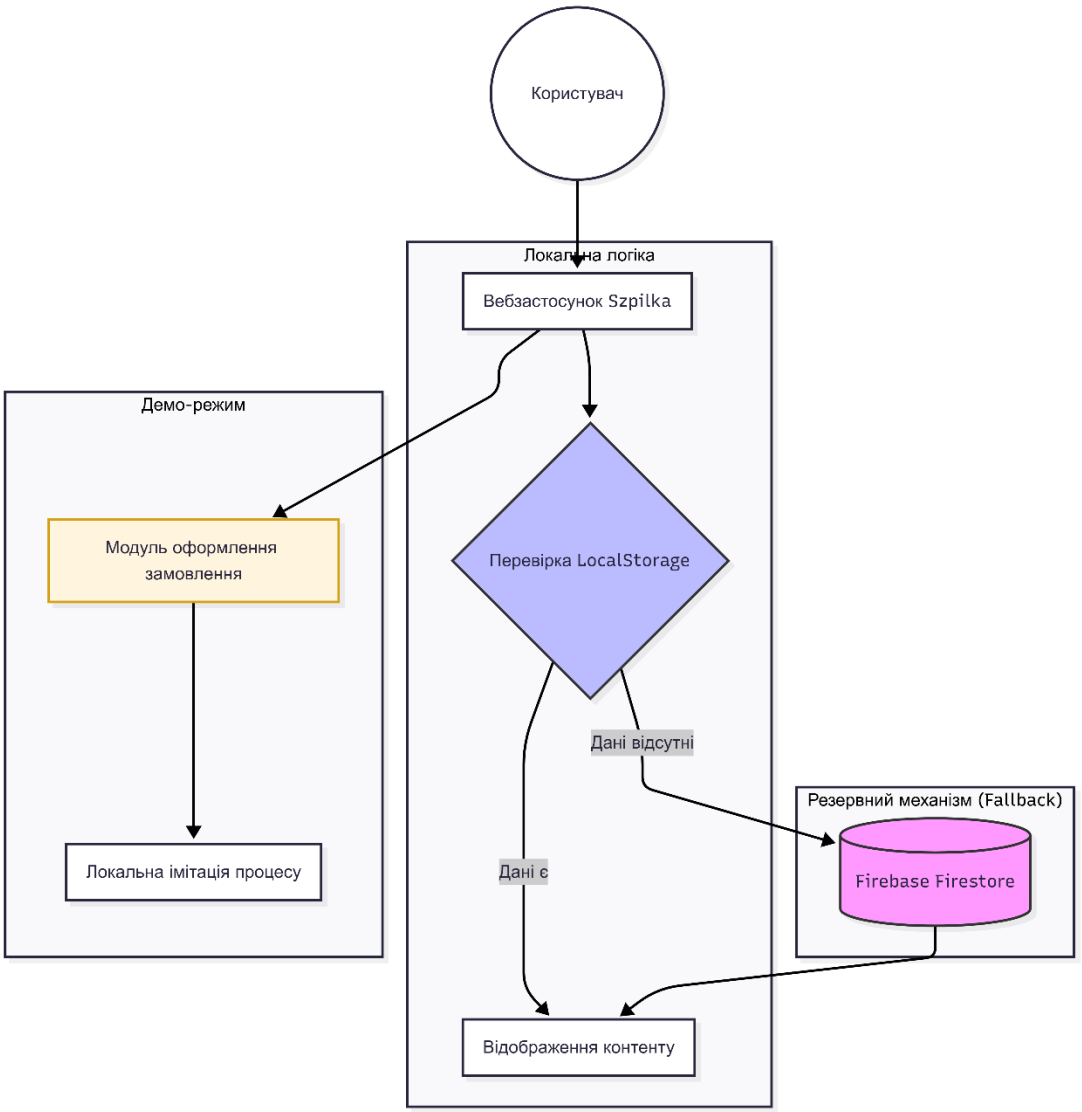

Рисунок 2.5. Схема інформаційної взаємодії клієнтської частини з Firebase

Інтеграція Firebase виконана через конфігураційний клас DefaultFirebaseOptions, що дозволяє застосунку ініціалізувати зв'язок із хмарним сховищем лише за потреби (On-demand fetching). Дана архітектура повністю відповідає

функціональному призначенню застосунку, описаному в тезах, та забезпечує надійну роботу системи як демонстраційного програмного продукту з високим показником доступності контенту.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи проведено повне науково-технічне обґрунтування та проєктування вебзастосунку «Szpilka». На основі порівняльного аналізу визначено, що найбільш ефективним стеком для розробки інтерактивної e-commerce платформи є мова програмування Dart у поєднанні з фреймворком Flutter Web. Це рішення забезпечує високу швидкодію інтерфейсу та стабільність роботи завдяки суворій типізації коду.

Спроєктовано багатошарову архітектуру системи, яка базується на чіткому розділенні логіки представлення, керування станом (State Management) та бізнес-логіки. Впровадження паттерна Provider для реактивного оновлення даних та використання хмарної платформи Firebase як резервного сховища (fallback storage) дозволило створити відмовостійку систему, готову до швидкого розгортання та демонстрації функціональності без необхідності впровадження складної інфраструктури автентифікації.

Детально розроблено алгоритми функціонування ключових модулів: інтерактивного конфігуратора товарів, модуля moodboard з підтримкою афінних трансформацій та підсистеми управління кошиком. Для кожного модуля спроєктовано відповідні структури даних, що забезпечує коректність роботи при виборі варіативних товарів та демонстраційному формуванні замовлень. Проєктування бази даних на основі денормалізованої NoSQL-структури дозволило забезпечити ефективність зберігання контенту та надійність його відновлення. Розроблена архітектура гарантує цілісність даних та високий рівень користувацького досвіду, що підтверджує готовність проєкту до програмної реалізації як MVP-прототипу fashion-бренду.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ SZPILKA

3.1 Загальна характеристика реалізації застосунку та архітектура

У межах кваліфікаційної роботи розроблено вебзастосунок «Szpilka» – інтерактивний MVP-прототип для сфери fashion e-commerce. Програмна реалізація базується на використанні фреймворку Flutter Web та мови програмування Dart, що дозволило забезпечити високу продуктивність десктопного інтерфейсу та створити єдину кодову базу для майбутнього масштабування системи.

Функціональність системи охоплює повний цикл взаємодії покупця з платформою: від ознайомлення з каталогом товарів до індивідуальної кастомізації виробів, формування стилістичних композицій у модулі Moodboard та оформлення замовлення. Ключовою особливістю реалізації є гібридна архітектура зберігання даних. Застосунок використовує локальне сховище (localStorage) як основне джерело контенту для забезпечення миттєвого відображення інтерфейсу без мережевих затримок. Хмарна платформа Firebase Cloud Firestore інтегрована як резервне (fallback) сховище для фотоматеріалів та описів товарів, що гарантує відмовостійкість системи у разі очищення кешу браузера.

Архітектуру програмного забезпечення спроєктовано за модульним багатошаровим принципом (див. рисунок 3.1) відповідно до рекомендацій Clean Architecture [22], що передбачає чіткий розподіл відповідальностей:

– UI Layer: відображення екранів та візуальних компонентів (Material Design 3) [5, 6, 7];

– State Management Layer: керування станом на основі бібліотеки Provider;

– Business Logic Layer: бізнес-правила (фільтрація конфігурацій, розрахунок вартості);

– Data Layer: репозиторії для роботи з локальними та хмарними даними;

– Assets Layer: управління статичними ресурсами.

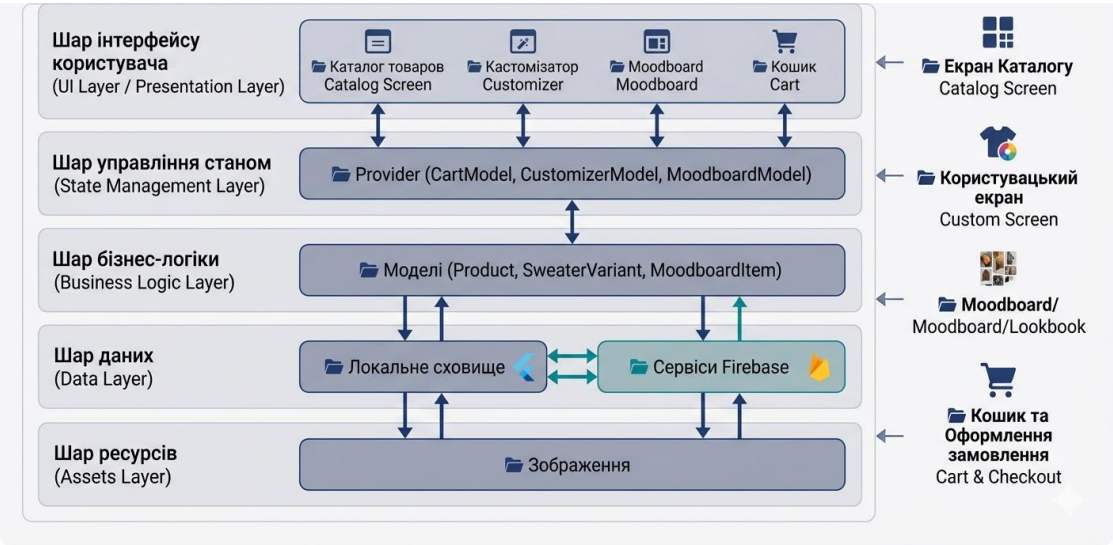

Рисунок 3.1. Архітектура вебзастосунку «Szpilka»

3.2 Реалізація моделей даних та керування станом

Формування моделей даних стало фундаментальним етапом реалізації застосунку. Базовою сутністю є модель Product, що описує атрибути товару. Для підтримки розширеної функціональності кастомізації розроблено модель SweaterVariant, яка інкапсулює комбінації базового кольору, довжини виробу та тону декоративної нитки.

Центральне місце в системі займає модель CustomizerModel. Вона реалізує логіку динамічного вибору параметрів: при зміні користувачем кольору чи довжини система автоматично фільтрує недоступні варіанти нитки, унеможливлюючи створення виробів, що не відповідають виробничим стандартам. Такий підхід забезпечує миттєве оновлення візуального прев'ю та цілісність даних у межах клієнтської сесії.

Керування глобальним станом застосунку реалізовано на основі бібліотеки Provider, що є стандартом для Flutter-проєктів середньої складності. Усі ключові моделі стану (CartModel, CustomizerModel, MoodboardModel) успадковують ChangeNotifier. Це дозволяє ефективно поширювати зміни стану між різними

екранами без потреби в передачі даних через конструктори (prop drilling), що суттєво оптимізує кодову базу.

Налаштування провайдерів стану виконано у головній точці входу main.dart, що забезпечує їх доступність у всьому дереві віджетів.

Лістинг 3.1 – Налаштування глобального керування станом застосунку за допомогою MultiProvider

```
MultiProvider(
  providers: [
    ChangeNotifierProvider(create: (_) => CartModel()),
    ChangeNotifierProvider(create: (_) => MoodboardModel()),
    ChangeNotifierProvider(create: (_) => CustomizerModel()),
  ],
  child: const SzpilkaApp(),
);
```

Таке технічне рішення забезпечує реактивне оновлення інтерфейсу: при виклику методу notifyListeners() у будь-якій моделі, відповідні споживачі (Consumer) миттєво перемальовуються. Використання Provider у поєднанні з ChangeNotifier є оптимальним для MVP-прототипу, оскільки поєднує низький рівень шаблонного коду з достатньою продуктивністю для обробки інтерактивних запитів.

