Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра факультету інформатики

E-commerce додаток для платформи iOS
Текстова частина до курсової роботи
за спеціальністю „Комп'ютерні науки"

Керівник курсової роботи
ас. Бітаєва О.В.

_____
(підпис)
"____" _____ 2023 р.

Виконав студент
КН-БП4 Колесніков А.О
"____" _____ 2023 р.

Київ 2023

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра факультету інформатики

ЗАТВЕРДЖУЮ

Доктор технічних наук, доцент.

_____А. М. Глибовець

(підпис)

“____”_____2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Колеснікову А.О. факультету інформатики 4 курсу

ТЕМА: E-commerce додаток для платформи iOS

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

Обґрунтування актуальності теми. Мета та завдання дослідження

Проєктування моделі бази даних

Розробка UI/UX дизайну для додатка

Тестування користувальницького інтерфейсу

Розробка програмної частини додатка використовуючи мову програмування Swift

Тестування готового додатка

Планування майбутнього додаткового функціонала та стратегія розвитку

Висновки

Використана література

Додатки

Дата видачі “____” _____ 2023 р. Керівник _____ (підпис)

Завдання отримав _____ (підпис)

Зміст

Вступ

Тема дипломної роботи полягає в розробці e-commerce додатку для платформи iOS. Основною метою роботи є створення функціонального та зручного в експлуатації мобільного додатку, який дозволить користувачам здійснювати покупки, а також отримувати різноманітну інформацію про продукти.

Для досягнення цієї мети, досліджуватимуться сучасні тенденції у сфері мобільного програмного забезпечення, зокрема звернемо увагу на технології розробки додатків для iOS, такі як мова програмування Swift, інтеграцію з Firebase, налаштування бази даних та серверної частини.

Також будуть розглянуті питання, пов'язані з дизайном інтерфейсу користувача, включаючи аналіз дизайну конкурентів та розробку ефективного UI/UX дизайну, а також тестування користувацького інтерфейсу.

Додатково буде розглянуто розробку програмної частини додатка, включаючи опис структури даних та реалізацію інтерфейсу додатка з використанням мови програмування Swift. Крім того, буде проведене тестування готового додатку на реальних користувачах та розроблені стратегії розвитку для подальшого вдосконалення та додавання нового функціоналу.

В результаті дослідження та розробки мобільного e-commerce додатку для iOS, дипломна робота має за мету забезпечити користувачів зручним та доступним інструментом для здійснення покупок та отримання різноманітної інформації про продукти.

РОЗДІЛ 1 Обґрунтування актуальності теми. Мета та завдання дослідження

1.1 Обґрунтування актуальності теми

Згідно з даними Statista [1] на березень 2023 року, на світі було понад 2,3 мільярда активних пристроїв iOS, що становить більше ніж 27% ринку мобільних пристроїв. Це робить iOS однією з найбільш популярних мобільних платформ у світі.

За дослідженням Salesforce [2], мобільні пристрої (включаючи iOS) продовжують зберігати позицію основного каналу для онлайн-шопінгу. За даними на березень 2023 року, понад 80% онлайн-покупок, що здійснюються в США, здійснюються через мобільні пристрої.
За даними Shopify [3], магазини, які використовують мобільні додатки, отримують у 3 рази більше замовлень в порівнянні з магазинами, які не мають мобільного додатка.

За дослідженням eMarketer [4], більше ніж 85% всіх користувачів мобільних пристроїв в США здійснюють щонайменше одну покупку через мобільний додаток щомісяця. Прогнозується, що до 2024 року, мобільні пристрої стануть основним каналом для онлайн-шопінгу, і понад 90% всіх онлайн-покупок будуть здійснюватись через мобільні пристрої.

Отже, створення E-commerce додатка для платформи iOS має високий потенціал для успіху та зростання в електронній комерції.

1.2 Мета та завдання дослідження

Метою дослідження є розробка та реалізація E-commerce додатку для платформи iOS з метою покращення процесу онлайн-шопінгу на мобільних пристроях для користувачів iOS.

Завданнями дослідження є:

1. Аналіз наявних E-commerce додатків для платформи iOS, їхніх особливостей та недоліків.
2. Проєктування моделі бази даних, необхідної для зберігання інформації про продукти, замовлення, користувачів та інших необхідних даних.
3. Розробка інтерфейсу додатка з урахуванням дизайну та зручності використання для користувачів.
4. Реалізація функціонала додатка, включаючи пошук продуктів, перегляд каталогу, додавання продуктів до кошика, оформлення замовлення та інші необхідні функції.
5. Тестування додатка з метою виявлення та усунення можливих помилок та недоліків.
6. Розробка стратегії розвитку додатка та визначення можливостей для покращення його функціональності та зручності використання.
7. Визначення ефективності додатка та його впливу на процес онлайн-торгівлі для користувачів iOS.

Мета та завдання дослідження будуть досягнуті шляхом аналізу інтернет-ресурсів, проєктування та розробки додатка, його тестування та вдосконалення. Також будуть проведені опитування користувачів для оцінки ефективності додатка та його впливу на їхню покупкову поведінку.

РОЗДІЛ 2 Проєктування моделі бази даних

2.1 Опис потреби у базі даних для розробки додатка

У сучасному світі електронна комерція є надзвичайно популярним видом бізнесу, і все більше клієнтів віддають перевагу онлайн-шопінгу. З ростом популярності електронної комерції зростає і потреба у відповідних

технологіях, які допоможуть забезпечити ефективну роботу онлайн-магазину.

Для розробки додатка e-commerce з продажу брендових речей необхідна відповідна база даних, яка забезпечує зберігання важливої інформації про товари, клієнтів, замовлення та оплати. Інформація про товари дозволяє відстежувати наявність товарів на складі, їх характеристики та ціни. Інформація про клієнтів допоможе встановити контакт з покупцями та відстежувати їх історію замовлень. Інформація про замовлення та оплати забезпечує можливість відстеження процесу доставки та оплати товарів. База даних є необхідною складовою для ефективної роботи додатка e-commerce, який забезпечує якість обслуговування клієнтів та оптимізацію роботи онлайн-магазину. Для досягнення успіху у сфері електронної комерції необхідно забезпечити надійну та швидку роботу додатка, що неможливо без належної бази даних. Тому розробка відповідної бази даних є однією з головних задач у процесі розробки додатка e-commerce з продажу брендових речей.

2.2 Вибір засобів розробки бази даних та обгрунтування цього вибору

Початково для серверної частини було обрано NoSQL базу даних – MongoDB, але після певного періоду розробки виявилося, що MongoDB не надає необхідний функціонал і доведеться використовувати додаткові сервіси. Тому було вирішено замінити MongoDB на Firebase. Firebase - це платформа для створення мобільних та вебзастосунків, розроблена компанією Firebase, Inc. у 2011 році. Вона включає 19 продуктів [5], які використовують 2602 [6] IT-компаній.

У застосунку використовуються такі сервіси Firebase:

- Google Analytics – безплатне рішення для аналізу використання додатка та його користувачів.

- Firebase Cloud Messaging (FCM) – кросплатформне рішення для обміну повідомленнями, яке дозволяє безплатно і надійно надсилати повідомлення.
- Firebase Authentication – це серверні послуги, прості SDK та готові бібліотеки UI для аутентифікації користувачів в додатку. Підтримуються аутентифікація за допомогою паролів, телефонних номерів та облікових записів Google, Facebook, Twitter і т.д.
- Cloud Firestore – гнучка, масштабована база даних для мобільних, вебзастосунків та серверів від Firebase і Google Cloud. Вона синхронізує дані між клієнтськими додатками у реальному часі та надає автономну підтримку для мобільних пристроїв, що дозволяє створювати додатки, які працюють незалежно від затримки мережі чи підключення до Інтернету. Cloud Firestore також пропонує інтеграцію з іншими продуктами Firebase та Google Cloud, включаючи хмарні функції.

Така конфігурація дозволяє зосередитись на одній платформі, що спрощує налаштування сервісів та гарантує оптимальну інтеграцію між ними. Оскільки Cloud Firestore не має схем, це дає вам повну свободу вибору полів, які додають до кожного документа, і типів даних, які зберігають у цих полях.

Використання Firebase для серверної частини додатка надає низку переваг, таких як:

1. Швидкість розробки: надає набір готових до використання сервісів, що дозволяє зосередитись на створенні функціонала додатка, замість витрати часу на налаштування інфраструктури.
2. Масштабованість: допомагає масштабувати додатки відповідно до потреб користувачів, завдяки високій пропускній здатності та автоматичному масштабуванню ресурсів.

3. Безпека: забезпечує захист даних та авторизацію користувачів, що дозволяє контролювати доступ до ресурсів та захищати приватність користувачів.

4. Інтеграція з іншими сервісами Google: пропонує взаємодію з іншими продуктами Google Cloud, що дозволяє створювати потужні та зручні застосунки.

Таким чином, використовуючи Firebase для серверної частини додатка, розробники отримують ідеально оптимізовані та інтегровані сервіси на одній платформі. Це значно полегшує розробку та налаштування сервісів, а також дозволяє створювати швидкі, масштабовані та безпечні мобільні застосунки.

2.3 Налаштування бази даних та серверної частини

Щоб інтегрувати Firebase з додатком, потрібно слідувати спеціалізованому керівництву [7], яке містить п'ять етапів:

1. Завести новий проєкт на платформі Firebase.
2. Зареєструвати додаток у Firebase.
    - Щоб це зробити, перейти до консолі Firebase;
    - Вибрати платформу, на якій розробляти додаток;
    - Ввести ідентифікатор додатка, який можна знайти в налаштуваннях проєкту у Xcode;
    - Натиснути "Зареєструвати додаток".
3. Додати файл налаштувань Firebase.
    - Завантажити GoogleService-Info.plist, щоб отримати файл налаштувань Firebase для відповідної платформи;
    - Розмістити цей файл у каталозі додатка.
4. Включіть SDK [8] Firebase.

- Відкрити файл проєкту додатка у Xcode, перейдіть до меню "Add Packages" та додати потрібну версію Firebase SDK;
- Вибрати бібліотеку, яку застосовувати.

5. Ініціалізувати Firebase у додатку.

- Імпортувати модуль FirebaseCore до класу UIApplicationDelegate та інші модулі залежно від необхідних сервісів Firebase.

2.4 Опис структури даних

В Cloud Firestore основною одиницею зберігання є документ. Документ це спрощений запис, що містить поля, які відображаються на значення. Кожен документ має унікальне ім'я. Документи знаходяться у колекціях, які слугують простими контейнерами для документів.

На момент написання дипломної роботи в базі даних використовуються колекції products, sizecharts та users.

У колекції users зберігаються записи з інформацією про зареєстрованих користувачів.

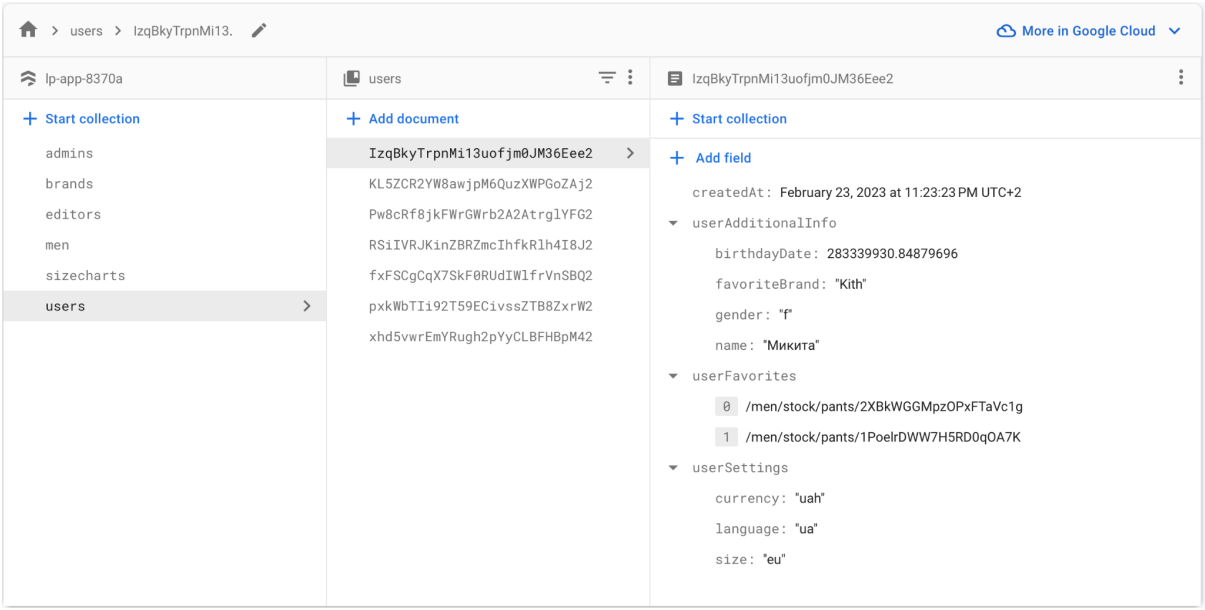


Рисунок 2.4.1 Колекція users

У sizechart знаходяться документи для кожного бренду одягу, у яких представлені категорії предметів та відповідні розмірні сітки для кожної категорії.

