Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Мобільний додаток для управління гардеробом.Front-end

(тема)

Виконав:
здобувач 4 року навчання
групи ПЗПІ-21-6

Олег МАЛЮТА
(Власне ім'я, ПРІЗВИЩЕ)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Наталя КРАВЕЦЬ
(посада, Власне ім'я, ПРІЗВИЩЕ)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Малюті Олегу Вадимовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Мобільний додаток для управління гардеробом.Front-end
Затверджена наказом по університету від 19.05.2025р. № 397 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 06.06.2025
3. Вихідні дані до роботи Розробити Front-end для серверної частини програмної системи у виді мобільного застосунку для реалізації функціоналу управління гардеробом, зберігання інформації про одяг, створення образів та отримання персоналізованих рекомендацій на платформі Android.
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	17.04.2025	*виконано*
2	Створення специфікації ПЗ	19.04.2025	*виконано*
3	Проектування ПЗ	27.04.2025	*виконано*
4	Розробка ПЗ	15.05.2025	*виконано*
5	Тестування ПЗ	20.05.2025	*виконано*
6	Оформлення пояснювальної записки	23.05.2025	*виконано*
7	Підготовка презентації та доповіді	27.05.2025	*виконано*
8	Оцінка роботи рецензентом, отримання відзиву від керівника кваліфікаційної роботи, попередній захист роботи та проходження нормо-контролю	04.06.2025	*виконано*
9	Здача роботи у електронний архів допуск роботи до захисту завідувачем кафедри	10.06.2025	*виконано*
10	Захист кваліфікаційної роботи	16.06.2025	*виконано*

Дата видачі завдання 17.04.2025

Здобувач _____
(підпис)

Керівник роботи _____ доц. кафедри ПІ Наталя КРАВЕЦЬ
(підпис) (посада, Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра, 98 стор., 25 рис., 13 джерел.

ГАРДЕРОБ, МОБІЛЬНИЙ ДОДАТОК, АВТОМАТИЧНА ГЕНЕРАЦІЯ ОБРАЗІВ, СТАТИСТИКА, ОФЛАЙН-РЕЖИМ, ANDROID, JETPACK COMPOSE, KOTLIN

Об’єкт розробки – Front-end мобільного додатка для управління гардеробом на платформі Android, який дозволяє користувачам каталогізувати одяг, створювати образи та отримувати персоналізовані рекомендації.

Мета розробки – створити зручний інструмент для організації гардеробу з функціями автоматичного підбору комбінацій одягу, аналізу статистики та роботи в офлайн-режимі.

Метод рішення – розробка на мові програмування Kotlin з використанням Android SDK, локальної бази даних Room для офлайн-роботи та REST API для синхронізації з серверною частиною. Для генерації образів застосовано алгоритми на основі вподобань користувача та погодних умов.

У результаті розробки створено мобільний додаток з інтуїтивно зрозумілим інтерфейсом, який дозволяє:

- додавати та редагувати елементи гардеробу з фотографіями та детальними характеристиками;
- генерувати образи вручну або автоматично з урахуванням сезонності, стилю та погоди;
- аналізувати статистику використання одягу за категоріями, матеріалами та сезонами;
- працювати без інтернет-з’єднання (у гостьовому режимі).

ABSTRACT

WARDROBE, MOBILE APPLICATION, AUTOMATIC IMAGE GENERATION, STATISTICS, OFFLINE MODE, ANDROID, JETPACK COMPOSE, KOTLIN

The object of development is a mobile application Front-end for wardrobe management on the Android platform, which allows users to catalog clothes, create looks and receive personalized recommendations.

The goal of the development is to create a convenient tool for organizing a wardrobe with the functions of automatic selection of clothing combinations, statistical analysis and work in offline mode.

The solution method is development in the Kotlin programming language using the Android SDK, the local Room database for offline work and the REST API for synchronization with the server part. Algorithms based on user preferences and weather conditions were used to generate images.

As a result of the development, a mobile application with an intuitive interface was created that allows:

- add and edit wardrobe items with photos and detailed characteristics;
- generate looks manually or automatically, taking into account seasonality, style and weather;
- analyze clothing usage statistics by category, material and season;
- work without an internet connection (in guest mode).

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface

DB – Data Base

HTTP – Hyper-Text Transfer Protocol

MVVM – Model-View-ViewModel

REST – Representational State Transfer

RGB – Red Green Blue

SDK – Software Development Kit

UI – User Interface

ВСТУП

Сучасний ринок мобільних додатків пропонує широкий спектр рішень для організації повсякденного життя, включаючи управління гардеробом. Однак багато з існуючих рішень мають обмежену функціональність, потребують постійного підключення до Інтернету або не забезпечують достатньо гнучких інструментів для аналізу гардеробу. Це обумовлює актуальність розробки нового мобільного додатку, який би поєднував зручність використання, широкий функціонал та можливість роботи в офлайн-режимі.

Метою даної роботи є створення мобільного додатку, призначеного для управління гардеробом, який дозволить користувачам:

- додавати та редагувати елементи одягу з детальним описом (категорія, сезонність, колір, матеріал тощо);
- генерувати комбінації одягу на основі вподобань користувача, погодних умов, призначення та стилю;
- аналізувати статистику використання гардеробу (наприклад, кількість речей за категоріями, середній вік елементів, розподіл за матеріалами);
- працювати з додатком без підключення до Інтернету (у гостьовому режимі).

Основні завдання розробки додатку включають:

- створення інтуїтивного інтерфейсу для зручного управління елементами гардеробу;
- реалізацію розумних алгоритмів автоматичного підбору образів;
- розробку детальної статистики використання гардеробу;
- забезпечення стабільної офлайн-роботи;
- оптимізацію продуктивності для різноманітних пристроїв.

Програмний застосунок ClothesAdvisor призначений для мобільних пристроїв на базі ОС Android, який реалізує вище перелічені функції..

Для реалізації проекту використано сучасні інструменти та технології. Основне середовище розробки - Android Studio, основною мовою програмування

обрано Kotlin. Для локального зберігання даних застосовано Room Database, інтерфейс розроблено за допомогою Jetpack Compose з підтримкою динамічних кольорів Material Design 3. Як архітектурну модель обрано MVVM (Model-View-ViewModel), що забезпечує чіткий поділ логіки та інтерфейсу.

Розроблений додаток знайде широке застосування у різних сферах. Він стане корисним інструментом для персонального використання при організації гардеробу, може бути використаний у консалтингових послугах зі стилю та образу життя. У сфері роздрібної торгівлі одягом він може служити додатковим сервісом для клієнтів, а в індустрії моди - корисним інструментом для професійних стилістів. Також додаток може бути включений до освітніх програм з основ стилю та гардероб-менеджменту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Сьогодні на ринку мобільних додатків існує чимало рішень для управління гардеробом, але більшість з них мають схожі проблеми та обмеження. Популярні додатки, такі як "Stylebook" (рис. 1.1), "Closet Space" (рис. 1.2) та "Smart Closet" (рис. 1.3), пропонують базовий функціонал: додавання фото одягу, створення комплектів і простий аналіз гардеробу. Однак вони часто не враховують індивідуальні потреби користувачів, працюють повільно або вимагають постійного підключення до інтернету.