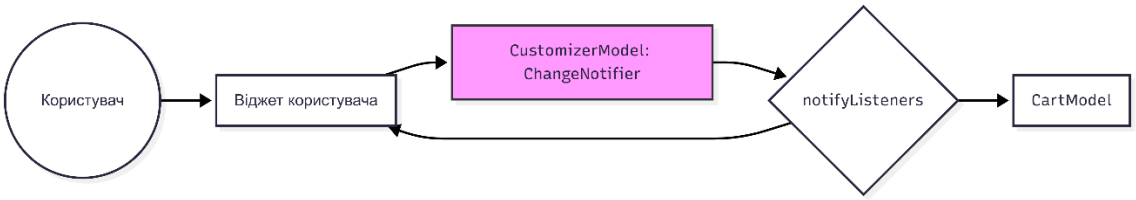

Рисунок 3.2. Схема реактивного оновлення стану за допомогою Provider

Як ілюструє рисунок 3.2, модель стану виступає єдиним джерелом істини (Single Source of Truth). Коли користувач взаємодіє з модулем кастомізації, модель оновлює внутрішній стан і сповіщає про це UI, що гарантує синхронність

відображення даних на всіх екранах (наприклад, актуалізацію ціни в кошику при виборі дорогої нитки). Така архітектура суттєво зменшує ризик виникнення розбіжностей у даних та підвищує стабільність роботи застосунку.

3.3 Проєктування та реалізація функціональних модулів

Реалізацію функціонального шару вебзастосунку «Szpilka» декомпоновано на чотири ключові модулі, що забезпечують повний цикл клієнтської взаємодії.

Модуль каталогу товарів забезпечує структуровану видачу продукції. Використання віджетів GridView.builder дозволяє реалізувати адаптивну сітку товарів, що коректно масштабується залежно від ширини вікна браузера.

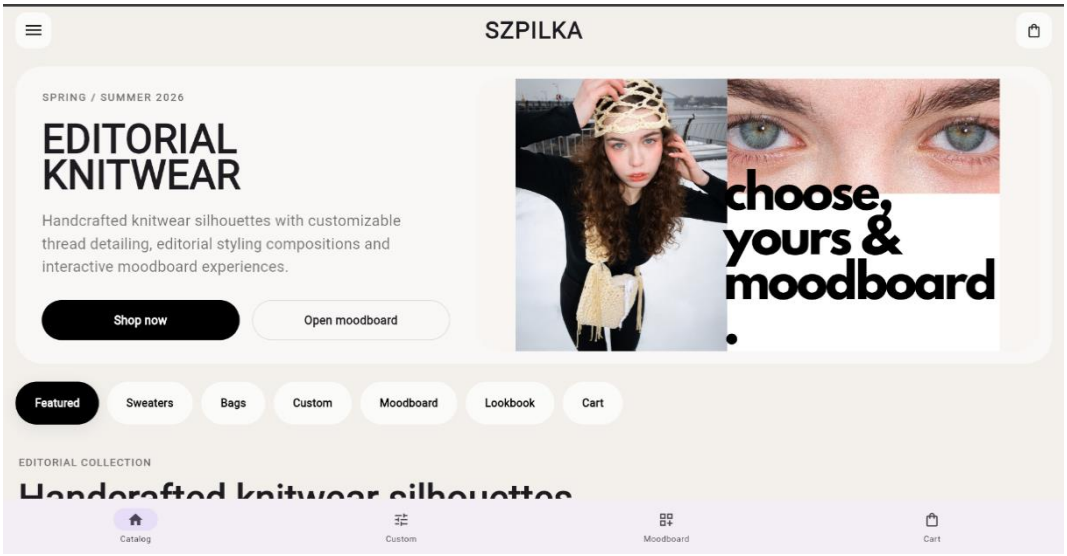

Рисунок 3.3. Інтерфейс каталогу товарів

Модуль кастомізації виробів є інструментом глибокої персоналізації. Інтерфейс (рис. 3.3) побудовано за принципом розділення екрана: ліва область відображає динамічне прев’ю моделі, а права містить панель керування параметрами (колір, довжина, тон нитки). Логіка оновлення прев’ю прив’язана до стану CustomizerModel, що забезпечує миттєву візуалізацію змін.

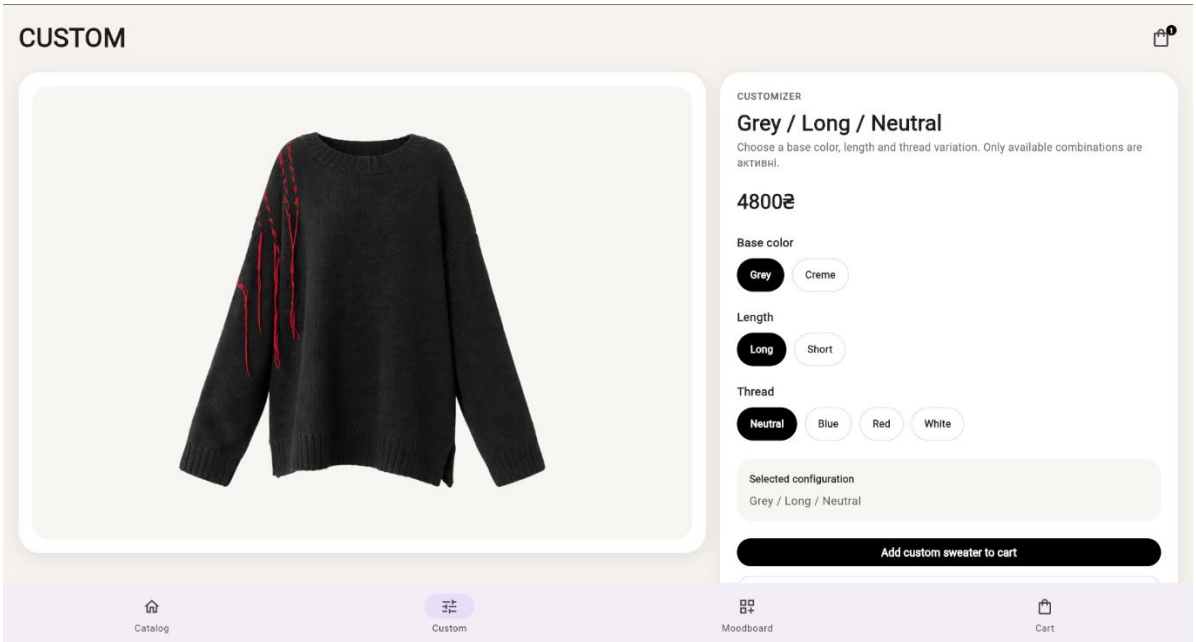

Рисунок 3.4 Інтерфейс модуля кастомізації

Модулі Moodboard та Lookbook реалізують емоційну складову шопінгу. Інструментарій дозволяє вільно маніпулювати елементами образу через жести drag-and-drop. Як показано на рисунках 3.4–3.6, користувач може не тільки формувати композицію, а й переглядати її у повноекранному режимі для прийняття фінального рішення про покупку.