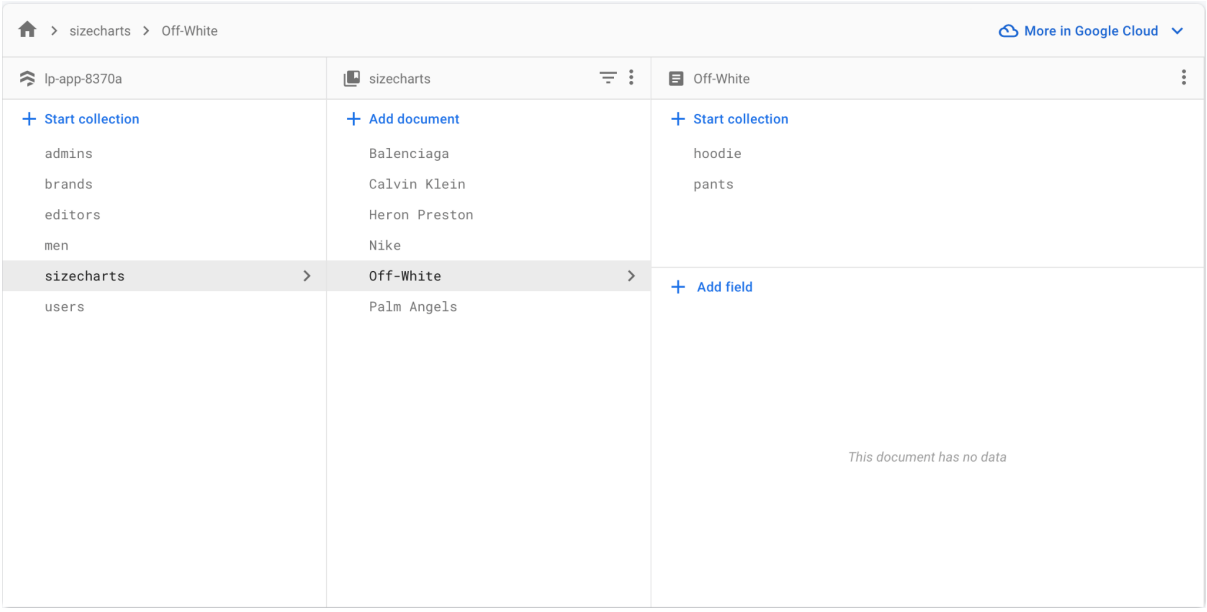


Рисунок 2.4.2 Колекція sizechart

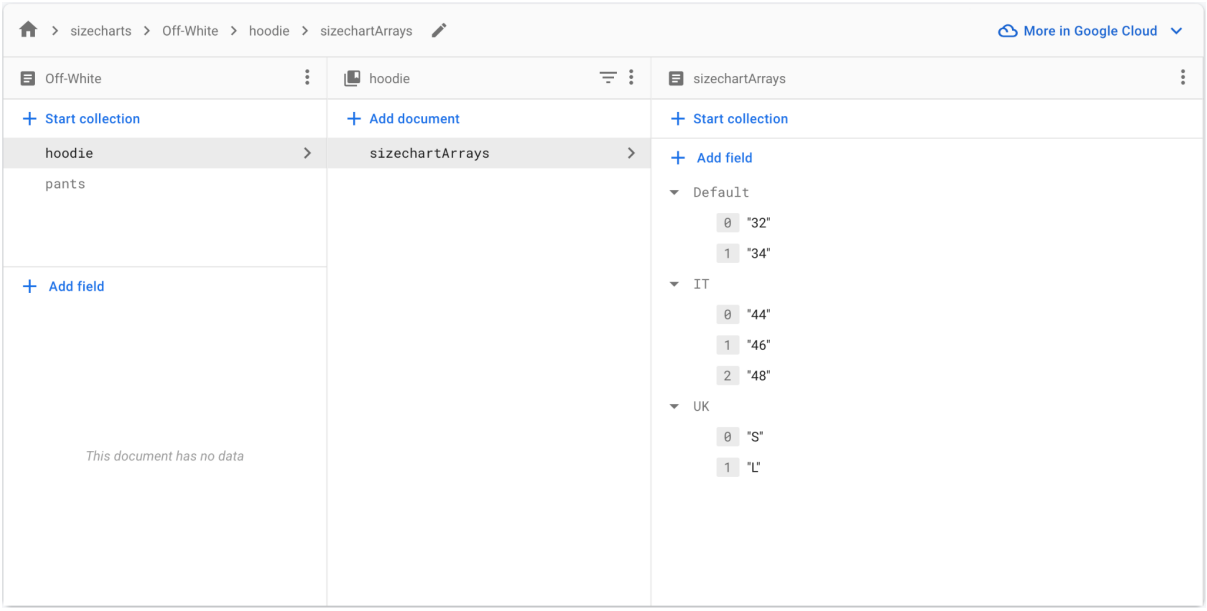


Рисунок 2.4.3 Колекція sizechart

Колекція products поділяється на order (предмети на замовлення) та instock (предмети в наявності). В кожній колекції є категорії товарів з предметами.

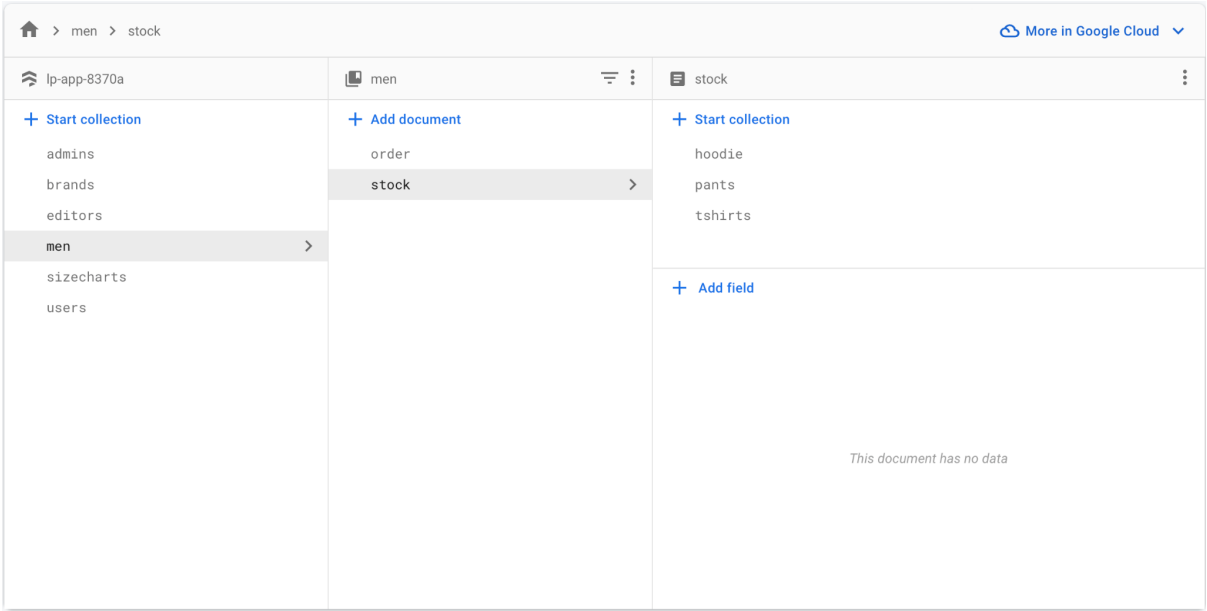


Рисунок 2.4.4 Колекція products/stock

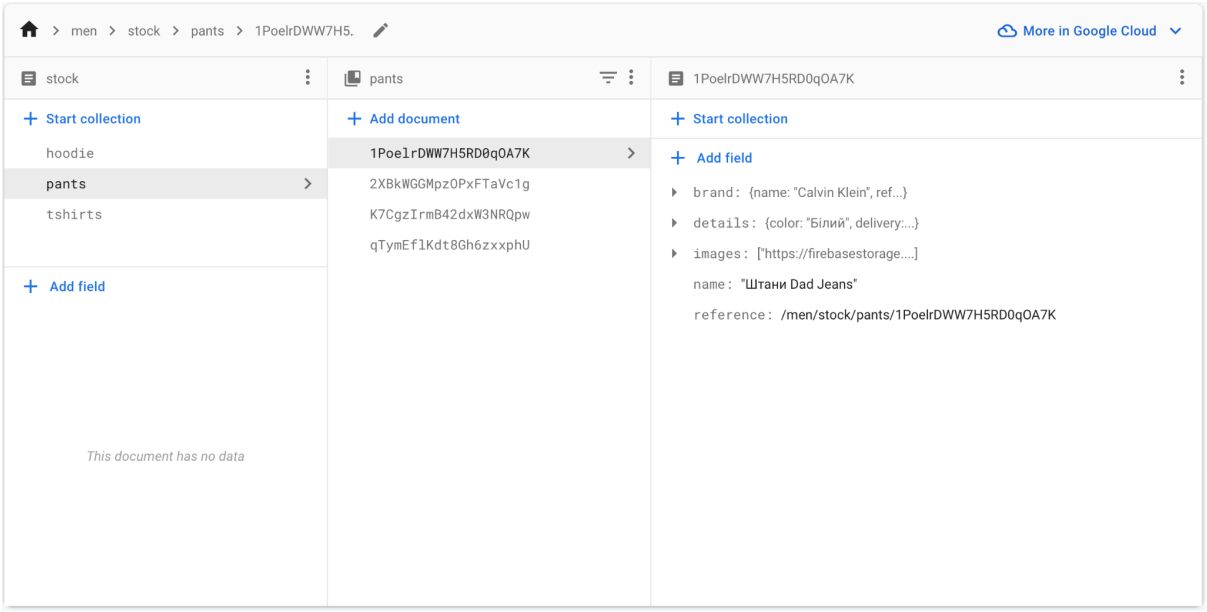


Рисунок 2.4.5 Колекція products/stock

РОЗДІЛ 3 Розробка UI/UX дизайну для додатка

3.1 Опис структури додатка

Додаток складається з 24 екранів (див. рис. 3.1.1).

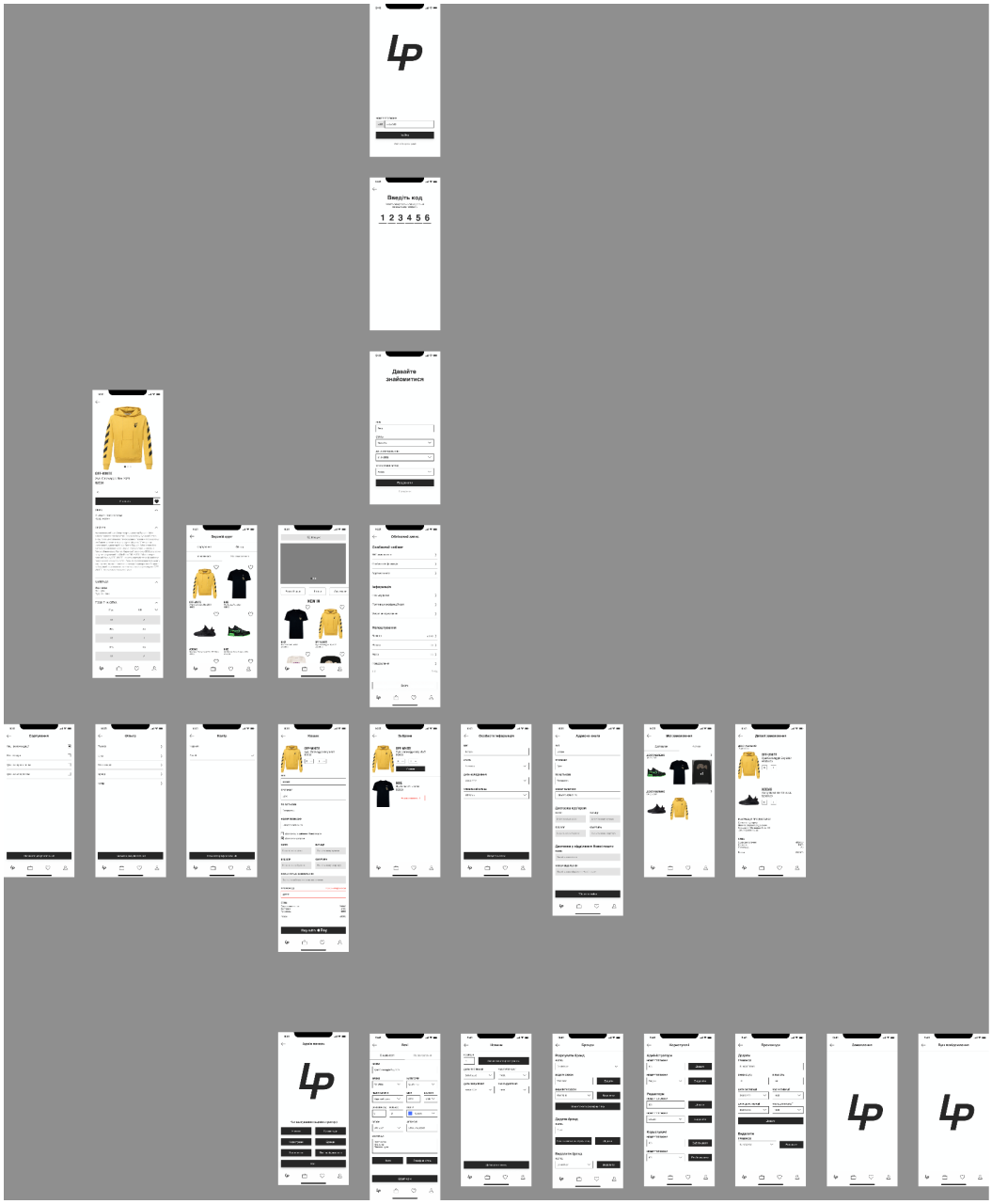


Рисунок 3.1.1 Всі екрани додатка

Відразу після запуску додатка користувач може здійснити реєстрацію за номером телефону, увійти, якщо вже має обліковий запис, або скористатися додатком без реєстрації, якщо бажає ознайомитися з його функціональністю.