В інтернеті було знайдено приклади скріншотів інтерфейсу "Stylebook" (рис. 1.1).

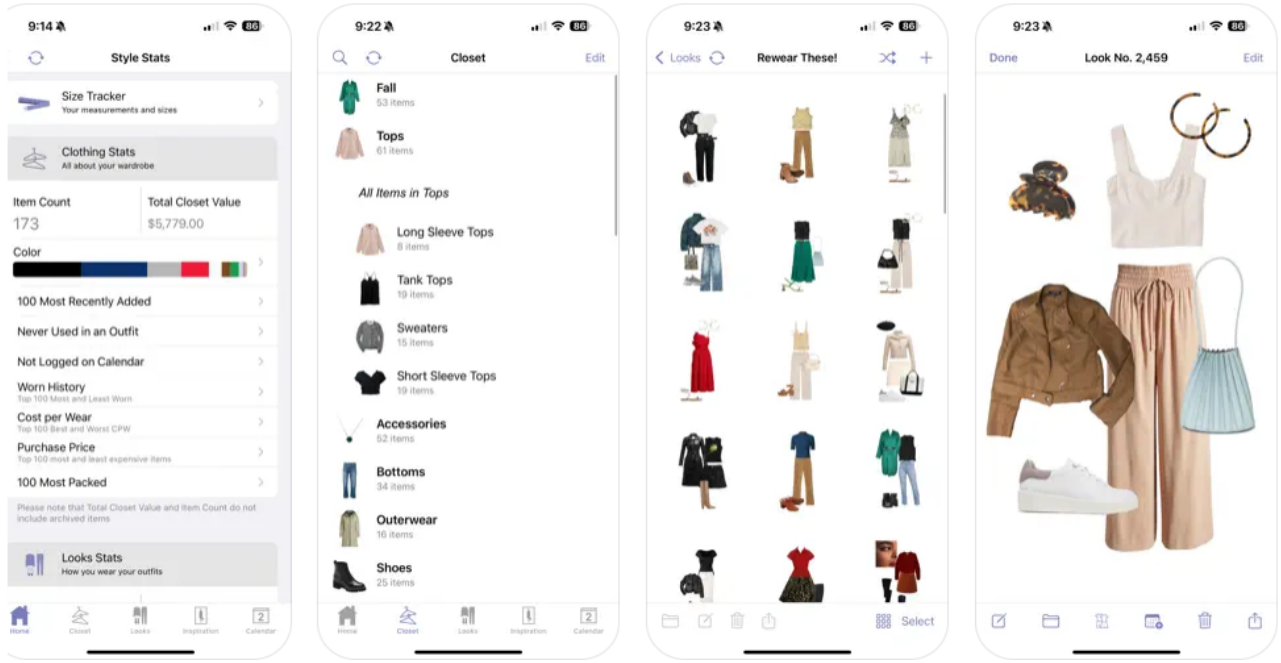

Рисунок 1.1 – Знімки екрану інтерфейсу додатку "Stylebook" ([1])

В інтернеті було знайдено приклади скріншотів інтерфейсу "Closet Space" (рис. 1.2).

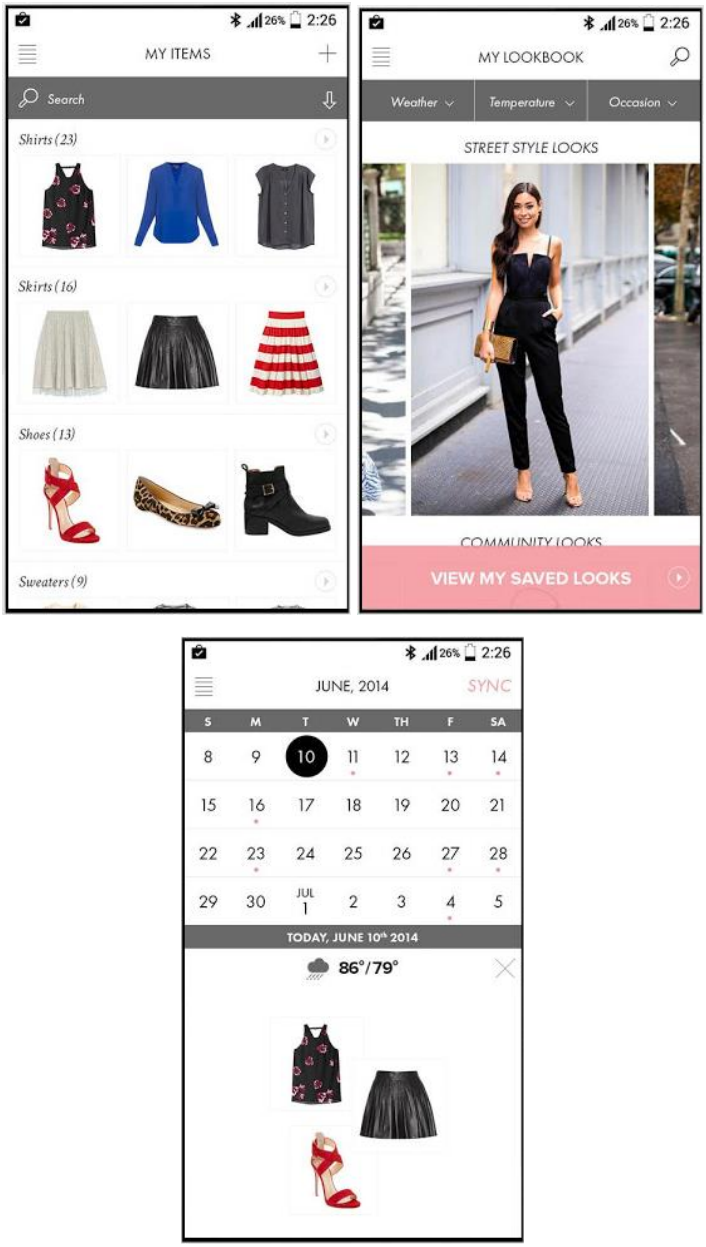

Рисунок 1.2 – Знімки екрану інтерфейсу додатку "Closet Space" ([2])

В інтернеті було знайдено приклади скріншотів інтерфейсу "Smart Closet" (рис. 1.3).