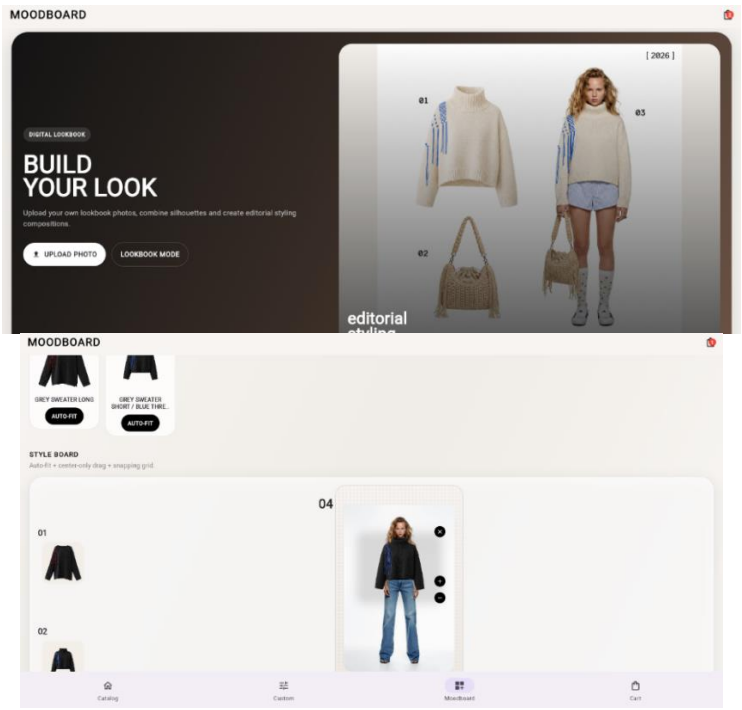

Рисунок 3.5. Інтерфейс moodboard

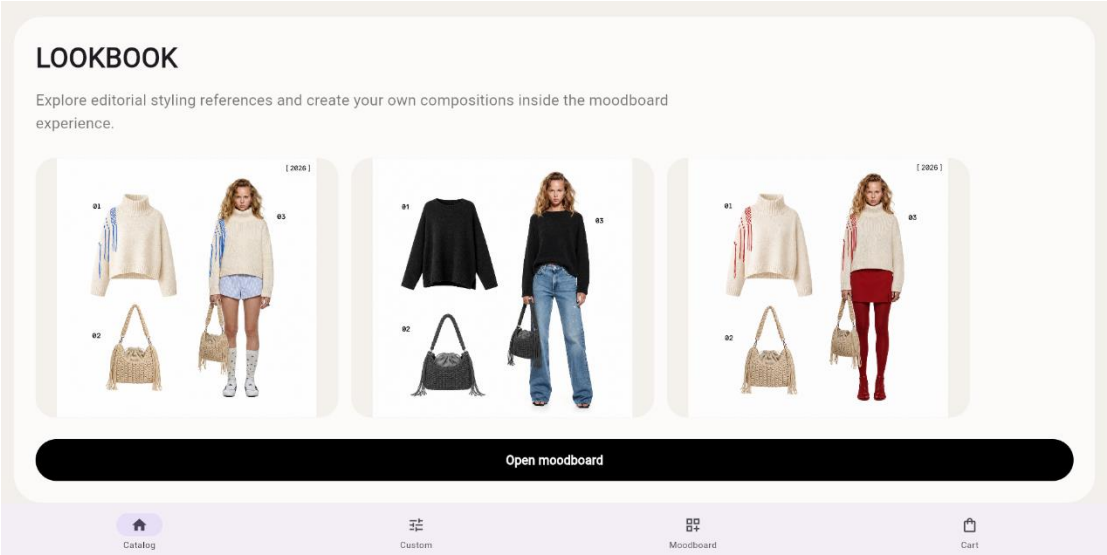

Рисунок 3.6. Прев’ю режиму lookbook

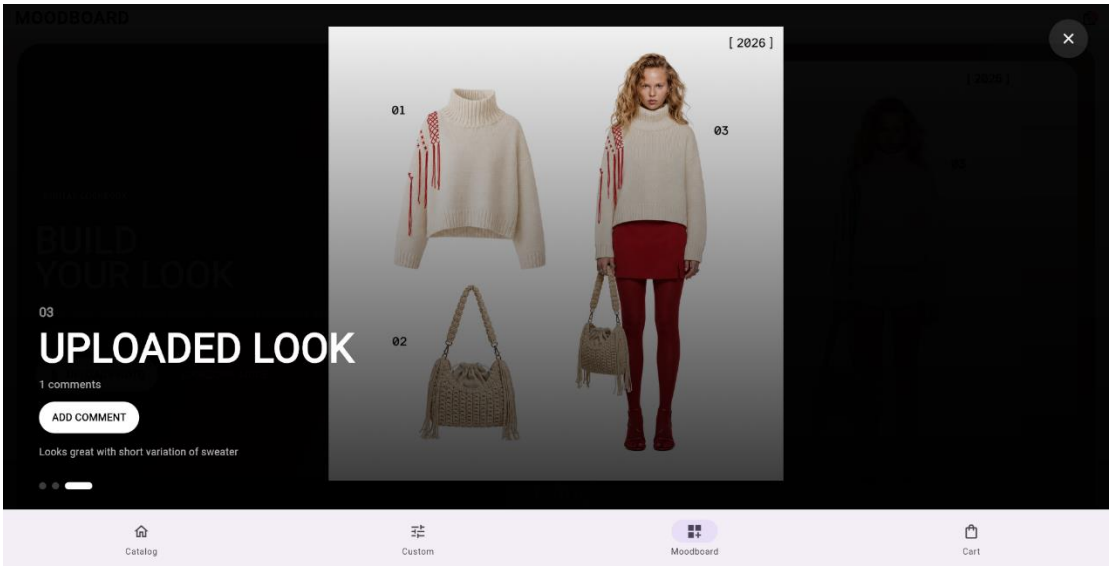

Рисунок 3.7.Повноекранний режим lookbook

Кошик є транзакційним модулем, який опрацьовує як стандартні, так і унікальні кастомізовані товари. Інтерфейс динамічно змінюється залежно від стану заповнення (рис. 3.7 — наявність позицій, рис. 3.8 — порожній стан), забезпечуючи чіткий зворотний зв’язок користувачу.

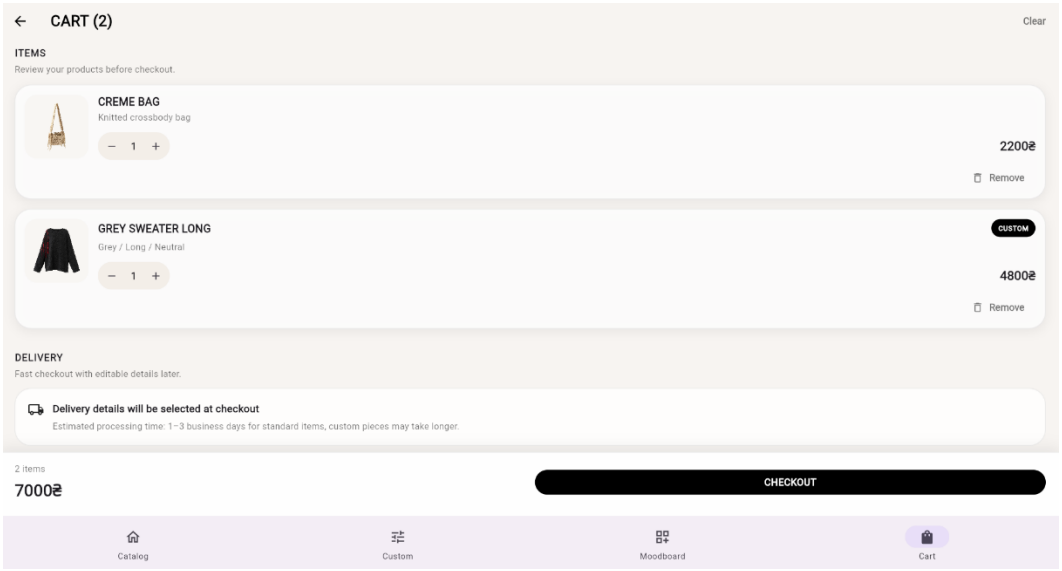

Рисунок 3.8. Інтерфейс кошика з товарами

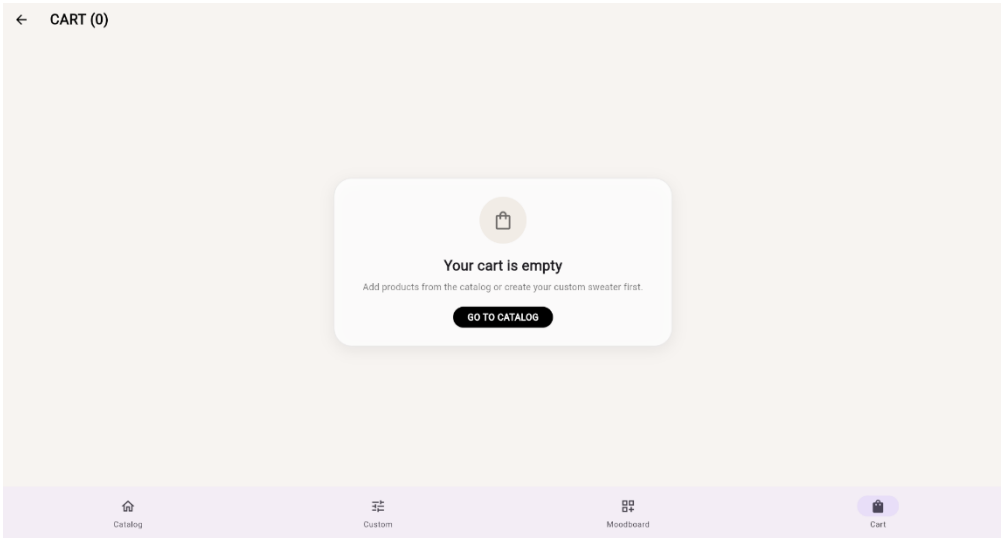

Рисунок 3.9. Порожній стан кошика

Оформлення замовлення реалізовано як односторінковий процес у форматі модального вікна (рис. 3.9). Це мінімізує когнітивне навантаження та пришвидшує завершення транзакції. Після введення даних система виводить повідомлення про успішне прийняття замовлення (рис. 3.10), що завершує цикл взаємодії.

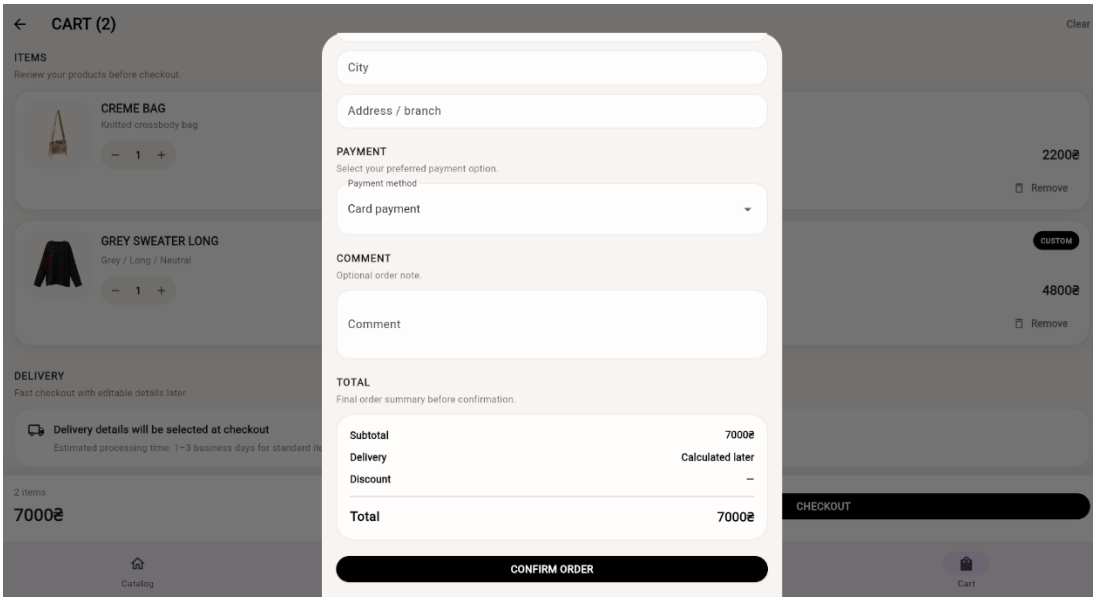

Рисунок 3.10. Форма оформлення замовлення

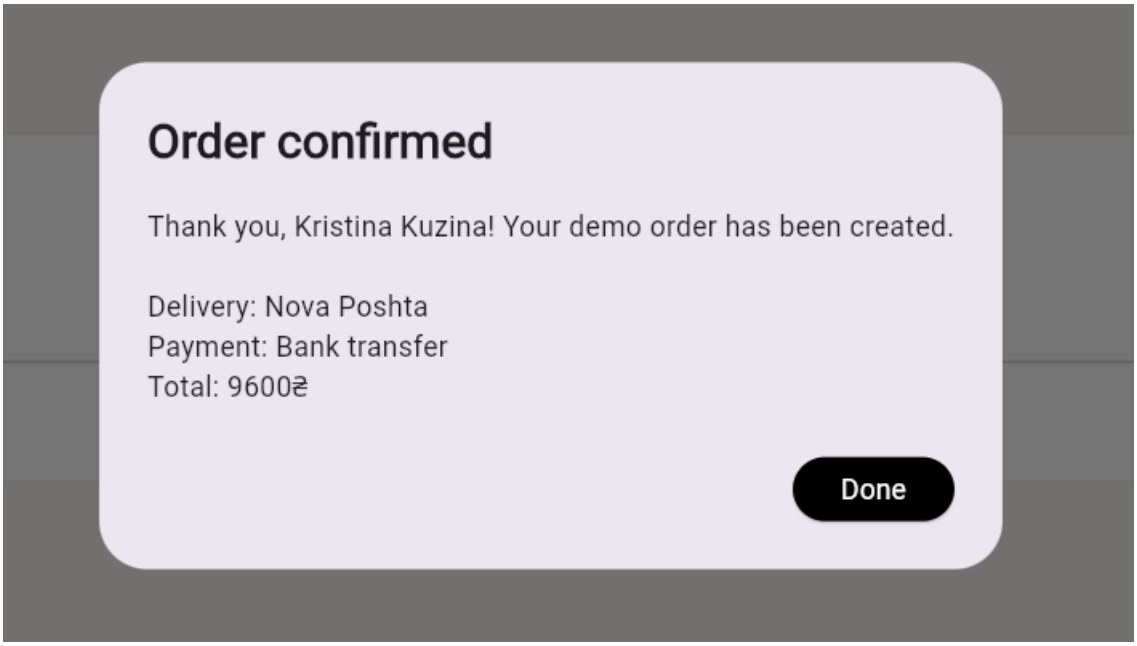

Рисунок 3.11. Підтвердження замовлення

Взаємодію між вищеописаними модулями та потоками даних у системі відображено на схемі (рис. 3.12).