Якщо користувач вирішує увійти за допомогою номера телефону, на наступному екрані йому слід підтвердити номер телефону,

використовуючи одноразовий код, який надійде у вигляді SMS-повідомлення.

Після успішного підтвердження номера телефону користувач потрапляє на екран, де йому пропонують розповісти трохи більше про себе та свої вподобання. Також існує можливість пропустити цей етап.

На головному екрані користувач може переглянути новини, анонси, використати пошук, ознайомитися з новими товарами та вибрати категорію для покупок. Саме сюди потрапляє користувач, який обрав опцію не реєструватися при вході. Внизу екрана є меню для доступу до головного меню, пошуку товарів, кошика, списку вподобань та налаштувань додатка.

Після вибору категорії для покупок з'являється екран з товарами, що відповідають цій категорії. Товари можна відсортувати за рекомендаціями, датою додавання, ціною за зростанням та спаданням. Також доступні фільтри за розміром, ціною, типом товару, брендом та кольором. На цій сторінці можна вибрати товари з наявності або під замовлення.

На сторінці товару представлено декілька зображень, можливість вибрати розмір та додати товар до кошика або списку вподобань. Для кожного товару наведена детальна інформація, яка включає:

- Опис (ID моделі та колір);
- Доставка (термін доставки);
- Матеріали товару;
- Розмірні сітки, де можна обрати потрібний розмір відповідно до декількох розмірних систем.

Якщо користувач хоче придбати товар, він переходить до оформлення замовлення, натиснувши кнопку "У кошик". На цьому етапі потрібно вказати кількість і розмір виробу, дані замовника, метод доставки (кур'єром або поштою), інформацію для доставки та промокод на знижку, якщо такий є. Після заповнення відповідної інформації, на екрані доступна кнопка Apple Pay для миттєвої оплати замовлення.
У разі успішної оплати клієнт може стежити за статусом замовлення у своєму кабінеті, де можна також переглянути деталі замовлення (ціну, приблизну дату та дані для доставки).

Екран списку вподобань дозволяє додати товар до кошика або, якщо він відсутній на складі, отримати push-повідомлення про наявність товару у продажі.

У розділі налаштувань присутні такі пункти:

- Особистий кабінет (мої замовлення, особиста інформація, адресна книга);
- Служба підтримки (інформація про магазин, політика конфіденційності, умови використання, контактні дані);
- Налаштування (валюта, розмір, мова, push-повідомлення).

На екрані особистої інформації користувач має змогу змінювати дані про себе, вказані під час реєстрації.

У розділі адресної книги можна внести інформацію (особисті дані, метод та адреса доставки), яка автоматично буде використовуватися для майбутніх замовлень.

Наступні екрани доступні виключно для адміністраторів магазину. З головного екрана можна перейти до налаштувань:

- Новини;
- Промокоди;
- Користувачі;
- Бренди;
- Замовлення;
- Push-сповіщень;
- Товарів.

3.2 Аналіз дизайну конкурентів

Згідно з опитуванням клієнтів магазину, найпопулярнішими мобільними додатками магазинів є Farfetch та Asos. Важливо, щоб новий додаток для користувачів був зручним, знайомим і інтуїтивно зрозумілим.

Основною метою аналізу дизайну конкурентів було створення гармонійної суміші дизайну цих двох додатків з власними ідеями щодо функціонала, використовуючи власний стиль оформлення та дизайну інтерфейсу.

На головному екрані Asos зверху розташований пошук, потім головні новини та вибірка товарів. У новому додатку було реалізовано пошук та головні новини, але також додано можливість переглядати кілька головних новин поспіль. Крім того, на цьому екрані можна відразу оглянути нові товари та вибрати категорію для покупок.

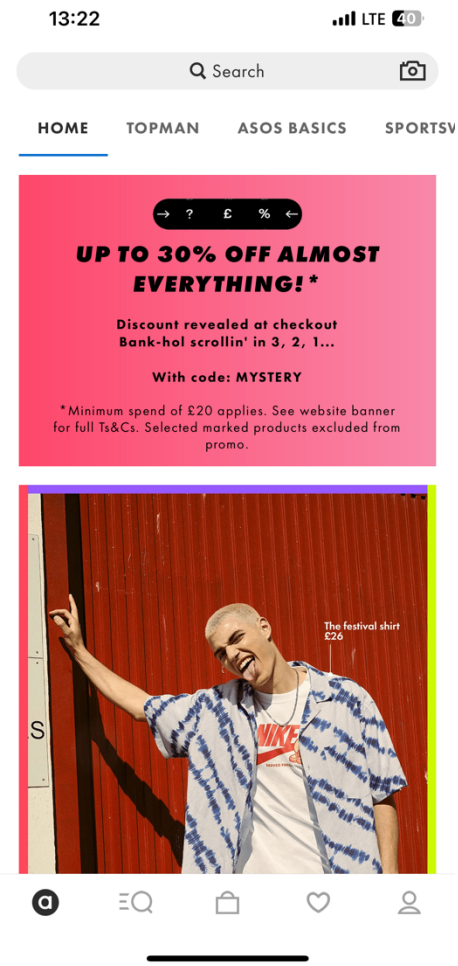


Рисунок 3.2.1 Додаток Asos

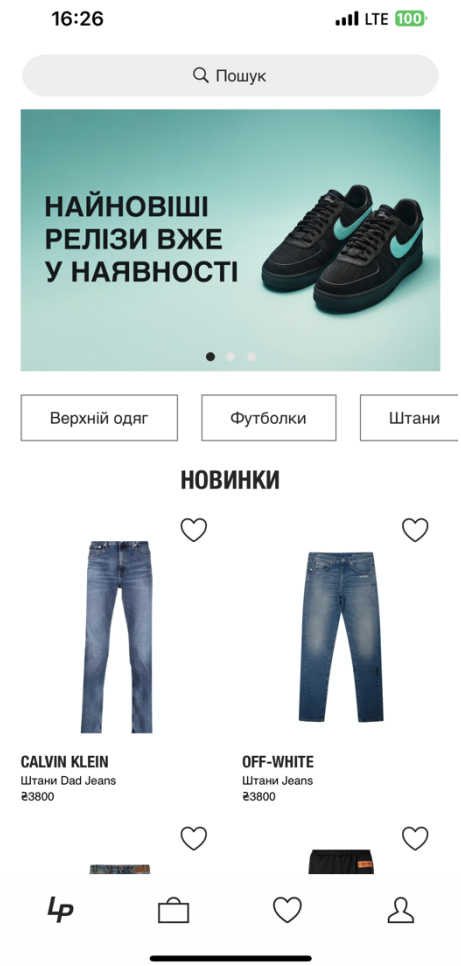


Рисунок 3.2.2 Мій *додат*ок

Екран для покупок має класичний вигляд, дозволяє відразу сортувати та фільтрувати товари, а також перемикатися між доступними товарами та товарами на замовлення.

На екрані товару в додатку Farfetch дизайн інформації про товар виявився привабливим. Схожий дизайн було впроваджено у новому додатку.

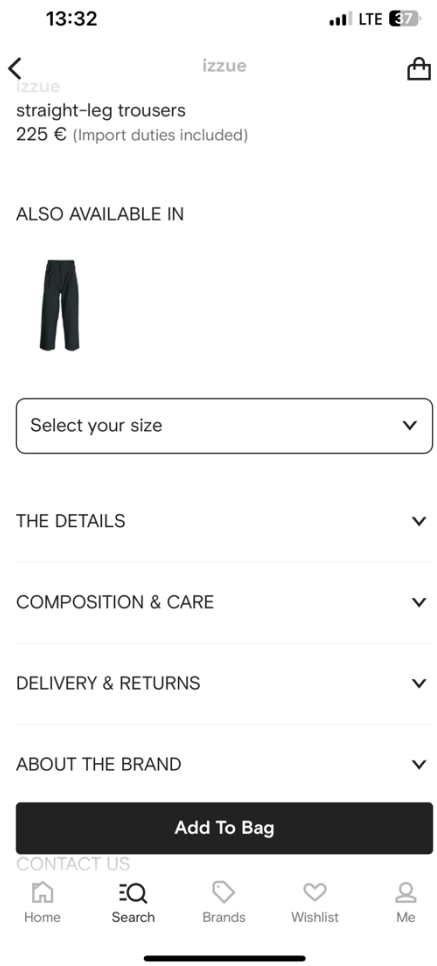


Рисунок 3.2.5 Додаток Farfetch

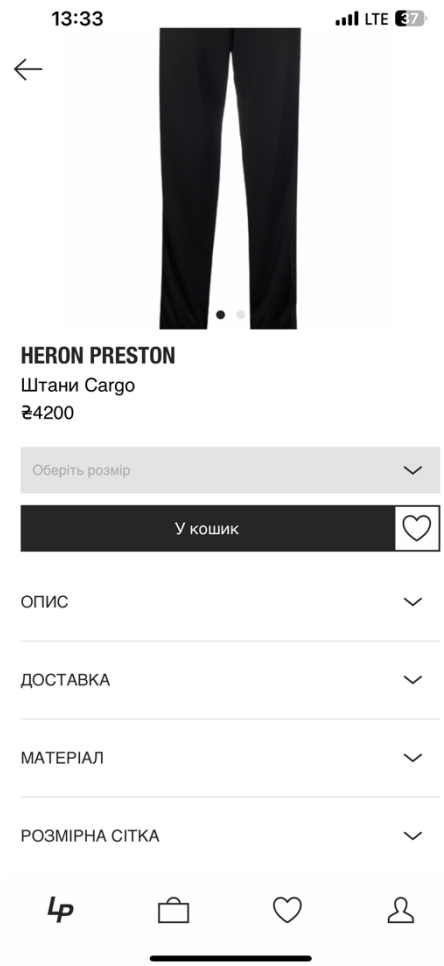


Рисунок 3.2.6 Мій додаток

Фільтри та сортування виконані у простому та мінімалістичному дизайні. При потребі можна швидко скинути всі вибрані фільтри. Також обрані фільтри відображаються поряд з категорією для фільтрації.

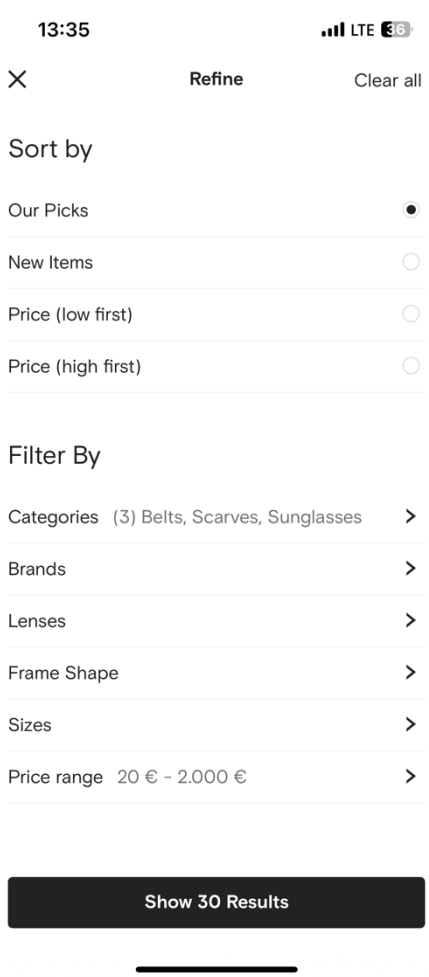


Рисунок 3.2.8 Додаток Farfetch

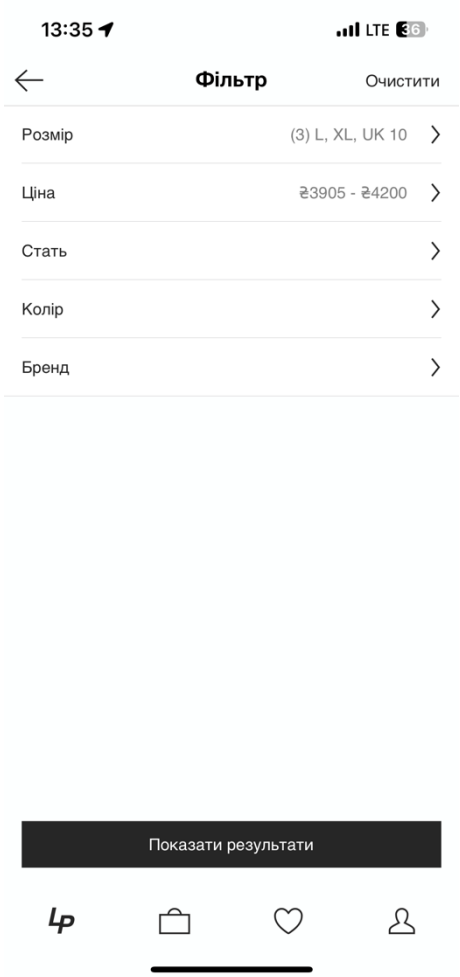


Рисунок 3.2.9 Мій додаток

Екран налаштувань було створено, використовуючи вдосконалене рішення з додатка Farfetch. Налаштування розбито на категорії та підкатегорії, що дозволяє користувачеві легко знайти потрібну опцію.