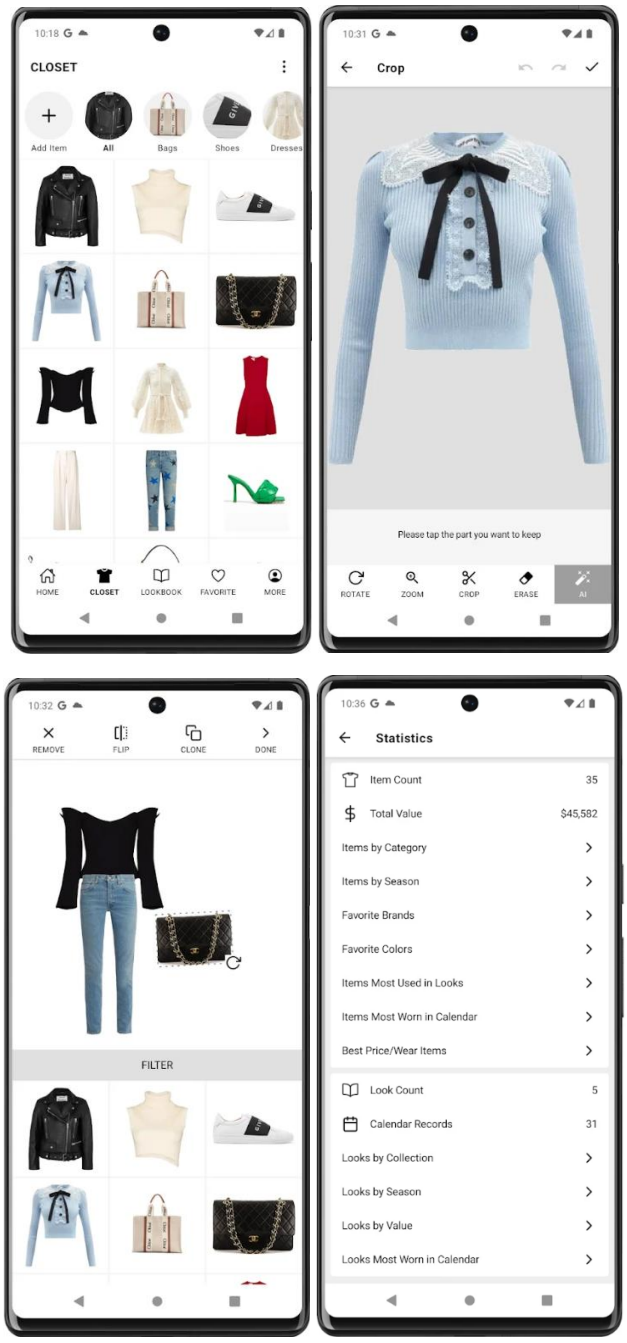

Рисунок 1.3 – Знімки екрану інтерфейсу додатку "Smart Closet" ([3])

Деякі додатки, наприклад "Pureple", намагаються запропонувати інтелектуальні рекомендації, але їх алгоритми обмежені простим підбором за кольором або категорією, без урахування таких важливих факторів, як поточна погода, стиль користувача або особливості матеріалів. Крім того, багато рішень не

мають української локалізації, що ускладнює їх використання для місцевих користувачів.

Сучасні мобільні додатки-органайзери для особистих речей, зокрема для управління гардеробом, базуються на кількох ключових принципах, серед яких центральне місце займає обробка [4] та зберігання фотографій одягу. Ця функціональність є фундаментальною, оскільки саме візуальне представлення речей дозволяє користувачам швидко орієнтуватись у своєму гардеробі та приймати рішення щодо комплектації образів.

Проте, аналіз існуючих рішень виявляє низку системних недоліків у реалізації цієї базової функції. По-перше, багато додатків використовують надмірно спрощені алгоритми обробки зображень, що призводить до втрати якості фотографій після завантаження. Це особливо критично для деталей, таких як текстура тканини або відтінки кольорів. По-друге, погана організація фотогалереї. Багато рішень не пропонують зручних інструментів для сортування або фільтрації зображень, що ускладнює пошук потрібних предметів у великому гардеробі. Особливо це відчутно при роботі з категоріями, де однакові типи одягу можуть мати суттєві візуальні відмінності.

Варто також відзначити проблеми з синхронізацією фотографій між пристроями. Часто знімки, зроблені на одному смартфоні, погано відображаються на іншому через відмінності в роздільній здатності екранів або обмеження платформ. Це особливо критично для користувачів, які працюють з гардеробом на кількох пристроях одночасно.

Аналіз відгуків у Google Play показує, що користувачі часто скаржаться на нерпиємний інтерфейс, відсутність офлайн-режиму та обмежені можливості персоналізації.

Все це підтверджує необхідність створення нового додатку, який би усунув ці недоліки та запропонував більш гнучке та зручне рішення для управління гардеробом.

Також було проведено більш детальний аналіз вище перелічених існуючих аналогічних рішень, який дозволив виявити їх сильні та слабкі сторони, що стало

основою для формування унікальних характеристик нашого додатка. Дослідження охопило популярні програми, доступні у Google Play, а також деякі спеціалізовані рішення. Нижче наведено порівняльний аналіз основних функцій та характеристик аналогів.

Stylebook (рис. 1.1) основні функції:

- каталогізація одягу;
- створення образів;
- планування гардеробу.

Переваги:

- високий рівень деталізації характеристик одягу;
- можливість створення календаря носіння.

Недоліки:

- відсутність автоматичної генерації образів;
- обмежена безкоштовна версія.

ClosetSpace (рис. 1.2) основні функції:

- управління гардеробом;
- створення комбінацій;
- статистика.

Переваги:

- простий інтерфейс;
- можливість синхронізації між пристроями.

Недоліки:

- відсутність інтеграції з погодними сервісами;
- обмежена підтримка мов.

Smart Closet (рис. 1.3) основні функції:

- управління гардеробом;
- автоматичні рекомендації;
- статистика.

Переваги:

- використання штучного інтелекту для генерації образів;
- детальна статистика використання одягу.

Недоліки:

- висока вартість преміум-версії;
- вибагливість до апаратних ресурсів.

1.2 Виявлення та вирішення проблем

Під час аналізу існуючих додатків було виявлено кілька ключових проблем, які потребують вирішення.

Перша проблема – це обмежена функціональність у безкоштовних версіях. Багато додатків блокують основні функції, такі як створення комплектів або детальна статистика, і пропонують їх тільки в платній версії. Це обмежує доступність для широкого кола користувачів.

Друга проблема – це відсутність якісного офлайн-режиму. Більшість додатків вимагають постійного підключення до інтернету для роботи з основним функціоналом, що робить їх малопридатними для використання, наприклад, у подорожах або місцях зі слабким зв'язком.

Третя проблема – це примітивні алгоритми підбору одягу. Існуючі додатки часто пропонують комплекти, які не враховують реальні уподобання користувача або поточні умови. Наприклад, вони можуть рекомендувати легкий светр у спекотний день або не поєднувати речі, які насправді гармоніюють між собою за стилем.

Для вирішення цих проблем у додатку ClothesAdvisor було запропоновано:

- офлайн-режим до більшості функціоналу із локальним зберіганням даних;
- розширені алгоритми підбору одягу, які враховують не тільки колір і категорію, але й погоду, призначення одягу (спорт, прогулянка та інші), стиль користувача;
- безкоштовний доступ до всіх основних функцій без обмежень.