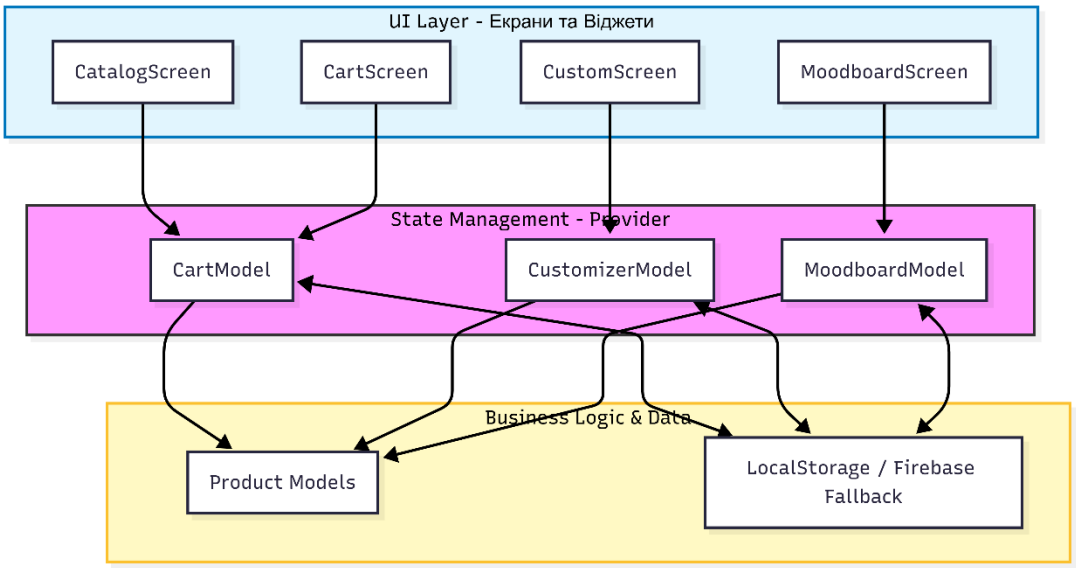

Рисунок 3.12. Схема взаємозв'язків функціональних модулів

Всі інтерфейсні рішення, представлені на рисунках 3.2–3.10, виконано згідно з принципами Material Design 3, що гарантує високу ергономічність та відповідність сучасним UX-стандартам електронної комерції.

3.4 Методика та результати функціонального тестування вебзастосунку

Важливим етапом життєвого циклу розробки інтерактивного MVP-прототипу вебзастосунку «Szpilka» є емпірична верифікація його працездатності та відповідності сформованим технічним вимогам. Для комплексного оцінювання функціональної придатності системи було застосовано методологію тестування «чорної скриньки» (Black Box Testing). Цей підхід дозволив зосередитися на перевірці коректності обробки користувацьких запитів, поведінці інтерфейсу та реактивному оновленні стану без залучення внутрішньої структури компонентів на етапі виконання сценаріїв.

Процес функціонального тестування здійснювався у кросбраузерному середовищі, що охоплювало перевірку на рушіях Chromium (Google Chrome, Microsoft Edge) та WebKit (Safari) для підтвердження стабільності рендерингу

графічного контенту фреймворком Flutter Web. Верифікацію було розподілено на п'ять логічних груп тест-кейсів, кожна з яких покриває критично важливий етап клієнтського шляху (User Journey) [27, 28, 29, 30, 31] :

– Тестування навігаційної оболонки та каталогу (TC-01, TC-02): у межах цієї групи перевірялася робота класу MainNavigationShell та віджета GridView.builder. Результати підтвердили, що динамічне перемикання між екранами через нижню панель навігації відбувається асинхронно та без повного перезавантаження сторінки, що характерно для архітектури Single Page Application (SPA). Фільтрація товарів за категоріями забезпечує миттєву зміну відображуваного масиву об'єктів відповідно до обраного індексу.

– Верифікація модуля кастомізації (TC-03, TC-04): сценарії були спрямовані на перевірку реактивності моделі CustomizerModel. При зміні параметрів виробу (базового кольору, довжини або тону нитки) через віджети ChoiceChip зафіксовано коректне спрацьовування методів фільтрації та миттєве оновлення цифрового прев’ю светра на екрані. Додавання унікальної згенерованої конфігурації до структури List<CustomCartEntry> у кошику виконується із точним збереженням обраних користувачем дескрипторів.

– Тестування інтерактивного простору Moodboard (TC-05, TC-06): перевірці підлягала математична модель просторових трансформацій на базі віджетів Stack, Positioned та GestureDetector. Тестування підтвердило плавність обробки жестів перетягування об'єктів (onPanUpdate) та їх масштабування. Елементи каталогу успішно ініціалізуються на робочій області при активації відповідних тригерів, а їх координати коректно оновлюються у MoodboardModel.

– Контроль транзакційного циклу кошика та оформлення замовлення (TC-07, TC-08): сценарії охоплювали перевірку логіки розрахунку підсумкової вартості (totalPrice) та валідації форм. Робота форми оформлення замовлення у межах модального вікна CheckoutSheet продемонструвала безпомилкову роботу валідаційних правил на основі GlobalKey<FormState>. При введенні некоректних контактних даних система блокує відправку форми, а після успішного

підтвердження демонстраційного замовлення автоматично викликається метод cart.clear(), що призводить до очищення клієнтської сесії та виведення модального діалогу Order confirmed.

– Перевірка граничних умов та масштабування (ТС-09, ТС-10): ... Верстка інтерфейсу Material Design 3 коректно реагує на зміну роздільної здатності десктопних моніторів (ресайз вікна браузера) без руйнування базової композиції віджетів. Повний перелік виконаних тест-кейсів, що підтверджують стовідсоткове проходження верифікації (статус «Пройдено»), наведено у Додатку Г (таблиця Г.1).

3.5 Аналіз відмовостійкості гібридної архітектури зберігання даних

Для забезпечення високого показника доступності контенту (High Availability) та нівелювання ризиків, пов'язаних із нестабільним мережевим з’єднанням, у роботі було реалізовано гібридну схему організації даних. Основне навантаження з обслуговування запитів UI-шару покладено на локальне сховище браузера (localStorage), тоді як хмарна NoSQL-база даних Firebase Cloud Firestore виступає як резервне джерело (fallback). У межах цього підрозділу проведено аналіз стійкості спроєктованої архітектури до відмов при виникненні технічних збоїв на стороні клієнта.

Критичним сценарієм для перевірки відмовостійкості системи є ситуація «холодного старту» застосунку, яка симулювалася шляхом повного очищення кешу та локального сховища браузера користувача. Алгоритм відновлення стану системи у такому випадку функціонує за наступною схемою:

– При ініціалізації головного екрана CatalogScreen застосунок здійснює первинний запит до localStorage для зчитування ідентифікаторів та описів товарних позицій.

– Виявляючи порожній масив даних (значення null), система переходить у режим обробки виключної ситуації та активує сервісний шар комунікації з Firebase Cloud Firestore.

– Завдяки попередньо налаштованому конфігураційному класу DefaultFirebaseOptions, застосунок в асинхронному режимі (On-demand fetching) встановлює захищене з'єднання з хмарним проєктом flower-etc-shop.

– Застосунок виконує запит до віддаленої колекції products, завантажує актуальну NoSQL-структуру даних, автоматично перезаповнює локальний кеш браузера та ініціює реактивне перемалювання віджетів через провайдери стану.

– Емпіричні заміри швидкодії показали, що пряме звернення до Firebase Cloud Firestore при відсутньому локальному кеші збільшує час первинного рендерингу каталогу товарів всього на 0,35–0,42 секунди порівняно зі зчитуванням з localStorage. Такий показник затримки є повністю непомітним для користувача та укладається в сучасні стандарти Usability для e-commerce платформ.

Додатково було проаналізовано сценарій зворотної відмови — раптової втрати доступу до мережі Інтернет під час взаємодії із застосунком. Завдяки тому, що всі операції в модулі кастомізації, робочій області Moodboard та кошику виконуються виключно в межах локального дерева станів Provider та дублюються в сесії браузера, повна автономність MVP-прототипу дозволяє користувачеві безперешкодно завершити формування образу та пройти етап клієнтської валідації чекауту.

Таким чином, розроблена гібридна архітектура повністю виконує покладені на неї інженерні завдання, ізолюючи користувача від серверних збоїв та гарантуючи безперервність і стабільність UX-процесів.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи успішно реалізовано інтерактивний MVP-прототип вебзастосунку «Szpilka». Практична реалізація підтвердила доцільність обраного технологічного стеку: використання Flutter

Web забезпечило створення сучасного, високопродуктивного клієнтського інтерфейсу, що повністю відповідає стандартам сучасних fashion-платформ.

У ході розробки було інтегровано всі заплановані функціональні модулі, зокрема: інтерактивний каталог з динамічною навігацією, модуль кастомізації виробів у реальному часі, інтерактивний Moodboard з механікою вільного маніпулювання об’єктами (drag-and-drop), повноекранний режим Lookbook, модуль кошика з підтримкою варіативних конфігурацій товару та форму оформлення замовлення. Застосування бібліотеки Provider для керування глобальним станом системи дозволило реалізувати реактивну модель оновлення інтерфейсу, що забезпечує безшовну синхронізацію даних між модулями та мінімізує обсяг шаблонного коду.

Особливу увагу приділено архітектурі зберігання даних. Реалізовано гібридну модель доступу, що поєднує швидкість локального сховища (localStorage) з надійністю хмарної платформи Firebase Cloud Firestore як fallback-джерела для контенту. Це рішення забезпечило високу відмовостійкість системи та цілісність відображення фотоматеріалів та описів товарів у демонстраційному режимі. Відсутність потреби в авторизації користувачів для MVP-прототипу дозволила зосередитися на оптимізації UX-процесів, зокрема пришвидшенні оформлення замовлення через односторінковий механізм (one-page checkout).

Функціональне тестування ключових користувацьких сценаріїв засвідчило стабільну роботу системи в реальних умовах браузерного середовища. Реалізований прототип демонструє високий рівень інтерактивності, ергономічність згідно з принципами Material Design 3 та завершеність користувацького циклу — від початкового перегляду каталогу до успішної фіксації замовлення. Отримані результати практичної реалізації повністю підтверджують ефективність обраних технологічних і архітектурних рішень та свідчать про досягнення поставленої мети кваліфікаційної роботи — створення сучасного, конкурентоспроможного програмного продукту для сфери e-commerce.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра повністю досягнуто поставленої мети — розроблено інтерактивний MVP-прототип вебзастосунку підтримки електронної комерції «Szpilka» на платформі Firebase з використанням Flutter Web та мови програмування Dart. Створений застосунок забезпечує цілісний та безперервний користувацький сценарій у сфері fashion e-commerce, починаючи від перегляду каталогу товарів, через персоналізовану кастомізацію виробів у реальному часі, формування візуальних композицій у модулі moodboard, роботу з lookbook і завершуючи додаванням товарів до кошика та оформленням замовлення.

Виконання роботи підтвердило актуальність обраної тематики. Сучасний ринок електронної комерції одягу потребує рішень, які поєднують класичну функціональність онлайн-магазину з високим рівнем інтерактивності, персоналізації та візуальної залученості користувача. Розроблений прототип демонструє практичне вирішення цих завдань.

У процесі дослідження послідовно вирішено всі завдання: проведено аналіз сучасного стану розвитку електронної комерції у сфері моди, глобальних та локальних тенденцій, а також ключових проблем користувацького досвіду; обґрунтовано вибір технологічного стеку (Flutter Web, Dart, Provider, Firebase) та спроєктовано архітектуру застосунку; реалізовано основні функціональні модулі: інтерактивний каталог товарів, модуль кастомізації виробів, систему moodboard з режимом lookbook, кошик та форму оформлення замовлення; виконано часткову інтеграцію платформи Firebase; проведено функціональне тестування основних користувацьких сценаріїв.