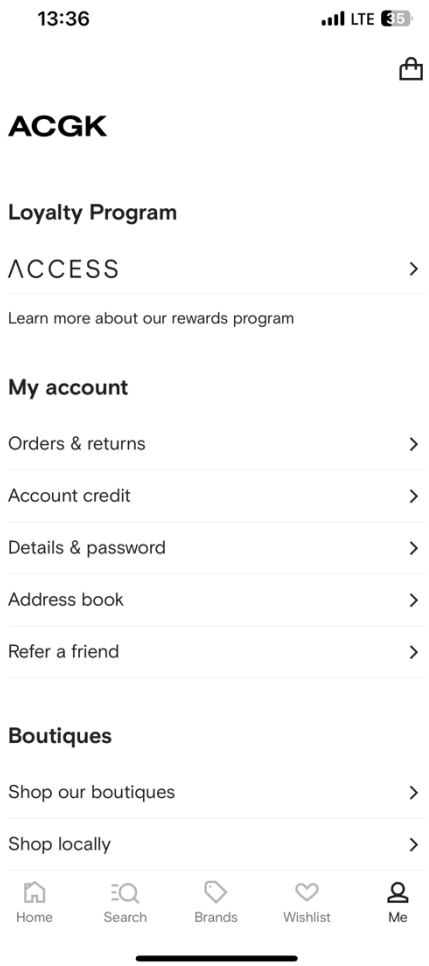


Рисунок 3.2.10 Додаток Farfetch

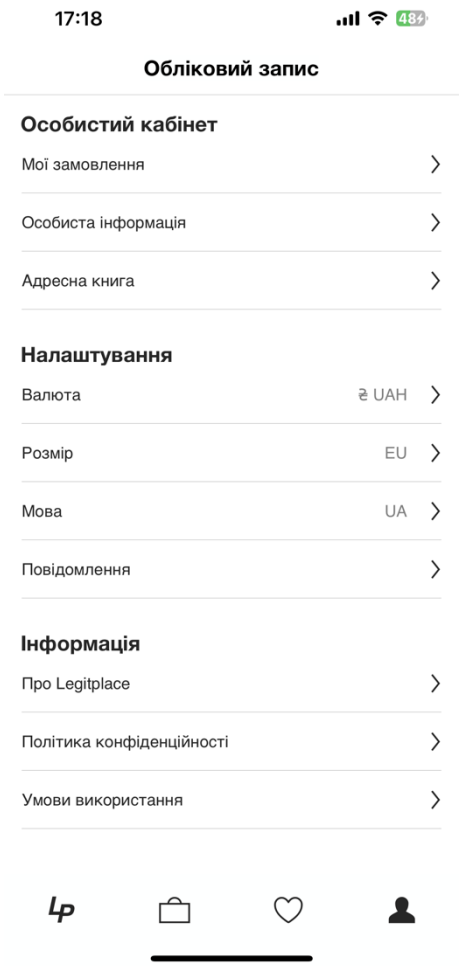


Рисунок 3.2.11 Мій додаток

Екран зі списком обраних товарів було максимально спрощено, сприяючи зосередженню користувача на основній меті – відшукати та замовити раніше вподобані речі. Якщо товар відсутній, користувач може активувати сповіщення, яке надійде, коли товар знову стане доступним.

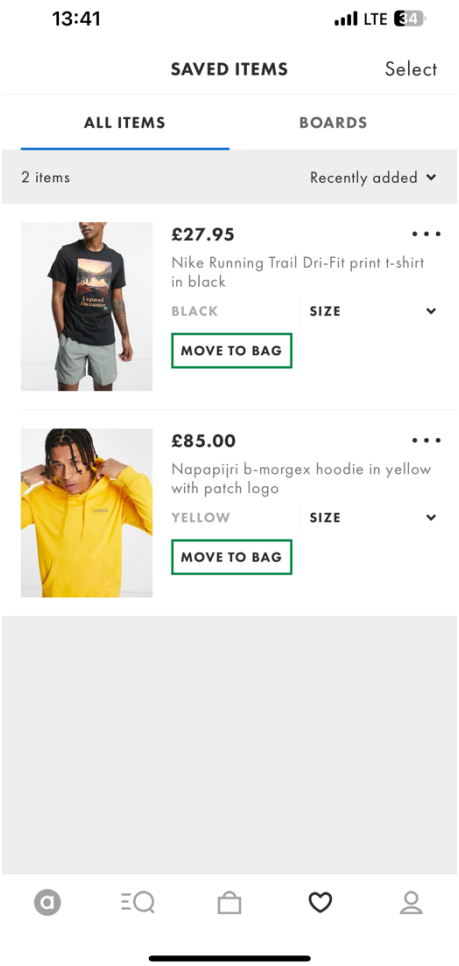


Рисунок 3.2.12 Додаток Asos

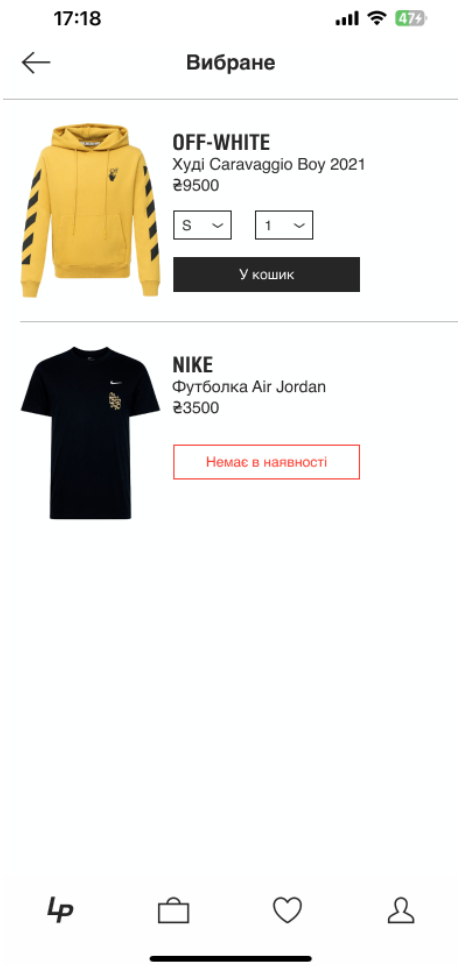


Рисунок 3.2.13 Мій додаток

Екран замовлень користувача розроблено на основі дизайну магазину Asos. Відразу видно товари у замовленні, статус та дату доставки. За потреби можна перейти до окремого замовлення для отримання додаткових деталей.

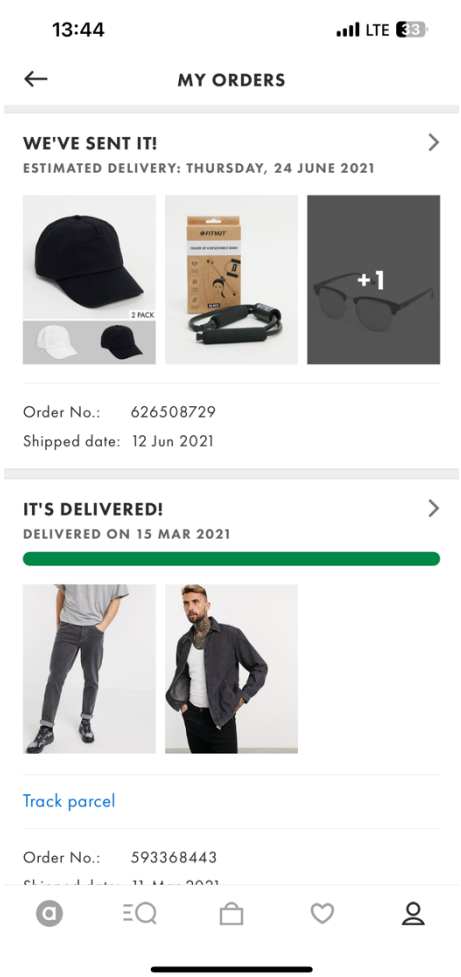


Рисунок 3.2.14 Додаток Asos

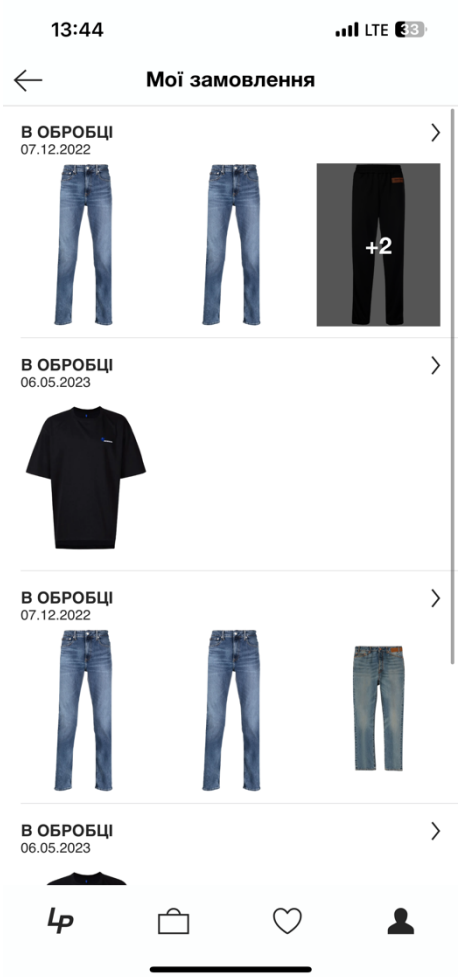


Рисунок 3.2.15 Мій додаток

3.3 Порівняння з функціоналом конкурентів

Asos та Farfetch – міжнародні онлайн-ритейлери модного одягу, взуття та аксесуарів. Spazio – український ритейлер.

Функціонал	Asos	Farfetch	Spazio	Мій додаток
Більшість брендів	+	+	+	+
Фільтри пошуку	+	+	+	+
Розширений пошук	+	+	-	+
Відгуки користувачів	+	+	-	-

Рекомендації	+	+	-	+
Збережені списки бажань	+	+	+	+
Отримання сповіщень	+	+	+	+
Швидке замовлення	+	+	+	+
Промокоди	+	+	-	+
Віртуальна примірочна	-	+	-	-
Онлайн-підтримка	+	+	-	*
Багатомовність	+	+	-	*

* – планується у найближчих оновленнях додатка.

Після аналізу порівняння можна зробити висновок, що додаток не поступається за базовим функціоналом світовим магазинам, що свідчить про його конкурентоспроможність навіть на міжнародному рівні. У порівнянні з українським додатком магазина Spazio, додаток надає значно ширший функціонал, який позитивно впливає на можливості та враження від шопінгу користувачів.

3.4 Опис дизайну інтерфейсу додатка

Дизайн має велике значення для комерційного успіху будь-якого мобільного додатка, і може бути розділений на дві частини: UI та UX.
UX (User Experience, досвід користувача) відіграє роль у взаємодії користувача з додатком, визначаючи, наскільки інтуїтивним та зручним є процес. Він також відповідає за легкість навігації та виконання необхідних дій. Досвід користувача залежить від взаємодії з елементами інтерфейсу.

UI (User Interface, інтерфейс користувача) відповідає за візуальний аспект інтерфейсу. Це включає оформлення кнопок, читабельність тексту, зображень та інших компонентів, з якими користувач взаємодіє.

Під час розробки UX та UI-частин додатка основною метою було створити простий та зрозумілий дизайн, який не відволікав би від покупок і був би легким у використанні.

У додатку використовується одна гарнітура шрифтів – Helvetica, зокрема Helvetica Regular та Helvetica Condensed. Другий варіант шрифту використовується для назв брендів товарів та на головному екрані для вказівки категорій товарів.