1.3 Постановка задачі

Розробка додатка ClothesAdvisor спрямована на вирішення конкретних проблем сучасних користувачів, які стикаються з труднощами в організації гардеробу та щоденному підборі одягу.

На основі проведеного аналізу було сформульовано основні завдання та частини (технології) для розробки додатка ClothesAdvisor.

Перше завдання – створити зручний інтерфейс, який дозволить користувачам легко додавати, редагувати та організовувати свій гардероб. Важливо, щоб процес був максимально інтуїтивним і не вимагав додаткових інструкцій. Також, реалізація зручного та приємного на вигляд користувацького інтерфейсу буде включати особливості реалізації, до яких можна буде віднести динамічну зміну кольорової схеми інтерфейсу в залежності від обраної системної кольорової схеми операційної системи Android, яка працює на сучасних версіях Andoid (версія 12 та вище).

Друге завдання – розробити систему генерації комплектів одягу, яка враховувала б не лише базові параметри (тип одягу, колір), але й додаткові фактори, такі як поточна погода, можливі кольорові палітри, обрані користувачем, та призначення одягу (наприклад, прогулянка або ділова зустріч). Це дозволить робити більш точні та персоналізовані рекомендації.

Третє завдання – реалізувати детальну статистику, яка показувала б, які речі використовуються найчастіше, загальну кількість речей в гардеробі, яка кількість речей відповідає поточному сезону тощо. Ця інформація допоможе користувачам оптимізувати свій гардероб і приймати обґрунтовані рішення при покупках.

Для розробки мобільного додатку з управління гардеробом було обрано сучасний стек технологій, що забезпечує високу продуктивність, безпеку та зручність використання. Основу розробки складає платформа Android з використанням середовища Android Studio, яке надає потужні інструменти для створення, тестування та оптимізації додатків.

Для створення інтерфейсу додатку було обрано Jetpack Compose, який дозволяє створювати адаптивні та інтерактивні екрани з мінімальним обсягом коду.

Для зручного завантаження і відображення зображень одягу інтегровано інструмент Coil, що надає готові рішення.

Мова програмування Kotlin стала основним інструментом для написання коду через її використання в Jetpack Compose. Також мова має сучасний синтаксис, безпеку типів даних та повну сумісність із Java. Для зручної роботи з даними користувачів використано Room – бібліотеку для локального зберігання даних, яка дозволяє ефективно керувати базою даних SQLite.

Важливим компонентом архітектури є Retrofit2 – потужна бібліотека для здійснення мережевих запитів, яка використовується для взаємодії із серверною частиною додатку. Retrofit2 забезпечує простий та елегантний спосіб роботи з REST API.

Цільова аудиторія додатка охоплює широке коло людей з різними потребами та звичками:

- студенти та молоді професіонали (18–30 років) – активні користувачі смартфонів, які цінують простоту та функціональність. Вони часто стикаються з нестачею часу по вранці та потребують швидкого підбору одягу для навчання, роботи або зустрічей з друзями. Для них важливі такі функції, як швидке створення комплектів, синхронізація з розкладом і рекомендації на основі погоди;
- офісні працівники (25–45 років) – користувачі, які дотримуються дрес-коду або хочуть виглядати презентабельно. Вони потребують інструменту для планування образів на тиждень, аналізу того, які речі найчастіше використовуються, а які взагалі лежать без діла. Також їм буде корисна функція створення "капсульних гардеробів" для роботи;
- модні ентузіасти (будь-якого віку) – люди, які слідкують за трендами і люблять експериментувати зі стилем. Для них важливі такі функції, як збереження ідей образів, можливість ділитися своїми комплектами в соціальних мережах та отримувати інспірацію на основі власного гардеробу;

- подорожуючі – користувачі, які часто змінюють кліматичні умови і потребують швидкого підбору одягу під конкретну подорож. Для них буде особливо корисна функція автоматичного підбору речей з урахуванням прогнозу погоди в пункті призначення;
- батьки сімейств – зайняті люди, які керують не лише своїм гардеробом, але й одягом дітей. Вони оцінять можливість створення окремих профілів для кожного члена сім'ї, нагадування про сезонну зміну одягу та поради щодо комплектів для різних подій.

Таким чином, основним завданням є створення не просто "електронного каталогу" одягу, а справжнього помічника, який би враховував індивідуальні потреби кожної категорії користувачів і допомагав їм приймати оптимальні рішення щодо свого гардеробу щодня.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Функціональні вимоги до мобільного додатку

Мобільний додаток для управління гардеробом повинен мати наступні функціональні можливості:

Авторизація та реєстрація користувачів:

- вхід у систему у ролі гостя з локальним зберіганням даних;
- реєстрація за електронною адресою для збереження даних у обліковому записі;
- авторизація з можливістю синхронізації даних між пристроями та сервером, де користувач буде вибирати, за якими даними (локальними або серверними) буде відбуватись синхронізація;
- перезапис пароля користувача.

Управління гардеробом:

- додавання елементів одягу з фотографуванням та збереженням зображень;
- редагування характеристик одягу: назва, категорія (головний убір, верхній одяг, взуття тощо), сезонність (зима, весна, літо, осінь, всесезонний), основний колір (формат RGB), матеріал (бавовна, вовна, поліестер тощо), додаткові параметри: бренд, дата придбання, вартість (за бажанням);
- видалення елементів одягу;
- можливість видалення заднього фону зображень для одягу з гардеробу;
- можливість позначання улюблених елементів одягу для покращення результату генерації комбінацій одягу.

Генерація наборів (комбінацій) одягу:

- автоматична генерація наборів на основі: стилю та кольорової гами, погодних умов та призначення (робота, відпочинок тощо), переваг користувача (кольорова палітра та інше);
- ручне створення наборів (комбінацій) з можливістю вибору елементів гардеробу;
- редагування збережених наборів;

– видалення збережених наборів.

Статистика:

– статистика кількості речей (загально, для кожного сезону, для кожної категорії);
– статистика за віком речей (топ 5 речей від старішого до новішого);
– статистика використання одягу в наборах (топ 10 речей);
– відсоткове співвідношення улюблених та не улюблених елементів одягу.

Особисті налаштування:

– зміна мови інтерфейсу, вибираючи між англійською та українською мовами;
– зміна електронної пошти користувача;
– зміна пароля користувача.

2.2 Нефункціональні вимоги до мобільного додатку

Сумісність:

– підтримка Android 7.0 (API 24) та вище;
– адаптація інтерфейсу для різних розмірів екранів (телефони, планшети).

Продуктивність:

– час запуску сторінок не повинен перевищувати 10 секунд;
– оптимізація використання батареї, уникнення зайвих фонових процесів.

Безпека:

– локальне зберігання даних у гостьовому режимі (офлайн);
– захищена передача даних при роботі з сервером (HTTPS).