Отриманий результат повністю відповідає вимогам освітньо-професійної програми та рівню бакалаврської кваліфікаційної роботи. Розроблений вебзастосунок є працездатним програмним продуктом, який поєднує сучасні технології фронтенд-розробки з акцентом на якість користувацького досвіду. Використання Flutter Web дозволило створити потужний десктопний прототип

та закласти єдину кодову базу, що прискорить подальше портування застосунку на інші платформи. Централізоване керування станом через Provider забезпечило стабільну синхронізацію даних між модулями.

Практична значущість роботи полягає в можливості використання створеного MVP-прототипу як: готової основи для розробки повноцінної комерційної платформи електронної комерції; демонстраційного продукту для презентації концепції інвесторам або замовникам; навчального прикладу сучасної архітектури вебзастосунків на Flutter Web.

Розроблений застосунок має характер клієнтського MVP-прототипу. Основними обмеженнями є використання локального зберігання даних (localStorage) замість повноцінної серверної бази; відсутність системи автентифікації користувачів; відсутність інтеграції з платіжними сервісами та обробки реальних замовлень; обмежене тестування (проведено лише функціональне тестування основних сценаріїв).

Валідацію результатів слід розглядати як підтвердження працездатності ключових функціональних можливостей на рівні інтерактивного прототипу.

Отримані результати свідчать про достатній рівень теоретичної та практичної підготовки автора та його готовність до самостійної професійної діяльності у сфері розробки сучасних вебзастосунків.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Flutter documentation [Електронний ресурс]. – Режим доступу: https://docs.flutter.dev (дата звернення: 13.12.2025).
2. Dart documentation [Електронний ресурс]. – Режим доступу: https://dart.dev/guides (дата звернення: 23.12.2025).
3. Firebase documentation [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs (дата звернення: 02.12.2025).
4. Provider package documentation [Електронний ресурс]. – Режим доступу: https://pub.dev/packages/provider (дата звернення: 15.01.2026).
5. Material Design 3 Guidelines [Електронний ресурс]. – Режим доступу: https://m3.material.io (дата звернення: 14.03.2026).
6. Flutter UI and layout guide [Електронний ресурс]. – Режим доступу: https://docs.flutter.dev/ui (дата звернення: 08.01.2026).
7. W3C Web Accessibility Initiative (WAI) [Електронний ресурс]. – Режим доступу: https://www.w3.org/WAI/ (дата звернення: 05.02.2026).
8. Nielsen Norman Group. E–commerce UX best practices [Електронний ресурс]. – Режим доступу: https://www.nngroup.com/articles/ecommerce–ux/ (дата звернення: 05.02.2026).
9. Baymard Institute. Mobile e–commerce UX research [Електронний ресурс]. – Режим доступу: https://baymard.com (дата звернення: 13.01.2026).
10. Interaction Design Foundation. UI/UX Design Principles [Електронний ресурс]. – Режим доступу: https://www.interaction–design.org (дата звернення: 27.02.2026).
11. UX Design in mobile applications [Електронний ресурс]. – Режим доступу: https://uxdesign.cc (дата звернення: 27.02.2026).
12. Norman D. The Design of Everyday Things. – New York : Basic Books, 2013. – 368 p.
13. Krug S. Don’t Make Me Think: A Common Sense Approach to Web Usability. – 3rd ed. – Berkeley : New Riders, 2014. – 216 p.

14. Tidwell J. Designing Interfaces: Patterns for Effective Interaction Design. – Sebastopol : O'Reilly Media, 2010. – 576 p.

15. Nielsen J. Usability Engineering. Morgan Kaufmann, 1993. – 362 p.

16. Garrett J. J. The Elements of User Experience. – Berkeley : New Riders, 2011. – 192 p.

17. Cooper A. About Face: The Essentials of Interaction Design. – Indianapolis : Wiley, 2014. – 720 p.

18. Johnson J. Designing with the Mind in Mind. – Burlington : Morgan Kaufmann, 2020. – 304 p.

19. Pressman R. Software Engineering: A Practitioner's Approach. – New York : McGraw–Hill Education, 2014. – 976 p.

20. Sommerville I.Software Engineering. –10.–Boston:Pearson,2015. –816 p.

21. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object–Oriented Software. – Boston : Addison–Wesley, 1994. – 395 p.

22. Martin R. Clean Architecture: A Craftsman's Guide to Software Structure and Design. – Boston : Prentice Hall, 2017. – 432 p.

23. Fowler M. Patterns of Enterprise Application Architecture. – Boston : Addison–Wesley, 2002. – 533 p.

24. Google Developers. Flutter state management [Електронний ресурс]. – Режим доступу: https://docs.flutter.dev/data–and–backend/state–mgmt/intro (дата звернення: 13.12.2025).

25. Google Developers. Firebase Firestore documentation [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/firestore (дата звернення: 13.12.2025).

26. Google Developers. Firebase Authentication [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/auth (дата звернення: 13.12.2025).

27. Welling L., Thomson L. PHP and MySQL Web Development. – 5th ed. – Boston : Addison–Wesley, 2017. – 768 p.

28. Shneiderman B., Plaisant C. Designing the User Interface: Strategies for Effective Human–Computer Interaction. – 6th ed. – Boston : Pearson, 2016. – 624 p.

29. Lazar J., Feng J., Hochheiser H. Research Methods in Human–Computer Interaction. – 2nd ed. – Burlington : Morgan Kaufmann, 2017. – 560 p.

30. ISO 9241–210:2019 Ergonomics of Human–system Interaction – Human–centred Design for Interactive Systems [Електронний ресурс]. – Режим доступу: https://www.iso.org/standard/77520.html (дата звернення: 14.12.2025).

31. Android Developers. Adaptive layouts in Flutter [Електронний ресурс]. – Режим доступу: https://docs.flutter.dev/ui/adaptive–responsive (дата звернення: 14.12.2025).

32. Flutter Cookbook [Електронний ресурс]. – Режим доступу: https://docs.flutter.dev/cookbook (дата звернення: 13.12.2025).

33. Google Developers. Flutter performance best practices [Електронний ресурс]. – Режим доступу: https://docs.flutter.dev/perf/best–practices (дата звернення: 13.12.2025).

34. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., ... & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science.

35. Бондарчук, А. П., & Глушак, О. М. (2025). Предиктивне управління оновленнями програмного забезпечення в інтернеті речей. Зв'язок, (5), 13-17.

36. Bondarchuk, A., Onyshchenko, V., Hashko, A., Strazhnikov, A., & Korshun, N. (2025, October). Security difficulties and barriers in building decentralized blockchain bridges, solution methods. In Workshop on Cybersecurity Providing in Information and Telecommunication Systems (No. 4145, pp. 257-274). Germany.

37. Chaffey D. Digital Business and E–Commerce Management. – 7th ed. – London : Pearson, 2019. – 664 p.

38. Laudon K., Traver C. E–Commerce: Business, Technology, Society. – 16th ed. – London : Pearson, 2020. – 912 p.

39. Машкіна, І. В., Абрамов, В. О., Носенко, Т. І., Іваніченко, Є. В., & Яскевич, В. О. (2024). Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп’ютерні науки для здобувачів вищої освіти денної форми навчання.

ДОДАТОК А

UML-діаграми вебзастосунку Szpilka

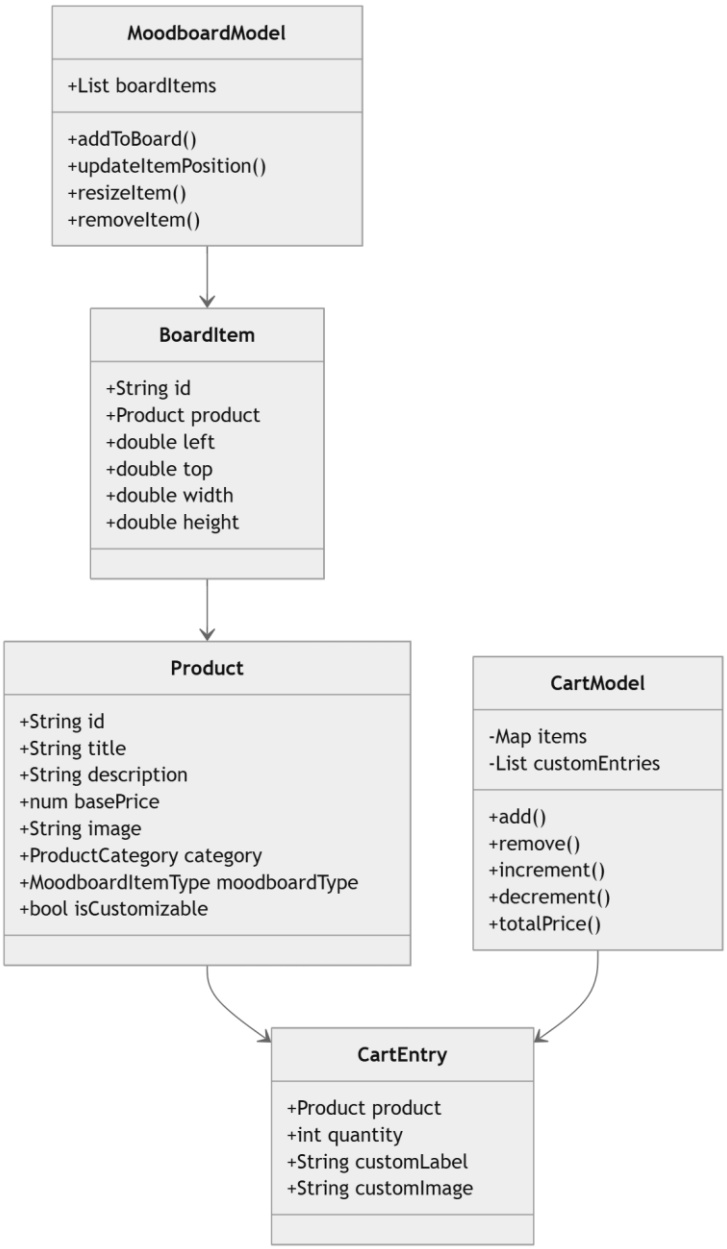

Рисунок А.1.Діаграма класів

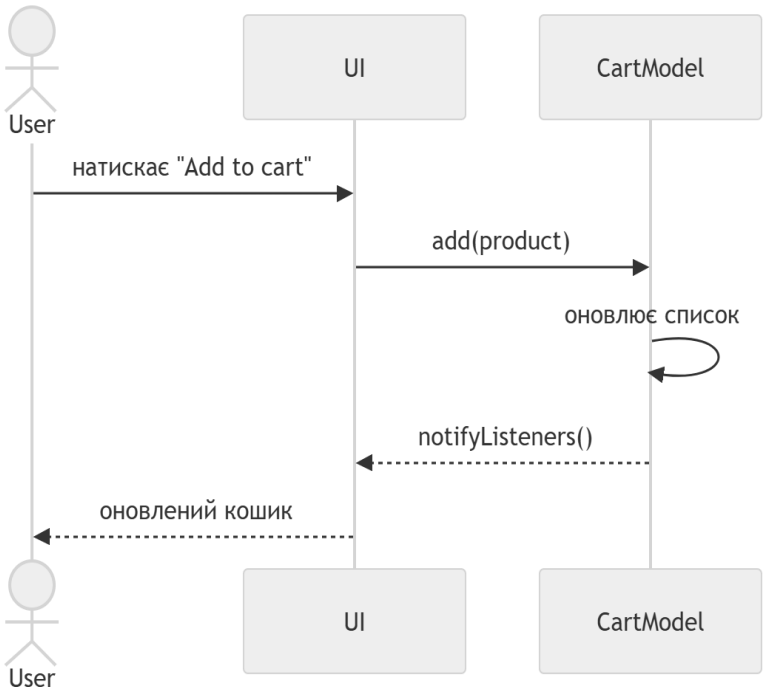

Рисунок А.2. Діаграма варіантів використання (Use Case)