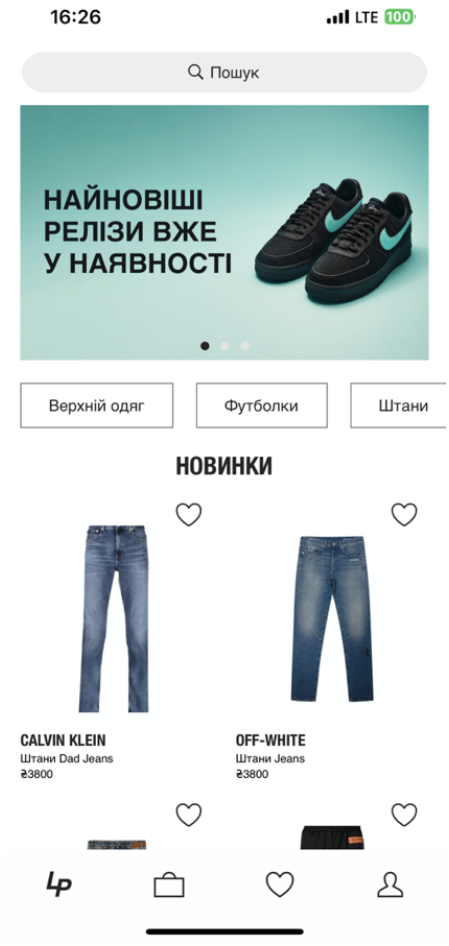


Рисунок 3.4.1 Головний екран додатка

Додаток розроблений для використання у світлій та темній темах. Кольори світлої теми:

- Чорний (#262626) – колір основної частини тексту;
- Білий (#FEFFFF) – колір фону;
- Світло сірий (#E3E3E3) – колір допоміжних елементів;
- Темно сірий (#7F7F7F) – другорядний колір тексту;
- Червоний (#FC3C2F) – колір для звернення уваги.

  Для темної теми використовуються ці ж самі кольори, але реверсивно.

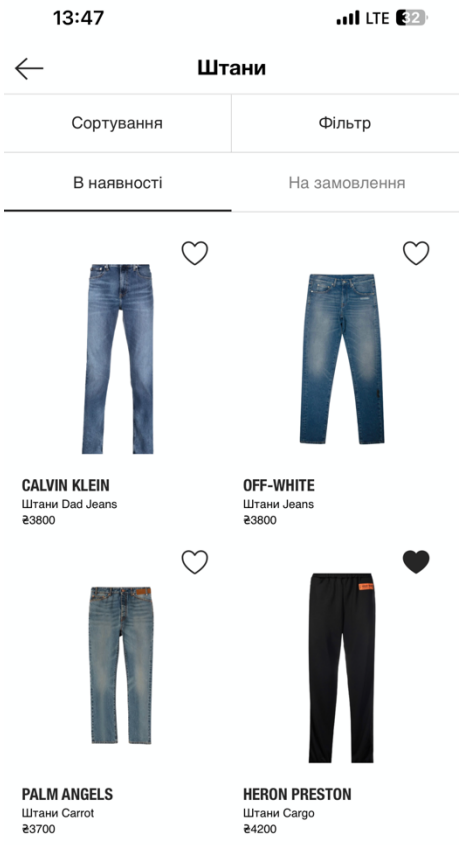


Рисунок 3.4.2 Світла тема

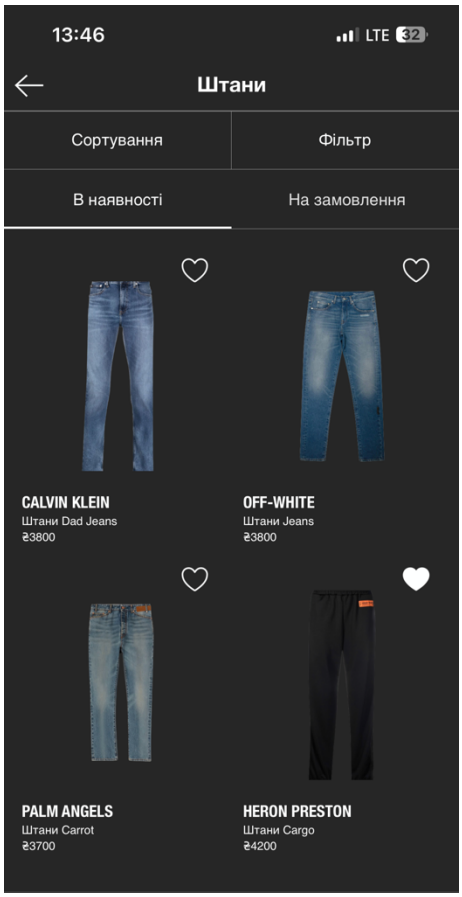


Рисунок 3.4.3 Темна тема

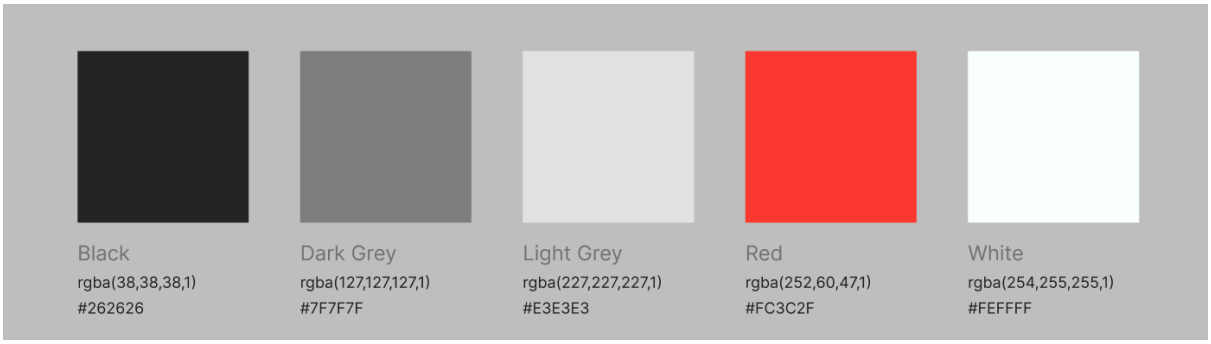


Рисунок 3.4.4 Кольори додатка

Найцікавішими рішенням в UX-частині програми є верхнє меню на екрані для покупок. Завдяки мінімалістичному та легкому дизайну, весь функціонал сортування, фільтрації товарів та вибір між двома типами одягу забезпечується. У першому ряду меню розташовані сортування та фільтрація, що допомагає користувачеві зрозуміти, що ці дії пов'язані між собою і можуть знадобитися йому на першому місці. Якщо фільтр та сортування не потрібні користувачеві, у меню є перемикач між речами у наявності й речами на замовлення. Він створений у подібному стилі, але основна відмінність полягає в тому, що активним може бути лише один варіант. Активний варіант виділяється за допомогою чорного кольору тексту та лінії під ним. Неактивний варіант має сірий колір та тонку лінію під текстом. Саме таким способом, використовуючи комбінацію спільного стилю, розташування елементів за важливістю та дизайн кнопок різних типів, можна створити гармонійний UX-елемент, який не заплутає користувача і буде відразу зрозумілим.

Таким чином, за допомогою добре продуманої структури меню та ефективного використання кольорів, додаток надає користувачам зручний та інтуїтивно зрозумілий інтерфейс для шопінгу. Це дозволяє користувачам легко знаходити товари, що відповідають їхнім потребам та

вподобанням, забезпечуючи позитивний досвід користувача і сприяючи комерційному успіху мобільного застосунку.

РОЗДІЛ 4 Тестування користувальницького інтерфейсу

4.1 Перевірка доступності додатка

Під час розробки додатка було зосереджено особливу увагу на доступності інтерфейсу та гармонійному поєднанні кольорів. Для перевірки кольорової схеми використовувався симулятор дальтонізму [9]. Рисунок 4.1.1 демонструє порівняння оригінального інтерфейсу з інтерфейсами, сприйманими людьми з протономальністю та монохромністю. Як видно з порівняння, здатність взаємодії з інтерфейсом не втрачається навіть при відображенні лише чорно-білих кольорів завдяки вдалому дизайну елементів.

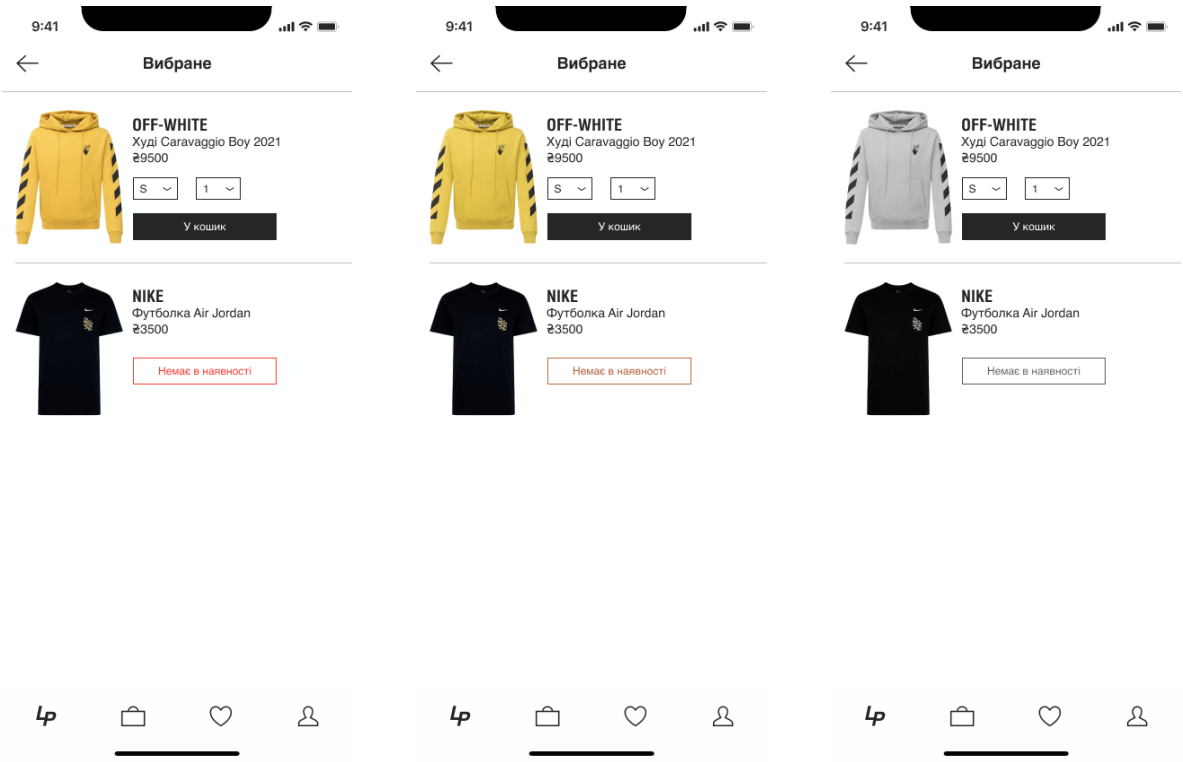


Рисунок 4.1.1 Симуляції дальтонізму

Усі елементи інтерфейсу коректно відображаються на актуальних моделях iPhone.

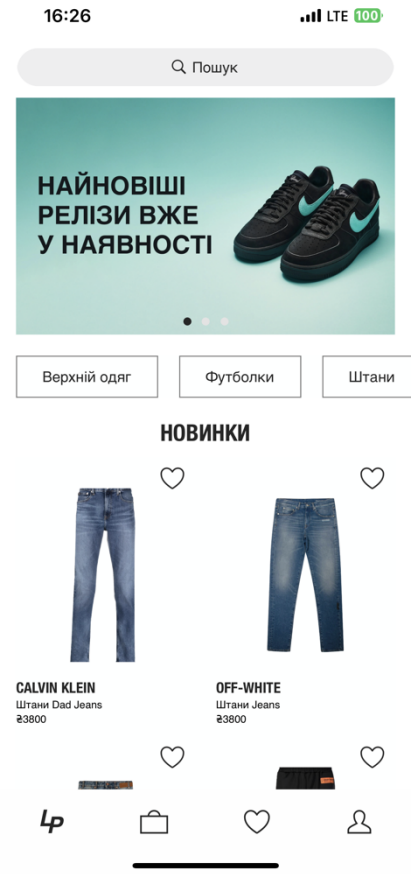


Рисунок 4.1.2 iPhone 13

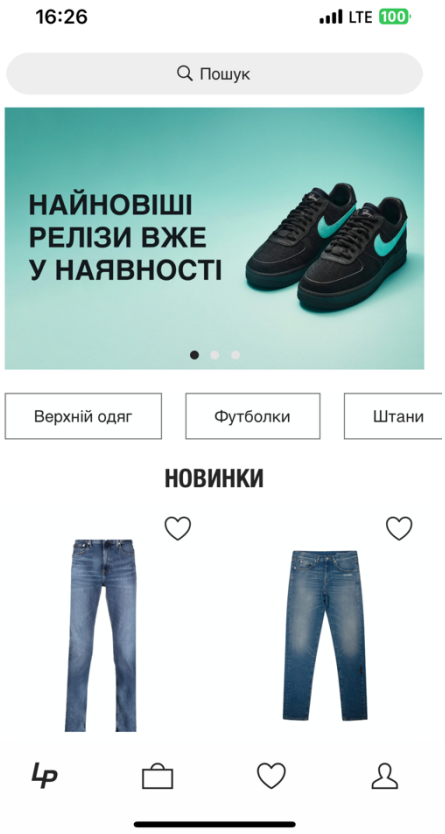


Рисунок 4.1.3 iPhone 8

Використовуючи сервіс expoze.io, було створено теплові карти уваги користувача. Цей сервіс застосовує штучний інтелект для визначення елементів, на які користувач звертає увагу спочатку. Така технологія дозволяє аналізувати пріоритети користувача під час використання кожного екрана додатка.