Інтерфейс:

– інтуїтивно зрозумілий UI з підтримкою темної та світлої тем;
– підтримка локалізації UI на українській та англійській мовах.

2.3 Взаємодія з серверною частиною

Мобільний додаток взаємодіє з серверною частиною для таких функцій:

- реєстрація користувача;
- синхронізація даних користувача та сервера після авторизації та її регулярна підтримка в авторизованому режимі;
- редагування облікових даних користувача (електронна адреса пошти, пароль);
- оновлення даних гардеробу користувача, які включають в себе список елементів одягу та комбінацій одягу;
- можливість видалення заднього фону зображень для одягу з гардеробу;
- автоматична генерація наборів (комбінацій).

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

3.1.1 Діаграма прецедентів

Було створено UML [5] діаграму прецедентів (рис. 3.1), яка була базована на аналізі функціонала, визначеному у третьому розділі. При розробці в даній системі було передбачено два типи користувачів: Гість та Авторизований користувач.

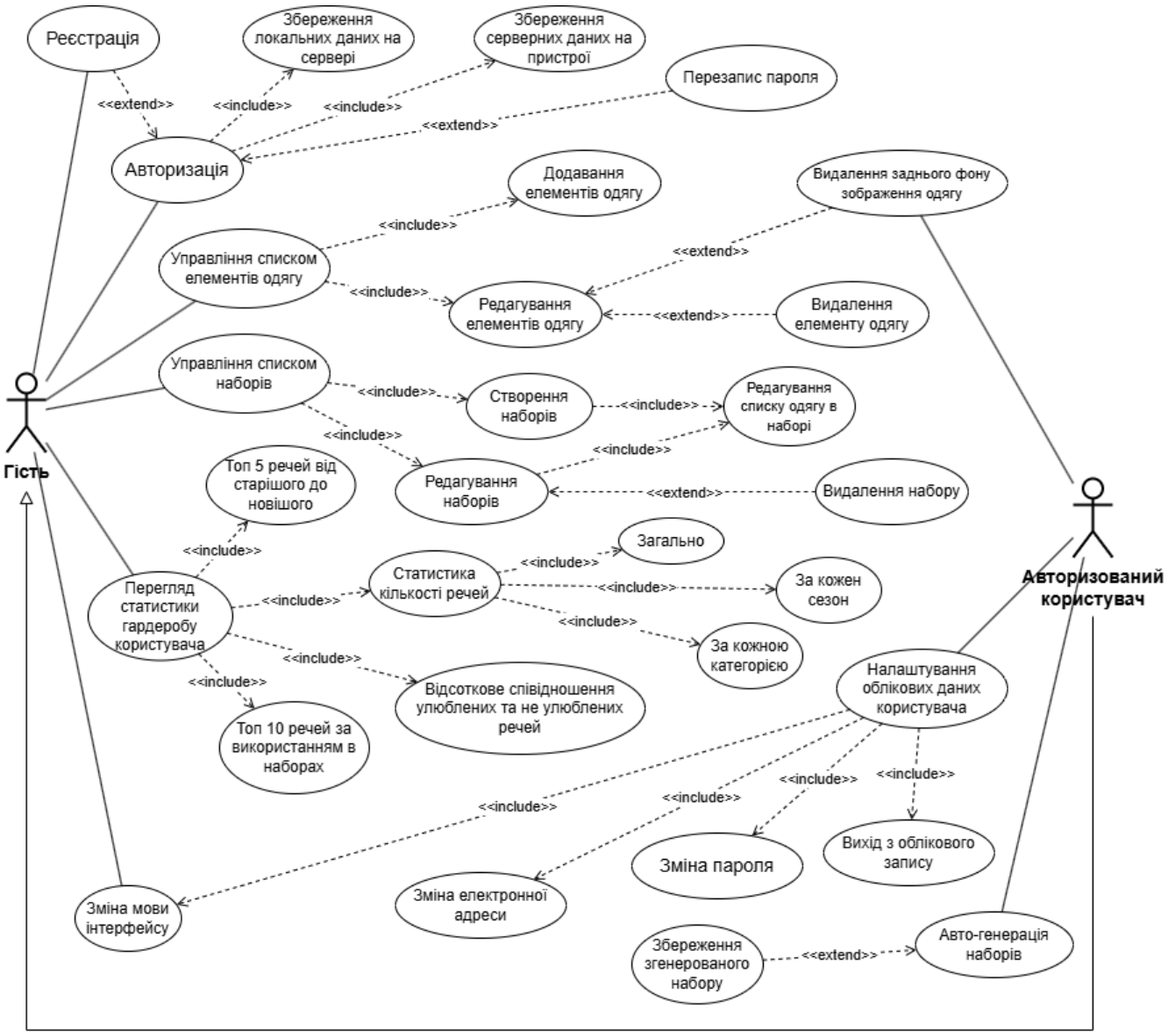

Рисунок 3.1 – UML діаграма прецедентів (рисунок виконаний самостійно)

3.1.2 Діаграма розгортання

Детальну інформацію про компоненти та розгортання поточної програмної системи можна побачити на UML діаграмі розгортання (рис. 3.2). Згідно неї, мобільний додаток складається з трьох частин:

- компонент про роботу з зображеннями;
- компонент про спосіб взаємодії користувача (в UI) з REST API через сервіси;
- компонент про роботу з локальною базою даних.