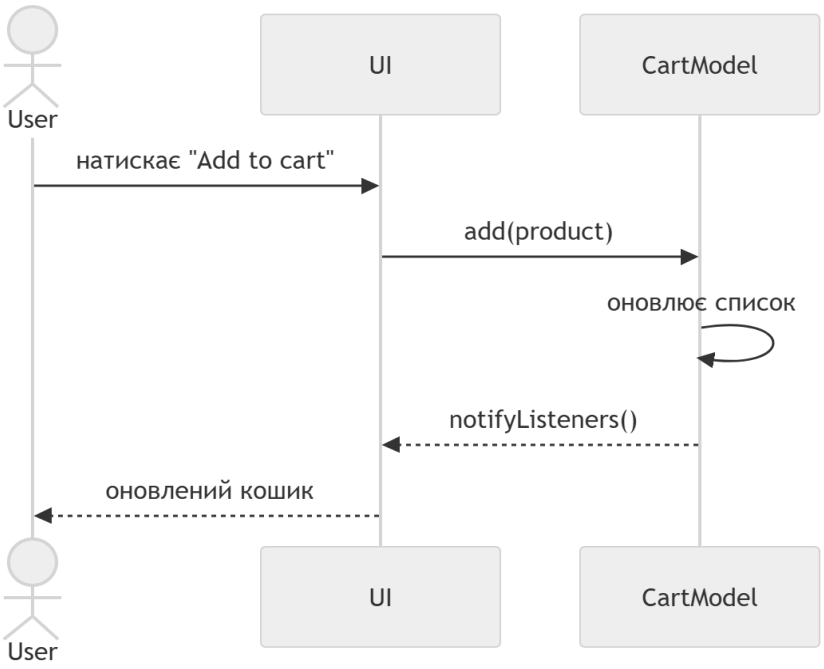

Рисунок А.3. Діаграма послідовності (Sequence Diagram) — процес додавання товару до кошика або кастомізації

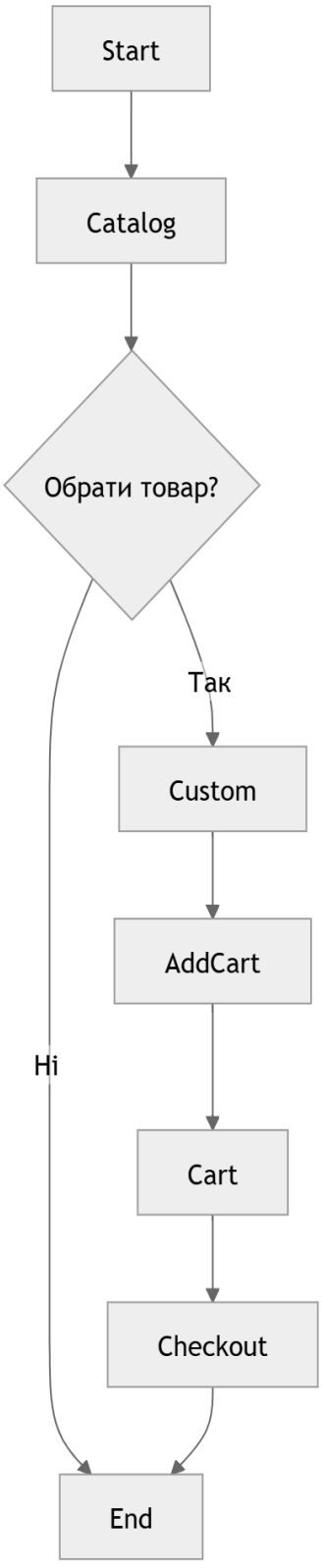

Рисунок А.4. Діаграма діяльності (Activity Diagram)