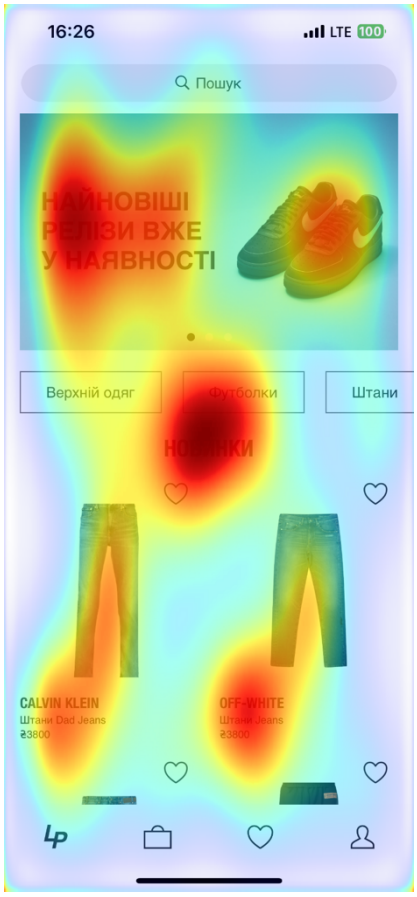


Рисунок 4.1.4 Головний екран

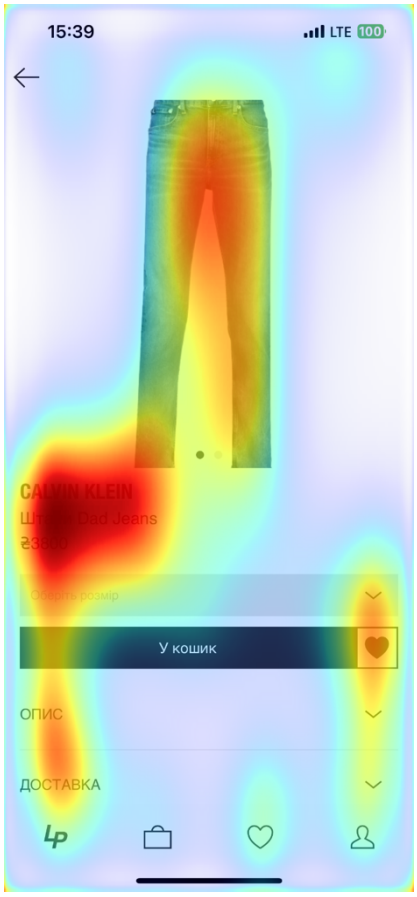


Рисунок 4.1.5 Екран шопінгу

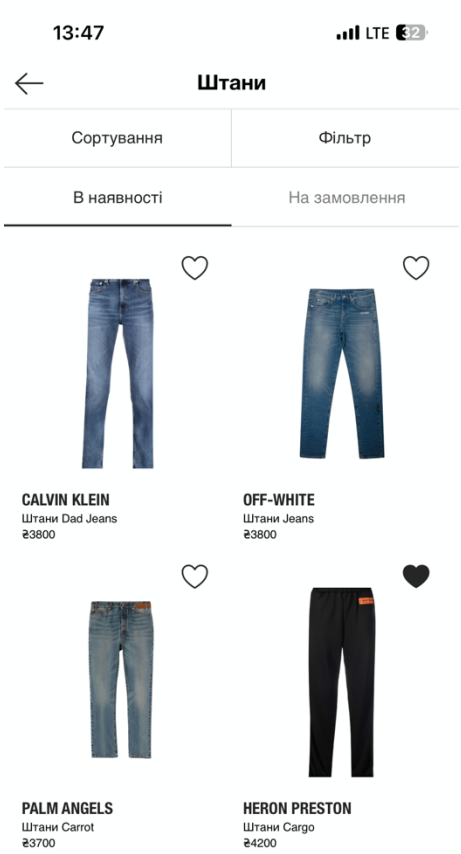


Рисунок 4.1.6 Екран шопінгу

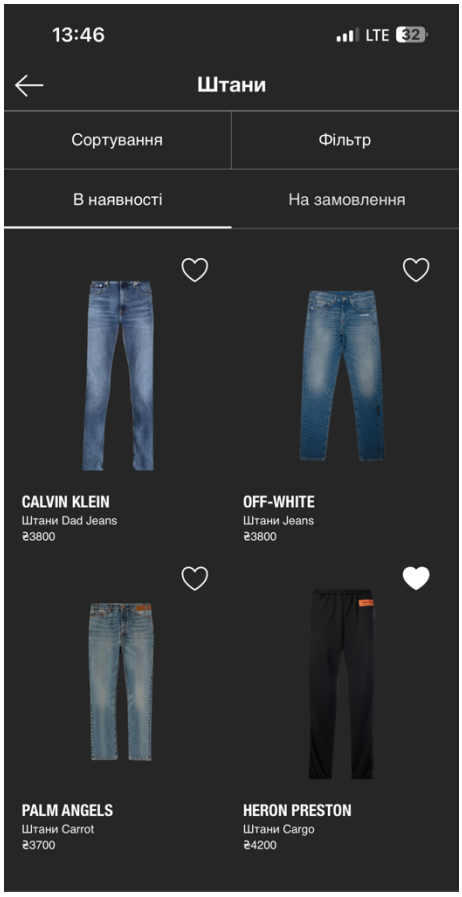


Рисунок 4.1.7 Екран речі

На прикладах видно, що всі ключові елементи розміщені в найбільш активних зонах уваги, що свідчить про добре розроблену логіку та сприйняття додатка.

4.2 Тестування додатка на вибірці користувачів

Для тестування додатка було завантажено 50 одиниць товарів різних категорій, доступних на складі, та 30 одиниць товарів на замовлення. Процес тестування проводився на таких пристроях: iPhone 8, iPhone X, iPhone 12 mini, iPhone 13, iPhone 14 Pro.

Тест	iPhone 8	iPhone X	iPhone 12 mini	iPhone 13	iPhone 14 Pro
Можливість реєструватися в додатку з телефонами різних операторів	+	+	+	+	+
Зручність використання інтерфейсу	+	+	+	+	+
Комбінація кольорів у світлій та темній версіях	+	+	+	+	+
Коректність розробки логічної структури додатка	+	+	+	+	+
Ефективність в роботі	+	+	+	+	+
Фільтрування одягу за всіма необхідними характеристиками	-	+	+	+	-
Легкість оформлення та здійснення оплати	+	+	+	+	+
Здатність відшукати всю важливу інформацію про замовлення	+	+	+	+	+
Якість сервісу технічної допомоги	+	-	-	-	+

Обрано групу осіб, які регулярно купують брендові товари в інтернеті та офлайн. Після обговорення всіх аспектів, зокрема, відзначались пропозиції щодо покращення фільтрації товарів (необхідність додавання більшої кількості опцій) та технічної підтримки (можливість зв'язку з адміністрацією магазину тільки через месенджер, а не в додатку). Ці пропозиції будуть враховані в майбутніх версіях мобільного додатка. Усі пристрої для тестування продемонстрували правильну роботу додатка, всі елементи відображалися коректно. Користувачам сподобалася кольорова схема як для світлої, так і для темної теми.

РОЗДІЛ 5 Розробка програмної частини додатка використовуючи мову програмування Swift

5.1 Обґрунтування вибору засобів розробки (Swift)

Для розробки додатка було обрано мову програмування Swift. Swift - це мова програмування з різними підходами, яку розробила Apple для створення застосунків на своїх операційних системах, таких як iOS, iPadOS, macOS, watchOS та tvOS. Мова була представлена на WWDC [10] у 2014 році й замінила мову програмування Objective-C. Swift використовує найновіші досягнення в галузі мов програмування, а також десятилітній досвід створення платформ Apple. Ця мова була розроблена для швидкої роботи, використовуючи високоефективну технологію компілятора LLVM. Swift є нащадком мов C та Objective-C, включаючи примітиви низького рівня, такі як типи, потоки та оператори, а також об'єктно-орієнтовані функції, наприклад класи, протоколи та загальні шаблони, що забезпечують продуктивність та потужність для розробників Cocoa та Cocoa Touch. Swift є однією з найпопулярніших мов програмування завдяки своєму прямому зв'язку з Apple та її застосунками,

які стають все більш популярними з кожним роком. Згідно з даними Statista [11], у першому кварталі 2022 року в App Store було доступно 2,11 мільйона мобільних додатків.

Мова Swift має два основних фреймворки: UIKit та SwiftUI. UIKit - це фреймворк, керований подіями, який вимагає обробки змін станів, наприклад під час натискання кнопок. Головним недоліком такого підходу є ускладнення синхронізації інтерфейсу користувача зі станом додатка. З іншого боку, SwiftUI - декларативний фреймворк, представлений Apple у 2019 році. Декларативність полягає в тому, що розробник повинен зазначити, чого він бажає досягти, і фреймворк сам вирішує, як це реалізувати. Фреймворк знає найкращий спосіб відображення користувацького інтерфейсу.

Я вирішив обрати фреймворк UIKit для розробки, оскільки мав попередній досвід роботи з ним і знав про його можливості. Крім того, для UIKit доступні більше гайдів та додаткових бібліотек.

5.2 Опис структури даних (Swift)

В основному для опису структури даних додатка використовувалися Enum та Structure.

Enum - це тип даних, що використовується для перерахування набору варіантів або значень.

- LanguageEnum: містить можливі значення для мови.
- CurrencyEnum: містить можливі значення для валюти.
- SizeEnum: містить можливі значення для розмірної сітки.

Structure – це тип даних, що дозволяє створювати власні комплексні типи, що можуть містити властивості та методи. Структури використовуються для представлення даних, які можуть бути складними й містити кілька різних значень.

- Product: містить дані про товар, включаючи назву, бренд, зображення, деталі, посилання на документ та ідентифікатор товару.
- ProductBrand: включає дані про бренд, такі як назву, сезон та посилання на документ.
- ProductDetails: містить інформацію про характеристики товару, такі як колір, матеріал, код стилю, тип, стать, розміри тощо, а також метод sortSizes для сортування розмірів.
- UserProduct: включає дані про товар користувача, такі як посилання на документ, розмір, кількість, наявність у вибраному списку, інформацію про товар та ідентифікатор товару в кошику.
- User: містить дані про користувача, включаючи дату створення облікового запису, налаштування, додаткові дані та контактну інформацію.
- ContactInfo: включає ім'я, прізвище, по батькові, телефон, адресу відділення Нової пошти або дані про кур'єра.
- UserSettings: структура, що містить налаштування користувача, такі як мова, валюта та розмір.
- UserAdditionalInfo: включає ім'я, стать, дату народження та улюблений бренд.
- NpPostalOffice: включає дані про відділення Нової пошти, такі як місто та назву відділення.
- NpCourier: містить інформацію про кур'єра Нової пошти, таку як місто, вулицю, номер будинку та квартири.

- Order: містить дані про замовлення, включаючи список товарів, контактну інформацію, коментар, промокод, вартість одягу, вартість доставки, знижку за промокодом, загальну вартість, статус замовлення та дату створення.
- OrderHistory: містить дані про історію замовлень користувача.
- OrderStatusTypes: включає статуси замовлень, такі як "в обробці", "у дорозі" та "доставлено".
- Size: структура, що містить дані про розмір, ціну та кількість.
- Sizechart: містить дані про розміри за замовчуванням та розміри для різних країн та брендів одягу.

5.3 Реалізація інтерфейсу додатка

Візуальний дизайн інтерфейсу програми складається з таких класів:

• Основний storyboard [12] Main, розроблений за допомогою Interface Builder, який містить всі екрани програми та переходи між ними;

Рисунок 5.3.1 Екрани у Main storyboard

• CustomHeaderView, що використовується для створення меню з детальною інформацією на екрані предмета.

Рисунок 5.3.2 Клас CustomHeaderView

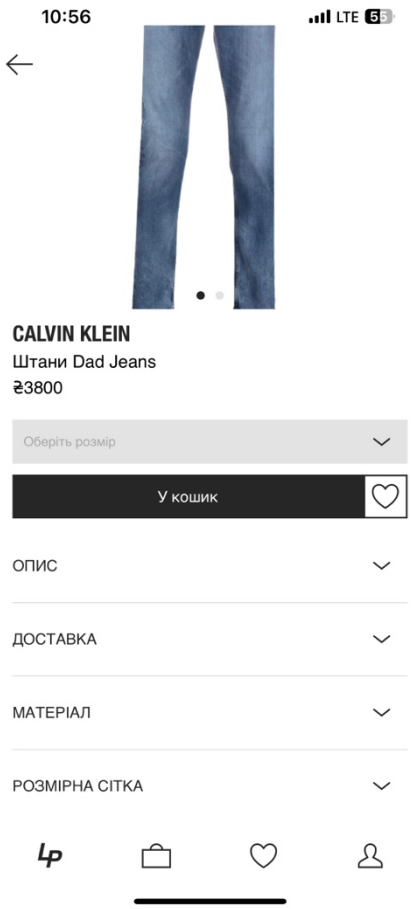


Рисунок 5.3.3 Реалізація CustomHeaderView у додатку

• SizeChartTableViewCell, TwoColumnsTableViewCell, призначені для відображення розмірної сітки [13].