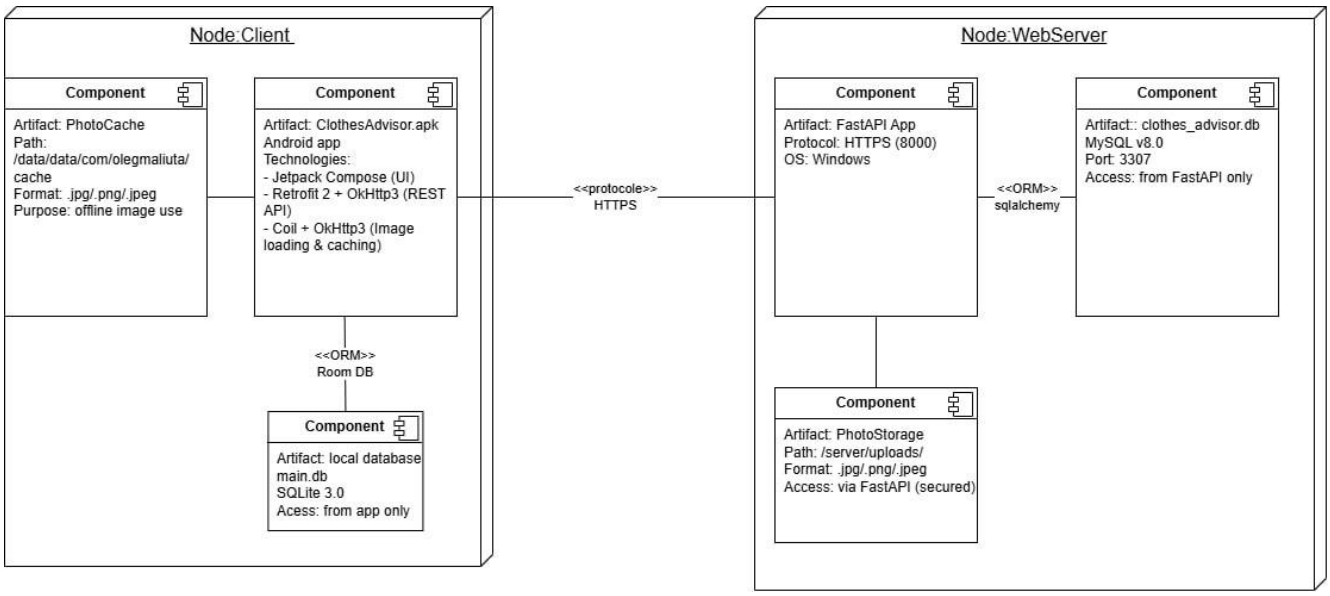

Рисунок 3.2 – UML діаграма розгортання (рисунок виконаний самостійно)

3.1.3 Діаграма активності

Детальні кроки процесу виконання основних дій користувача показано на рисунках UML діаграм активності. статистика (рис. 3.5), налаштування користувача (рис. 3.6), перезапис пароля (рис. 3.7).

На рисунку 3.3 продемонстровано процеси реєстрації та авторизації, які відбуваються на відповідних сторінках.

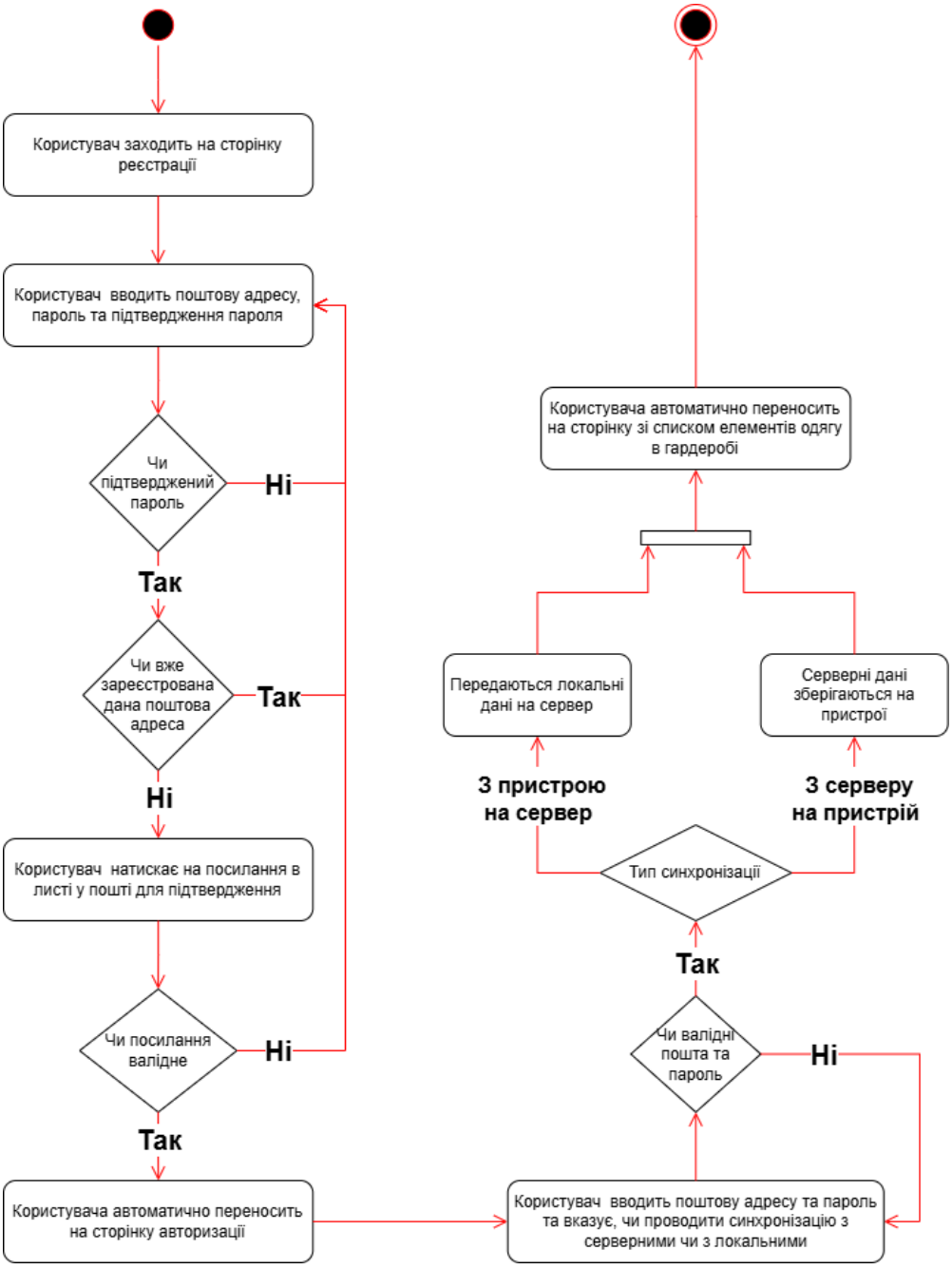

Рисунок 3.3 – UML діаграма активності для входу в систему (рисунок виконаний самостійно)

На рисунку 3.4 продемонстровано процес авто-генерація наборів, який відбуваються на спеціально створеній для цього сторінці з усім потрібним функціоналом.

Рисунок 3.4 – UML діаграма активності для авто-генерації наборів (рисунок виконаний самостійно)

На рисунку 3.5 продемонстровано процес перегляду особистої статистики гардеробу користувача.