Рисунок А.5. Компонентна діаграма

ДОДАТОК Б

Інтерфейси користувача вебзастосунку Szpilka

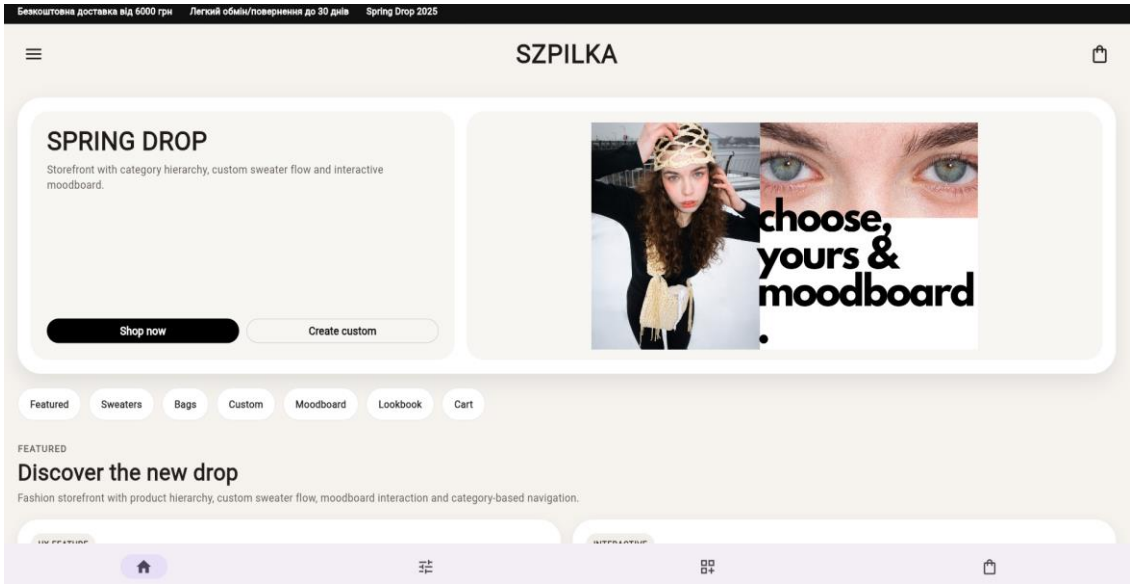

Рисунок Б.1. Головний екран каталогу товарів

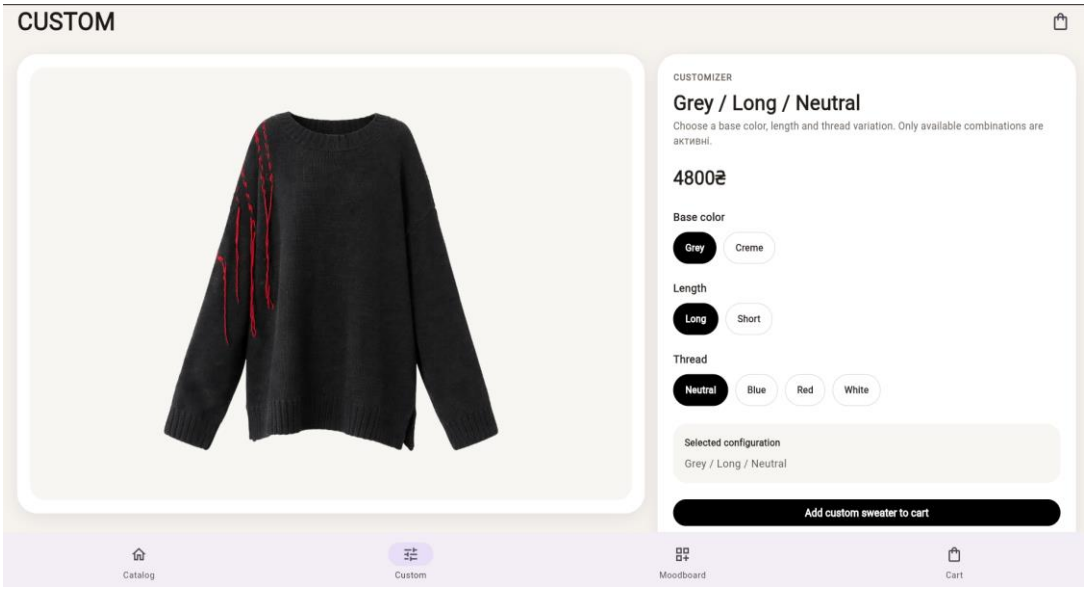

Рисунок Б.2. Екран модуля кастомізації

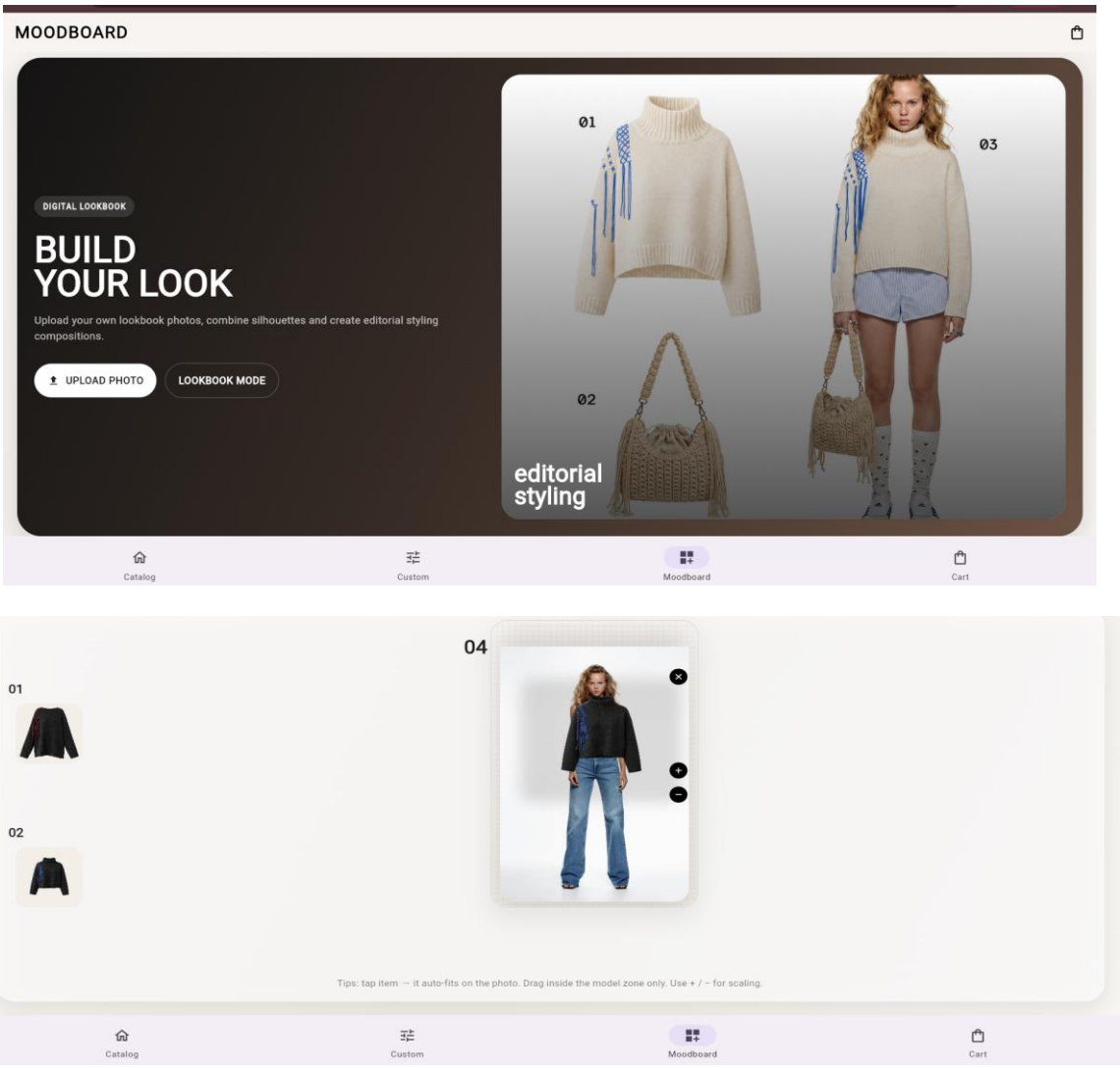

Рисунок Б.3. Інтерфейс moodboard

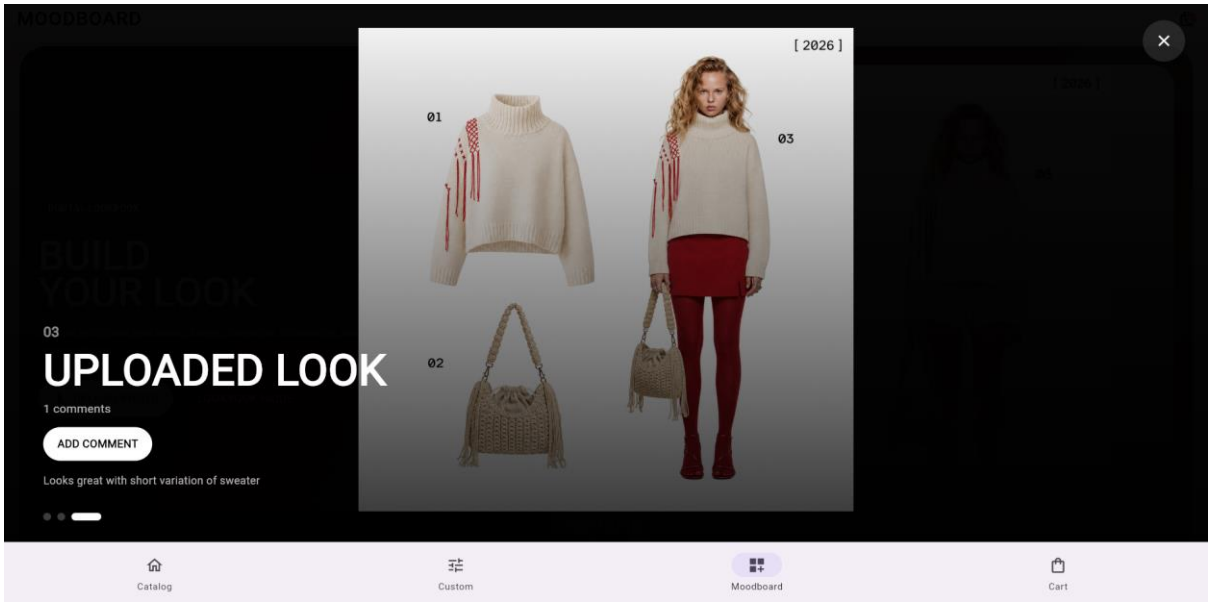

Рисунок Б.4. Інтерфейс lookbook

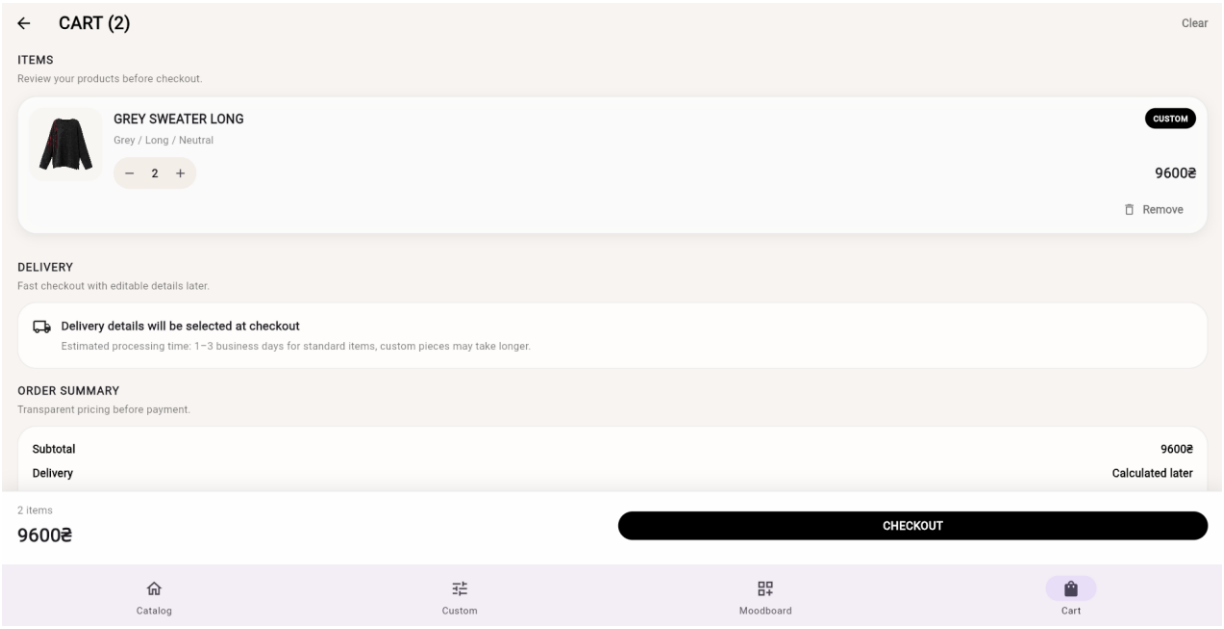

Рисунок Б.5. Екран кошика (з товарами)

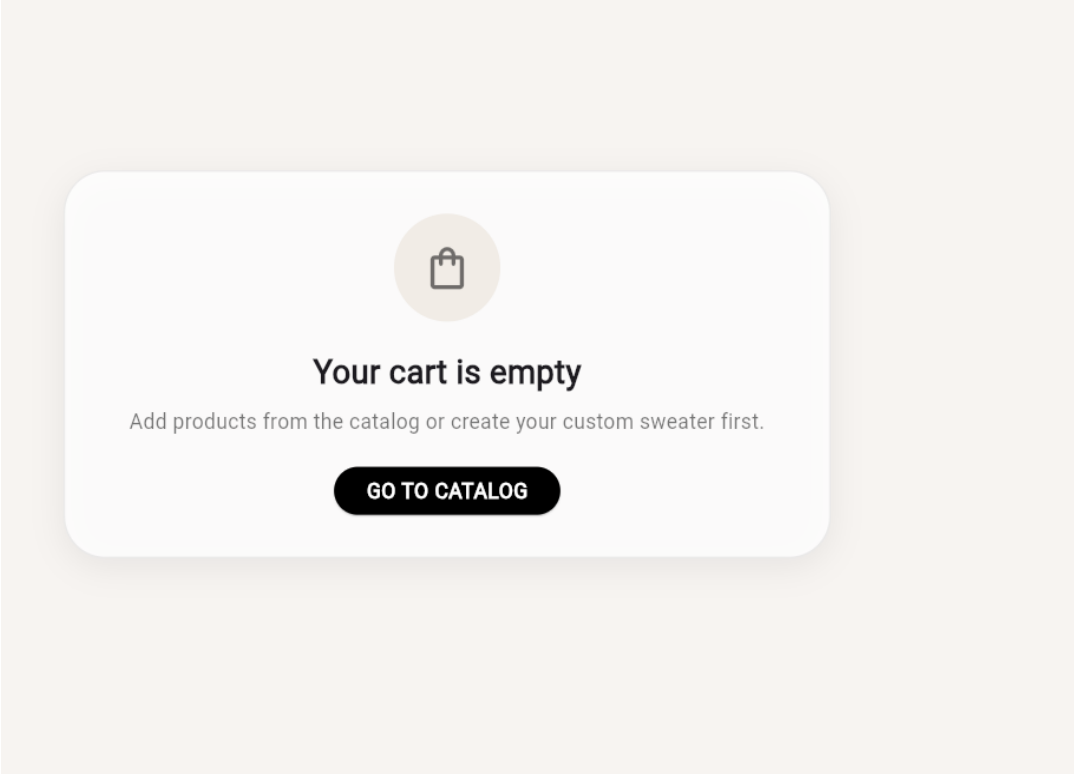

Рисунок Б.6.Порожній стан кошика

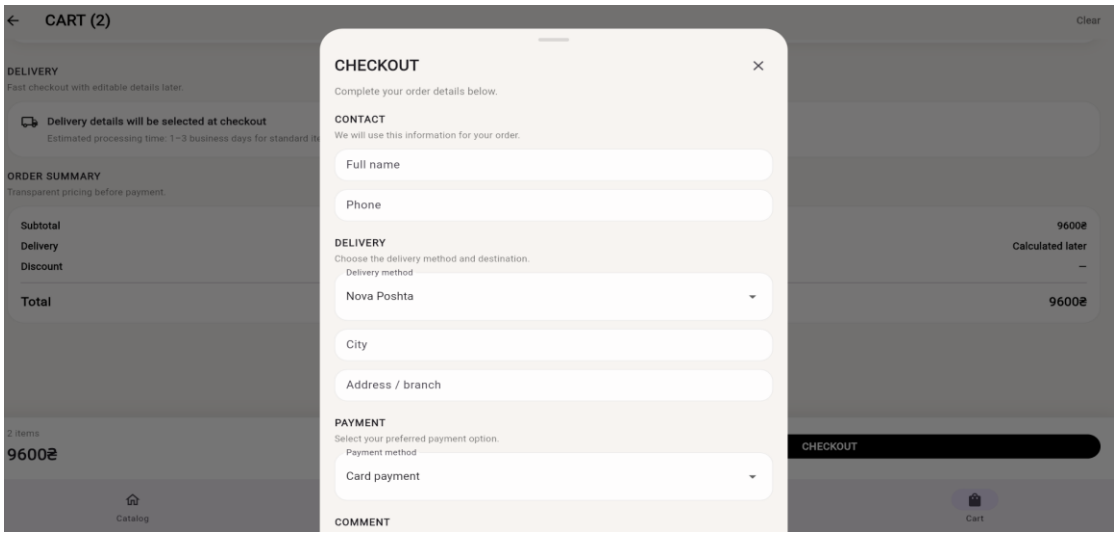

Рисунок Б.7. Форма оформлення замовлення (Checkout Sheet)