Рисунок 5.3.4 Клас TwoColumnsTableViewCell

Рисунок 5.3.5 Клас SizeChartTableViewCell

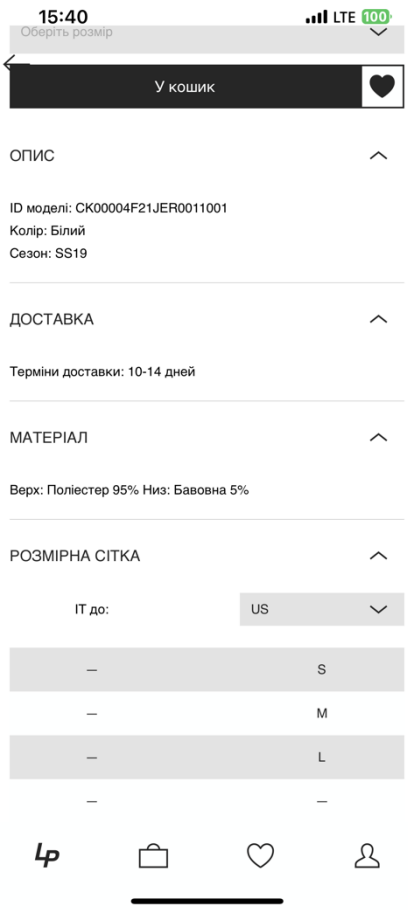
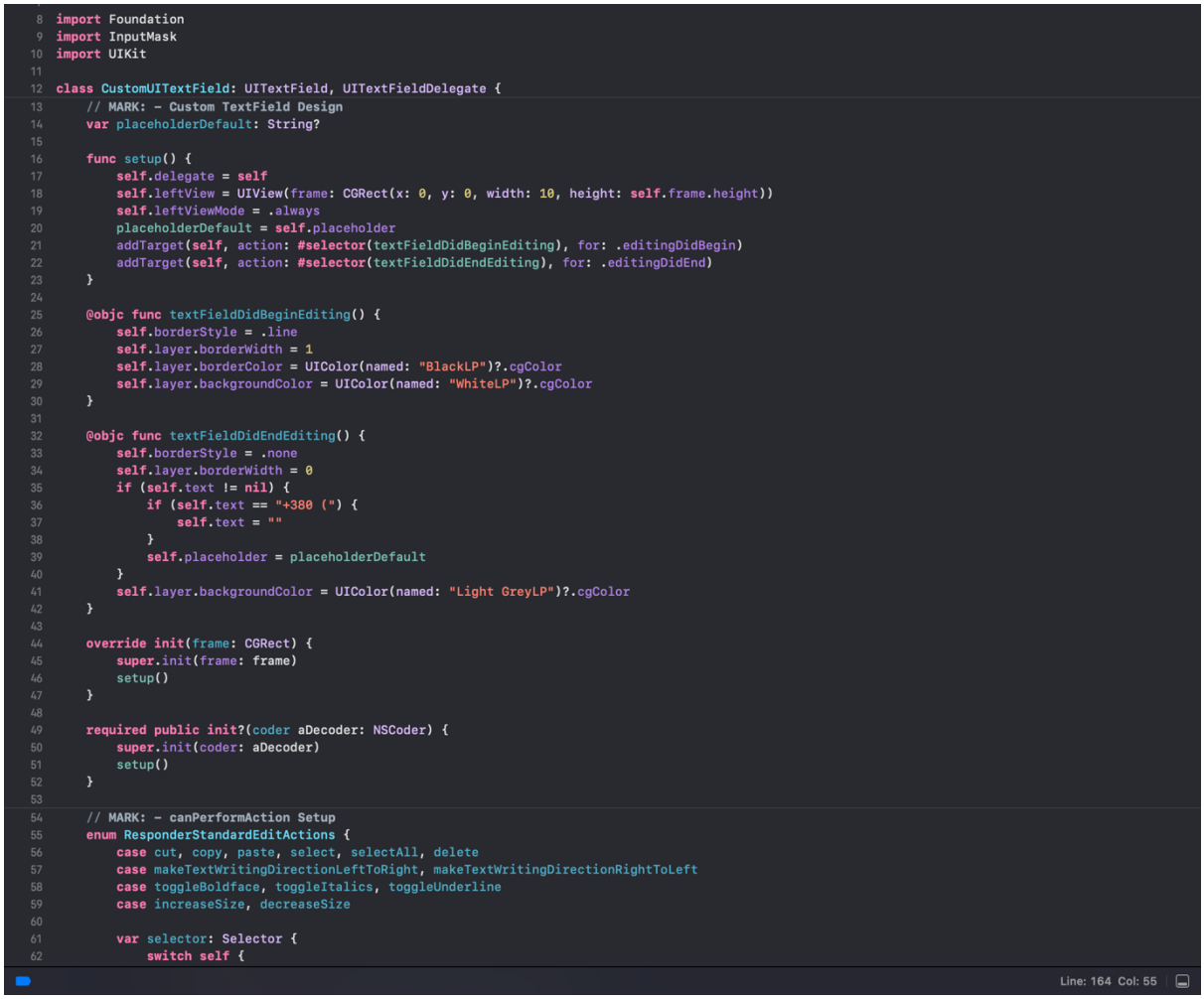


Рисунок 5.3.6 Реалізація класів у додатку

Програмний компонент кастомного дизайну містить такі класи:

• CustomUITextField для налаштування функціонала всіх текстових полів та перевірки введених даних. У цьому класі також налаштовується UIPickerView для вибору дати, розміру предмета та інших даних.



```
import Foundation
import InputMask
import UIKit

class CustomUITextField: UITextField, UITextFieldDelegate {
    // MARK: - Custom TextField Design
    var placeholderDefault: String?

    func setup() {
        self.delegate = self
        self.leftView = UIView(frame: CGRect(x: 0, y: 0, width: 10, height: self.frame.height))
        self.leftViewMode = .always
        placeholderDefault = self.placeholder
        addTarget(self, action: #selector(textFieldDidBeginEditing), for: .editingDidBegin)
        addTarget(self, action: #selector(textFieldDidEndEditing), for: .editingDidEnd)
    }

    @objc func textFieldDidBeginEditing() {
        self.borderStyle = .line
        self.layer.borderWidth = 1
        self.layer.borderColor = UIColor(named: "BlackLP")?.cgColor
        self.layer.backgroundColor = UIColor(named: "WhiteLP")?.cgColor
    }

    @objc func textFieldDidEndEditing() {
        self.borderStyle = .none
        self.layer.borderWidth = 0
        if (self.text != nil) {
            if (self.text == "+380 (") {
                self.text = ""
            }
            self.placeholder = placeholderDefault
        }
        self.layer.backgroundColor = UIColor(named: "Light GreyLP")?.cgColor
    }

    override init(frame: CGRect) {
        super.init(frame: frame)
        setup()
    }

    required public init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        setup()
    }

    // MARK: - canPerformAction Setup
    enum ResponderStandardEditActions {
        case cut, copy, paste, select, selectAll, delete
        case makeTextWritingDirectionLeftToRight, makeTextWritingDirectionRightToLeft
        case toggleBoldface, toggleItalics, toggleUnderline
        case increaseSize, decreaseSize

        var selector: Selector {
            switch self {
```

Line: 164 Col: 55

Рисунок 5.3.7 Клас CustomUITextField

Додатково, для анімації кнопок була використана бібліотека WMSegmentControl. Для дизайну поля одноразового SMS-коду для реєстрації використана бібліотека AEOTPTextField [13].

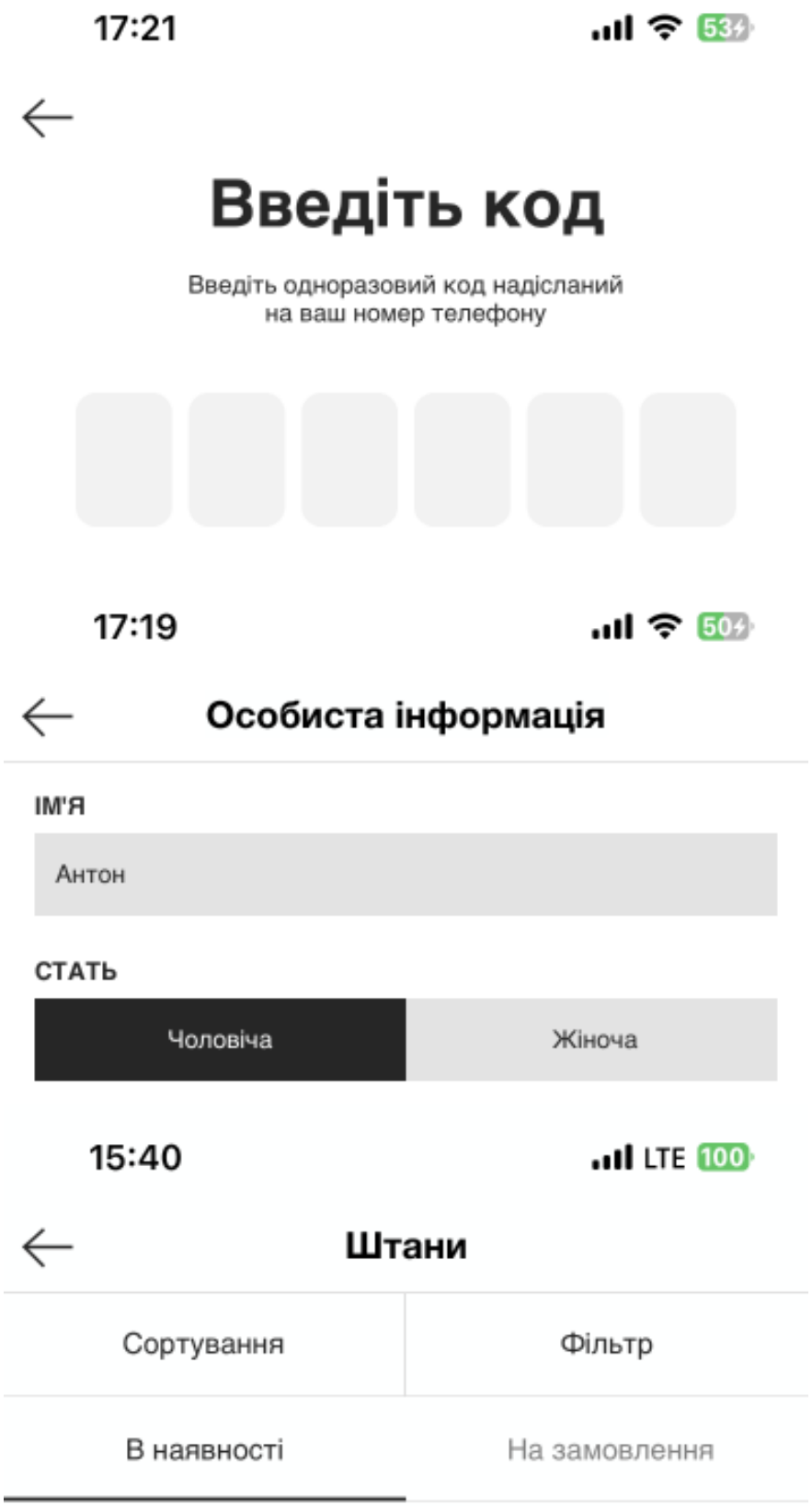


Рисунок 5.3.8 Елементи інтерфейсу створені за допомогою WMSegment та AEOTPTextField

5.4 Опис розробки коду додатка

В розробленій частині програми використовується 6 storyboards та 41 класи.