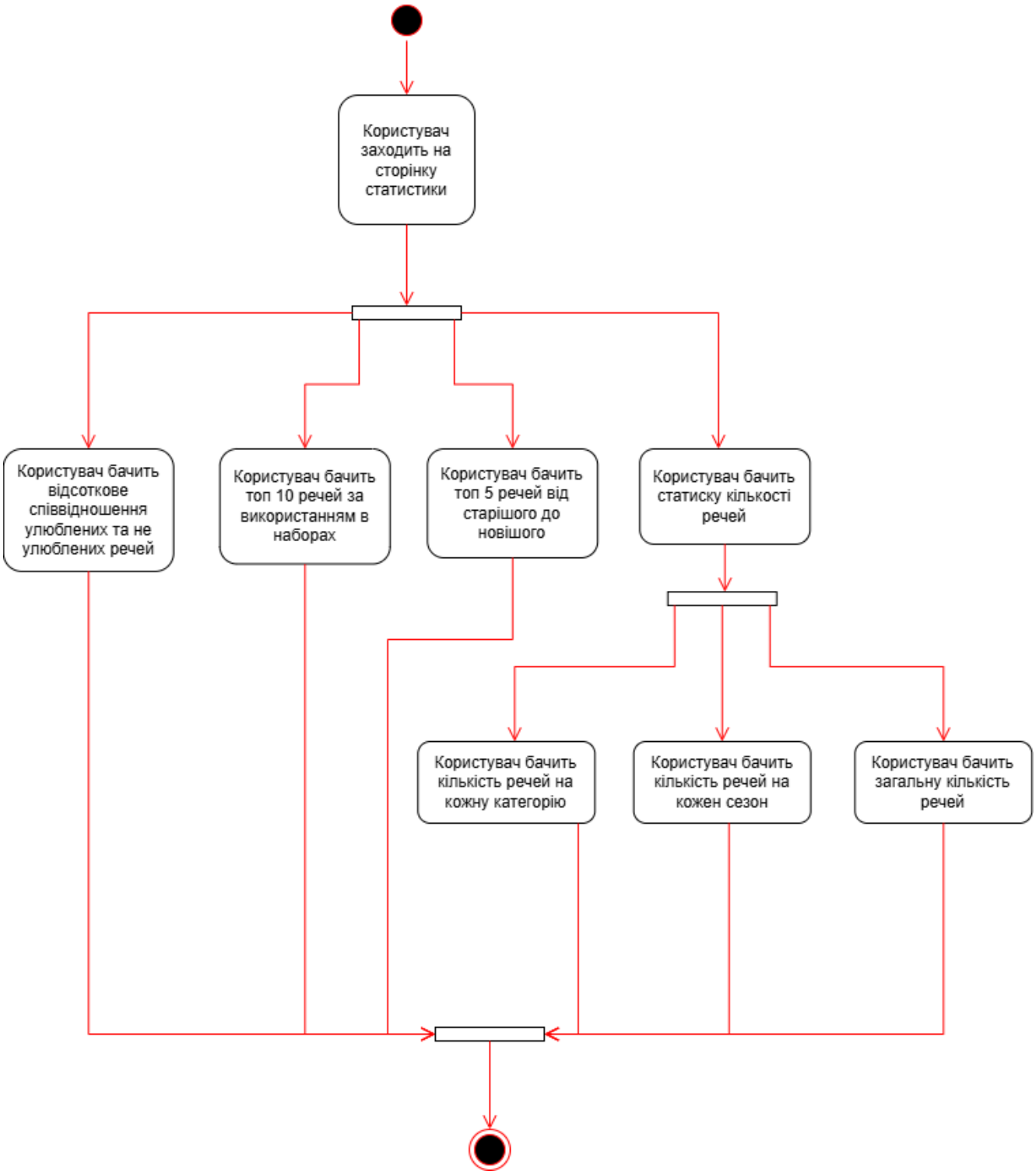

Рисунок 3.5 – UML діаграма активності для статистики (рисунок виконаний самостійно)

На рисунку 3.6 продемонстровано процес використання сторінки особистих користувацьких налаштувань.

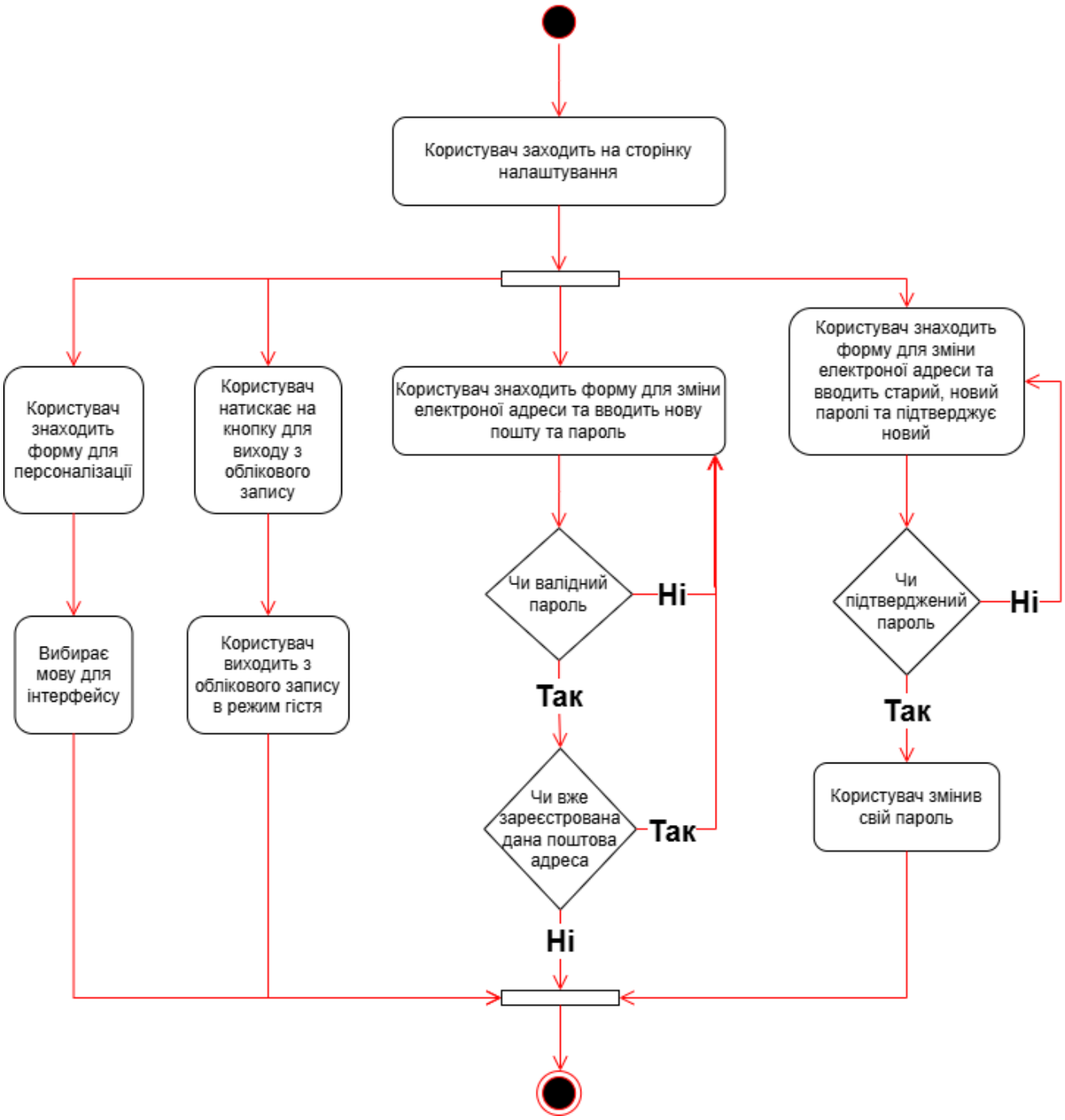

Рисунок 3.6 – UML діаграма активності для налаштувань користувача (рисунок виконаний самостійно)

На рисунку 3.7 продемонстровано процес перезапису пароля користувача, який відбуваються на сторінці авторизації у випадку, якщо користувач забув свій пароль.