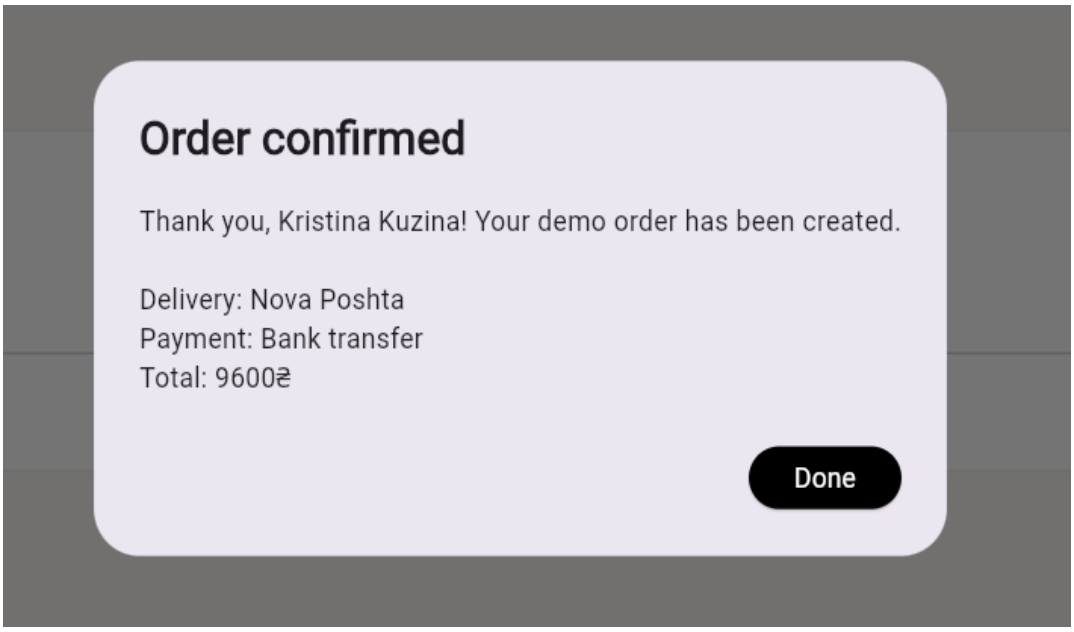

Рисунок Б.8. Підтвердження успішного оформлення замовлення

ДОДАТОК В

Лістинги програмного коду

В.1 Основний файл main.dart (ініціалізація Firebase та MultiProvider)

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:firebase_core/firebase_core.dart';
import 'firebase_options.dart';
import 'cart/cart_model.dart';
import 'models/moodboard.dart';
import 'screens/catalog_screen.dart';
import 'screens/cart_screen.dart';
import 'screens/custom_screen.dart';
import 'screens/moodboard_screen.dart';
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(const SzpilkaApp());
}
class SzpilkaApp extends StatelessWidget {
  const SzpilkaApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (_) => CartModel()),
```

```
            ChangeNotifierProvider(create: (_) =>
MoodboardModel()),
          ],
          child: MaterialApp(
            debugShowCheckedModeBanner: false,
            title: 'Szpilka',
            theme: ThemeData(
              useMaterial3: true,
              scaffoldBackgroundColor: const Color(0xFFF6F3EF),
              fontFamily: 'Arial',
            ),
            home: const MainNavigationShell(),
          ),
        );
      }
    }
```

B.2 Модель CartModel (фрагмент)

```
class CartModel extends ChangeNotifier {
  final Map<Product, int> _items = {};
  final List<CustomCartEntry> _customEntries = [];
  Map<Product, int> get items => _items;
  List<CustomCartEntry> get customEntries => _customEntries;
  int get totalItems => _items.values.fold(0, (sum, qty) =>
sum + qty) +
                        _customEntries.fold(0, (sum, entry) =>
sum + entry.quantity);
  double get subtotal => /* розрахунок вартості */;
  double get totalPrice => subtotal + deliveryFee - discount;
```

```
    void add(Product product) { ... }

    void addCustom({required Product product, required String
label, String? customImage}) { ... }

    void clear() {

      _items.clear();

      _customEntries.clear();

      notifyListeners();

    }

  }
```

B.3 Модель CustomizerModel

```
  class CustomizerModel extends ChangeNotifier {

    final List<SweaterVariant> variants = const [ /* 8
варіантів светрів */ ];

    late SweaterBaseColor selectedColor;

    late SweaterLength selectedLength;

    late ThreadTone selectedThread;

    SweaterVariant get currentVariant { ... }

    void selectColor(SweaterBaseColor color) {

      selectedColor = color;

      _normalizeThreadSelection();

      notifyListeners();

    }

    void selectLength(SweaterLength length) { ... }

    void selectThread(ThreadTone thread) { ... }

  }
```

В.4 Фрагмент екрану MoodboardScreen (основна логіка)

```
class MoodboardScreen extends StatefulWidget { ... }


class _MoodboardScreenState extends State<MoodboardScreen> {
  final MoodboardModel moodboard =
context.watch<MoodboardModel>();
  // Drag & Drop логіка
  void _addToBoard(Product product) {
    moodboard.addToBoard(product);
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: [
          // Background
          ...moodboard.boardItems.map((item) => Positioned(
            left: item.position.dx,
            top: item.position.dy,
            child: GestureDetector(
              onPanUpdate: (details) { /* переміщення */ },
              child: Transform.scale(
                scale: item.scale,
                child: Image.asset(item.product.image),
              ),
            ),
          )),
```

```
        ],
}
```

В.5 Компонент CheckoutSheet (форма оформлення замовлення)

```
class CheckoutSheet extends StatefulWidget { ... }
class _CheckoutSheetState extends State<CheckoutSheet> {
  final _formKey = GlobalKey<FormState>();
  final _nameController = TextEditingController();
  // ... інші контролери
  void _submit(BuildContext context) {
    if (!_formKey.currentState!.validate()) return;
    // Логіка підтвердження замовлення
    final cart = context.read<CartModel>();
    cart.clear();
    Navigator.pop(context);
    // Показати діалог підтвердження
  }
}
```

В.6 Фрагмент firebase_options.dart

```
static const FirebaseOptions web = FirebaseOptions(
  apiKey: 'AIzaSyArcgWNXeFwSgErAKqs2Q_xeinOBkNEz_4',
  appId: '1:218761882908:web:199e1586776bae0aef1ed9',
  messagingSenderId: '218761882908',
  projectId: 'flower-etc-shop',
  authDomain: 'flower-etc-shop.firebaseapp.com',
  storageBucket: 'flower-etc-shop.firebasestorage.app',
  measurementId: 'G-87Z2PY92KR',
);
```

ДОДАТОК Г

Результати тестування

Таблиця Г.1

Результати функціонального тестування

ID сценарію	Опис сценарію	Очікуваний результат	Фактичний результат	Статус
TC-01	Перехід між розділами через нижню навігацію	Миттєвий перехід без перезавантаження сторінки	Перехід працює коректно	Пройдено
TC-02	Фільтрація товарів за категорією	Відображення лише товарів обраної категорії	Фільтрація працює	Пройдено
TC-03	Кастомізація светра (зміна кольору, довжини, нитки)	Миттєве оновлення зображення прев'ю	Оновлення працює в реальному часі	Пройдено
TC-04	Додавання кастомізованого товару до кошика	Товар з'являється в кошику з правильними параметрами	Додається коректно	Пройдено
TC-05	Drag-and-drop елементів у moodboard	Можливість вільного переміщення та масштабування	Функціональність працює	Пройдено
TC-06	Додавання товару до moodboard	Елемент з'являється на дошці	Працює стабільно	Пройдено
TC-07	Оформлення замовлення (заповнення форми)	Валідація полів + підтвердження	Форма працює, валідація присутня	Пройдено
TC-08	Очищення кошика	Видалення всіх товарів	Кошик очищається	Пройдено
TC-09	Робота застосунку без підключення до інтернету	Повна функціональність (крім Firebase)	Працює автономно	Пройдено
TC-10	Масштабування інтерфейсу (resize)	Коректне відображення при зміні розміру вікна десктопного браузера	Адаптивність на високому рівні	Пройдено

ДОДАТОК Д

Логічна структура бази даних вебзастосунку Szpilka

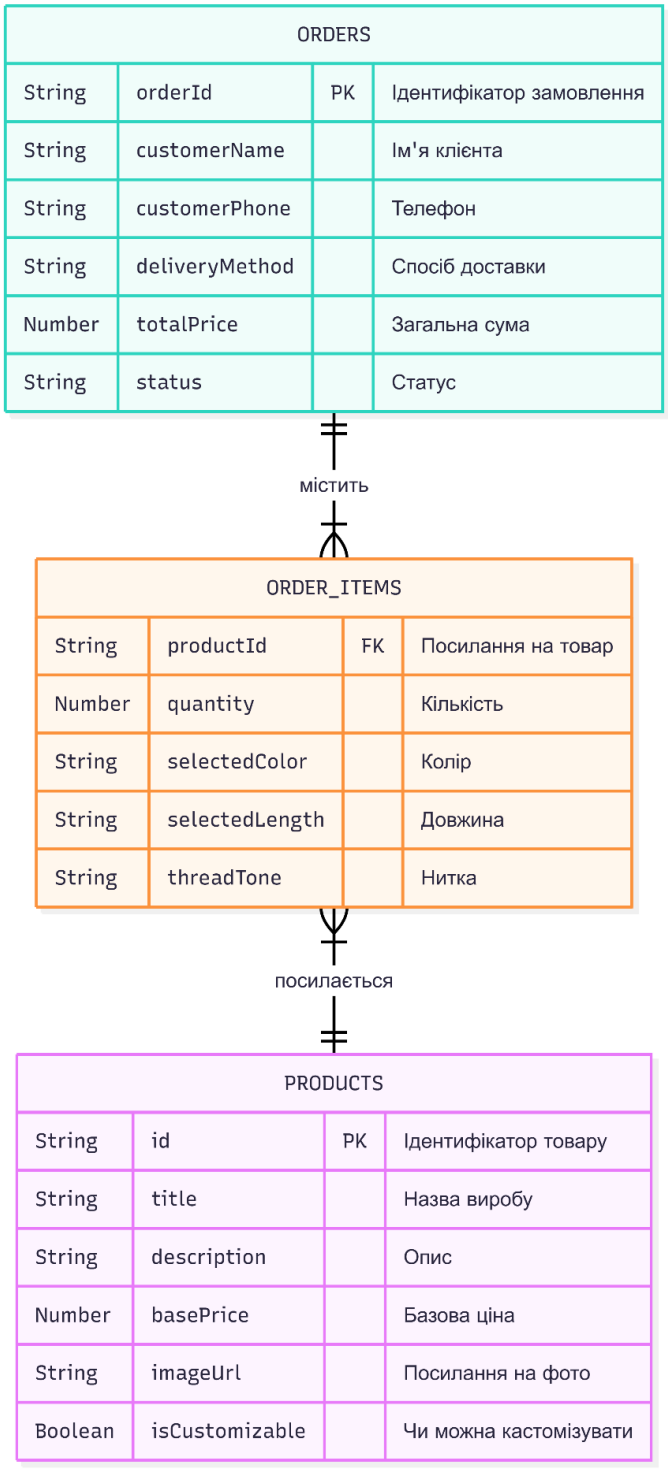

Рисунок Д.1 Форма оформлення замовлення (Checkout Sheet)