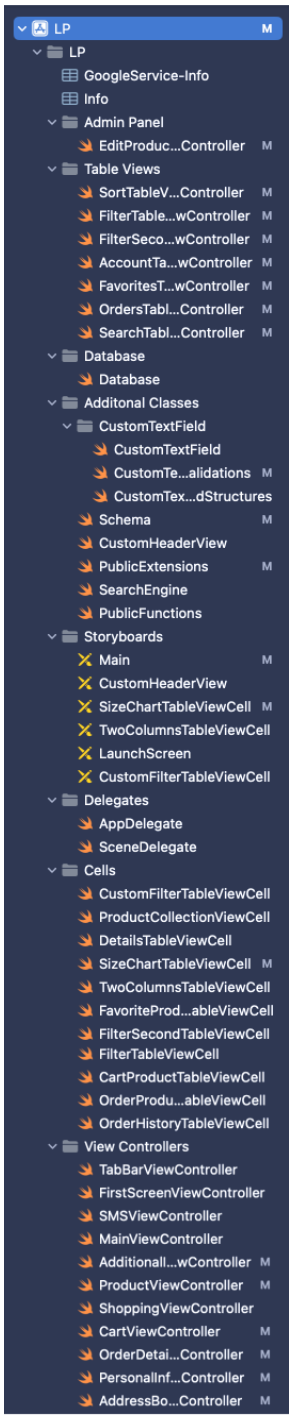


Рисунок 5.4.1 Класи додатка

З'єднання та синхронізація з базою даних Firestore здійснюється у класі Database. Функція fetchData отримує дані про товари з відповідної

категорії, а за допомогою функції build дані у форматі JSON записуються в структуру StockProduct та додаються до масиву предметів, які повертаються в ViewController. Схожим чином працює функція getSizechart, вона використовується для запису інформації у структуру Sizechart. Оновлення даних відбувається в реальному часі, при зміні будь-якого поля у базі даних, воно відразу змінюється й у програмі.

```
46      func getUserDetails(handler: @escaping (User) -> Void) {
47          let docRef = db.collection("users").document(userID)
48          docRef.getDocument { documentSnapshot, err in
49              guard let data = documentSnapshot else {
50                  return
51              }
52              handler(User.build(from: data))
53          }
54      }
55
56      func editUserDetails(userDetailsType: userDetailsTypes, userData: [String: Any]) {
57          let docRef = db.collection("users").document(userID)
58          docRef.setData(["\(userDetailsType)": userData], merge: true)
59      }
60
61      func getProducts(availabilityCollection: availabilityCollectionTypes, productCollection: productCollectionTypes, handler: @escaping
            ([UserProduct]) -> Void) {
62          let docRef = db.collection("men").document("\(availabilityCollection)").collection("\(productCollection)")
63          docRef.addSnapshotListener { querySnapshot, err in
64              guard let data = querySnapshot?.documents else {
65                  return
66              }
67              handler(UserProduct.build(from: data, and: []))
68          }
69      }
70
71      func getProductsWithFavorites(availabilityCollection: availabilityCollectionTypes, productCollection: productCollectionTypes, handler: @escaping
            ([UserProduct]) -> Void) {
72          let docRef = db.collection("men").document("\(availabilityCollection)").collection("\(productCollection)")
73          docRef.addSnapshotListener { querySnapshot, err in
74              guard let data = querySnapshot?.documents else {
75                  return
76              }
77              if self.anonymousUser == false {
78                  self.getUserProducts(collection: .favorites) { favorites in
79                      handler(UserProduct.build(from: data, and: favorites))
80                  }
81              } else {
82                  handler(UserProduct.build(from: data, and: []))
83              }
84          }
85      }
86
87      func getFavoriteProductsForSearch(searchProducts: [[UserProduct]], handler: @escaping ([[UserProduct]]) -> Void) {
88          if self.anonymousUser == false {
89              getUserProducts(collection: .favorites) { favorites in
90                  handler(UserProduct.buildSearch(from: searchProducts, and: favorites))
91              }
92          } else {
93              handler(searchProducts)
94          }
95      }
```

Рисунок 5.4.2 Клас Database

Клас Schema містить усі структури, наприклад, public enum FilterTypes, який використовується у FilterTableViewController для фільтрації предметів (див. додаток В).

Всі інші класи асоціюються з екранами у Main storyboard та відповідають за їхню логіку та налаштування. Наразі найскладнішими у реалізації є екрани FilterTableViewController та FilterSecondTableViewController. Через

необхідність фільтрувати предмети за кількома параметрами, заданими користувачем, потрібно було розробити відповідну логіку та якісний інтерфейс для користувача. Ці фільтри співставні за функціоналом з такими програмами, як Farfetch або Asos, над якими працювали цілі команди розробників. (див. додатки Г, Д) [13].

РОЗДІЛ 6 Тестування готового додатка

6.1 Опис процесу тестування додатка на реальних клієнтах

Для цього тестування був створений телеграм чат з 5 постійними клієнтами магазину, яким був наданий доступ до останньої версії мобільного застосунку. На тестування було виділено 20 хвилин і прописані сценарії використання: швидко зайти та замовити річ, пошук товарів по фільтрах та сортуванню, а також замовлення раніше вподобаних предметів.

У результаті тестування для всіх сценаріїв було встановлено успішність процесу, але було виявлено одну проблему. Зокрема, виявлено, що при зміні категорії товарів (наявність або під замовлення), налаштування фільтрів скидувалися, і користувачам доводилось знову налаштовувати їх.

РОЗДІЛ 7 Планування майбутнього додаткового функціонала та стратегія розвитку

7.1 Опис додаткового функціонала додатка

Важливою частиною розробки додатка є планування його майбутнього функціоналу, щоб з самого початку побудувати структуру та вже наявні функції з можливістю їх розширення.

У якості додаткового функціонала, який буде впроваджуватися у додаток поступово, будуть наступні функції:

- Інтеграція відправки замовлень з API Нової пошти. Після підтвердження наявності товару, буде автоматично створюватись відправлення з даними покупця (ПІП, номер телефона тощо). І одразу після успішного створення, відображати ТТН (номер відправлення Нової пошти) в історії замовлень клієнта і також в адмін панелі з усіма замовленнями. Це значно пришвидшить та полегшить роботу й допоможе уникнути помилок при створенні відправлення власноруч.
- Створення чату для технічної підтримки користувачів прямо у застосунку. Це допоможе швидше розв'язувати будь-які питання та під час спілкування завжди мати усю поточну інформацію про клієнта і його замовлення.
- Рекомендації товару на основі шуканих або обраних товарів. Рекомендації будуть побудовані на ключових словах, які притаманні певній категорії одягу. Даний функціонал допоможе користувачу переглядати більше речей, які можливо йому підходять і також покращить конверсію замовлень для самого магазину.

7.2 Розробка стратегії розвитку

Розробка стратегії розвитку та просування додатка містить в собі визначення способів залучення нових користувачів та збільшення його популярності на ринку. Оскільки на цей час у магазина, для якого створювався додаток, є тільки Instagram сторінка, це створює додаткові виклики для просування додатка.

Одним зі способів залучення нових користувачів є використання інструментів соціальних мереж, таких як реклама та просування на популярних платформах, наприклад Instagram, Facebook, TikTok тощо. Реклама на соціальних мережах може бути ефективною, оскільки вона дозволяє дійти до широкої аудиторії та залучити нових користувачів. Для цього можна використовувати цільову рекламу, яка дозволяє налаштувати рекламу на конкретну аудиторію з певними інтересами, демографічними характеристиками тощо.

Крім того, можна співпрацювати з іншими брендами або інфлюенсерами, які популярні у вашій цільовій аудиторії. Наприклад, співпраця з інфлюенсерами, які вже мають свою аудиторію, може допомогти привернути нових користувачів та збільшити популярність додатка.

Крім цього, важливо проводити аналіз конкурентів та вивчати їхні підходи до просування та реклами. Це дозволить розробити власну стратегію диференціації та підвищення конкурентних переваг на ринку.

Отже, стратегія розвитку та просування додатка має включати широкий спектр дій, що дозволяють залучити нових користувачів та збільшити його популярність на ринку. Необхідно використовувати різноманітні інструменти соціальних мереж, такі як реклама та співпраця з інфлюенсерами, проводити аналіз конкурентів та розробляти стратегію диференціації. Крім того, важливо створювати привабливі пропозиції та привілеї для користувачів додатка, що дозволяє зберегти і налагодити стосунки з вже існуючою аудиторією та залучити нових користувачів. Всі ці дії дозволять збільшити конкурентоспроможність додатка на ринку та забезпечити його успішний розвиток.

Висновки

8.1 Підсумок дослідження

Дослідження показало, що розробка мобільних додатків є актуальною темою в сучасному світі технологій. За останні роки кількість користувачів смартфонів та планшетів зросла значно, тому мобільні додатки стали важливим інструментом для залучення нових користувачів та збереження вже наявної аудиторії.

У ході дослідження було виявлено, що вибір мови програмування та бази даних є важливими етапами розробки додатка. Обрання мови Swift та бази даних Firebase для реалізації додатка було обґрунтовано на основі їх ефективності, швидкості та простоти використання.

Окрім того, дослідження в галузі дизайну інтерфейсу користувача дозволило розробити дизайн додатка, який відповідає потребам користувачів та покращує їх взаємодію з додатком. Важливою складовою дизайну є візуальна привабливість та зручність використання.

Отже, підсумовуючи, дослідження у сфері розробки мобільних додатків та вибору технологій дозволило обрати оптимальний шлях для реалізації додатка, що відповідає потребам користувачів та покращує їх взаємодію з додатком.

8.2 Основні результати та висновки з розробки додатка

Під час розробки додатка було успішно реалізовано практично весь запланований функціонал, включаючи відображення історії замовлень, адмін панель для усіх членів команди та оплату використовуючи платіжний метод Apple Pay. Було досягнуто певного рівня інтерактивності

та зручності в користуванні додатком за допомогою розробленого інтерфейсу.

Також, було проведено успішне тестування додатка на різних пристроях та в різних умовах використання, що забезпечує його надійну та стабільну роботу.

Одним з важливих результатів є те, що додаток задовольняє основні потреби користувачів та відповідає сучасним тенденціям у мобільній розробці. Це дозволить йому успішно конкурувати на ринку та залучати нових користувачів.

Результатом успішної розробки додатка є його готовність до використання та задоволення потреб користувачів, що забезпечує його успішний розвиток та популярність на ринку мобільних e-commerce додатків.

8.3 Рекомендації щодо подальшого вдосконалення додатка

Хоча додаток ще не запущений, але вже слід планувати розвиток напрямків, які можуть підвищити його ефективність та забезпечити більш зручне користування для клієнтів.

Одним з напрямків є розширення функціонала додатка. Наприклад, можна додати можливість зберігання кредитних карт користувачів в профілі та автопоповнення балансу, щоб уникнути потреби постійно вводити дані картки. Також можна додати функцію розрахунку промокодів та знижок для користувачів.

Другим напрямком є підвищення ефективності додатка. Наприклад, можна оптимізувати роботу додатка для зниження часу завантаження сторінок та підвищення швидкості реакції на дії користувачів. Також можна розглянути можливість використання кешування для збільшення швидкості роботи додатка.

Третім напрямком є поліпшення інтерфейсу користувача. Наприклад, можна розглянути можливість додати анімації для покращення візуального враження від взаємодії з додатком. Також можна покращити навігацію по додатку, зробивши її більш інтуїтивно зрозумілою для користувачів.

Розроблений додаток має потенціал для подальшого розвитку та покращення, що дозволить підвищити ефективність та забезпечити більш зручне користування для клієнтів. Рекомендації щодо подальшого вдосконалення включають розширення функціонала, підвищення ефективності додатка та поліпшення інтерфейсу користувача. Реалізація цих напрямків дозволить зберегти та залучити нових клієнтів, збільшити конкурентоспроможність додатка на ринку та забезпечити його успішний розвиток.

Використана література

1) https://www.statista.com/statistics/320101/global-smartphone-shipments-by-operating-system/ – статистика Statista

2) https://www.salesforce.com/products/commerce-cloud/resources/mobile-shopping-report/ – статистика Salesforce

3) https://www.shopify.com/enterprise/mobile-commerce-statistics – статистика Shopify

4) https://www.emarketer.com/content/mobile-will-make-up-almost-75-of-time-spent-on-the-internet-in-2021 – статистика eMarketer

5) https://firebase.google.com/ – Firebase

6) https://careerkarma.com/blog/companies-that-use-firebase/ – стаття про Firebase

7) https://firebase.google.com/docs/ios/setup – налаштування Firebase

8) https://uk.wikipedia.org/wiki/SDK – про SDK

9) https://pilestone.com/pages/color-blindness-simulator-1 – симулятор дальтонізму

10) https://uk.wikipedia.org/wiki/Apple_Worldwide_Developers_Conference – WWDC

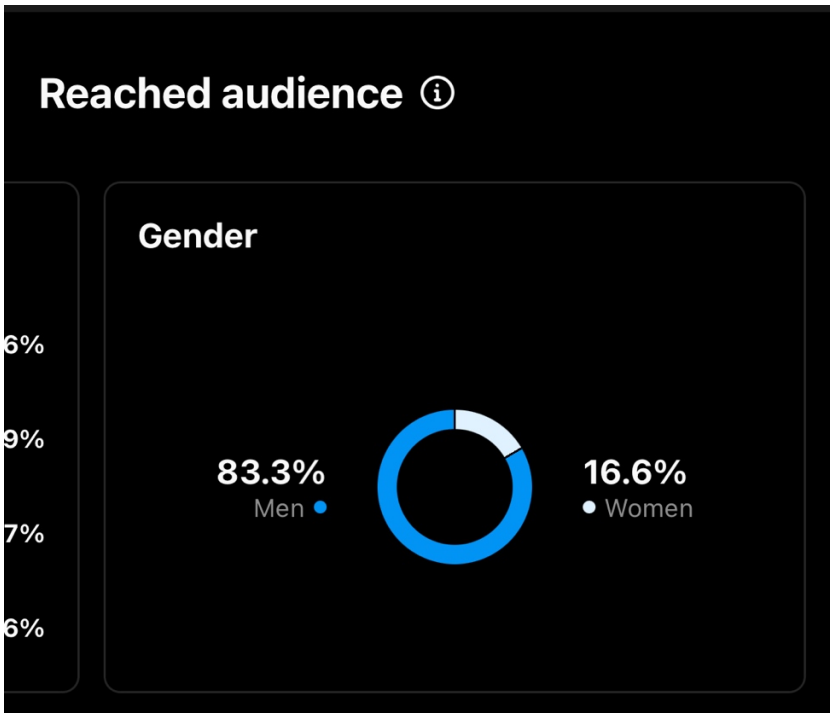
11) https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/ – кількість мобільних додатків доступних в App Store

12) https://www.kodeco.com/5055364-ios-storyboards-getting-started - про Apple storyboards

13) Колесніков, Антон (2022). Особливості дизайну мобільних застосувань написаних з використанням мови Swift. Києво-Могилянська академія.

Додатки

Додаток А. Статистика клієнтів за статтю



Додаток Б. Статистика клієнтів за віком