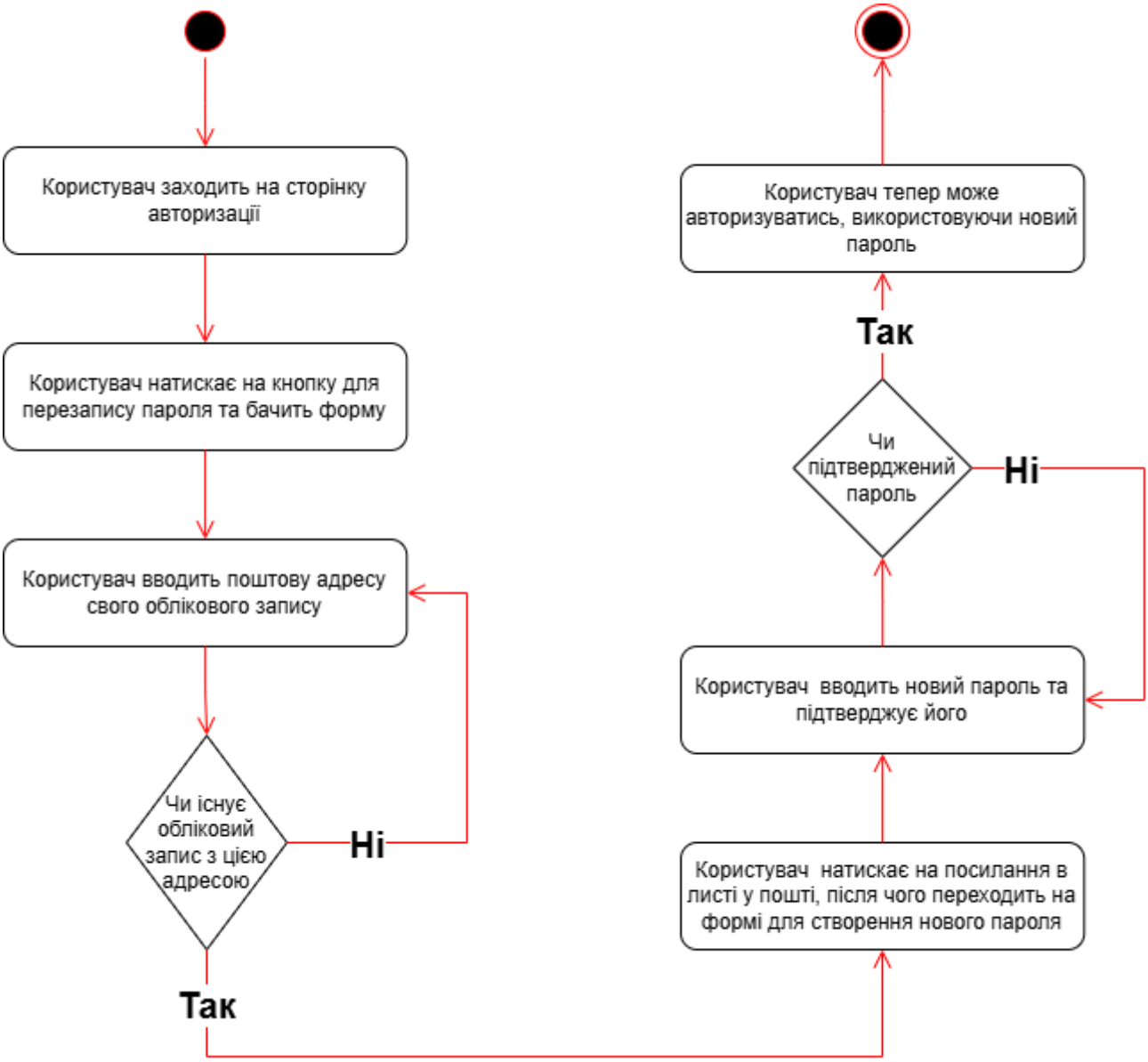

Рисунок 3.7 – UML діаграма активності для перезапису пароля (рисунок виконаний самостійно)

3.1.4 Діаграма станів

Було створено UML діаграму станів для демонстрації процесу додавання та редагування елементів одягу (рис. 3.8).

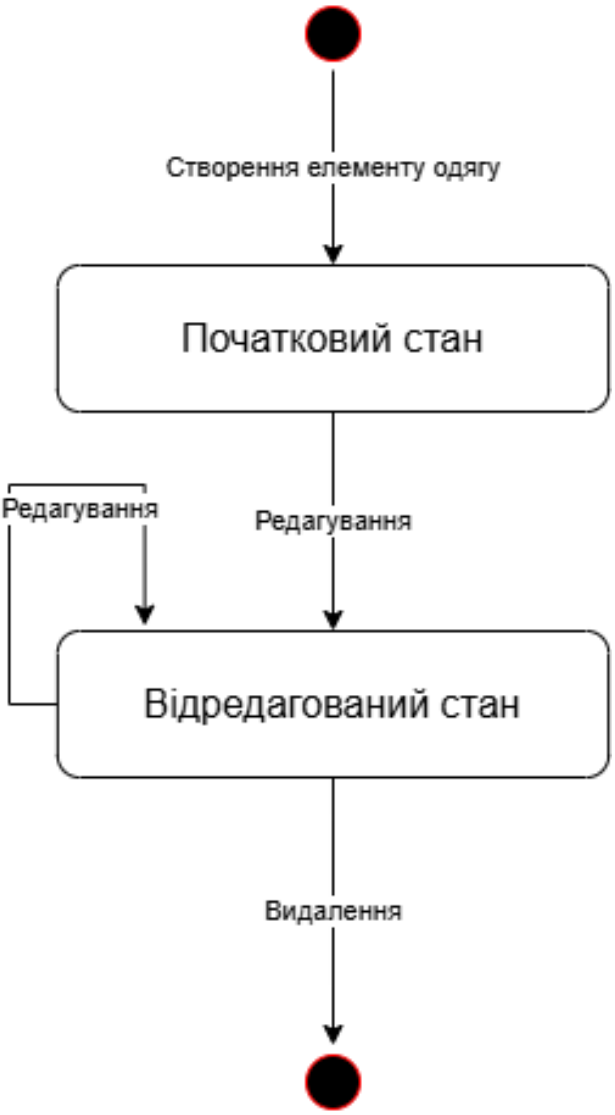

Рисунок 3.8 – UML діаграма станів для додавання та редагування елементів одягу (рисунок виконаний самостійно)

Було створено UML діаграму станів для демонстрації процесу додавання та редагування наборів (рис. 3.9).

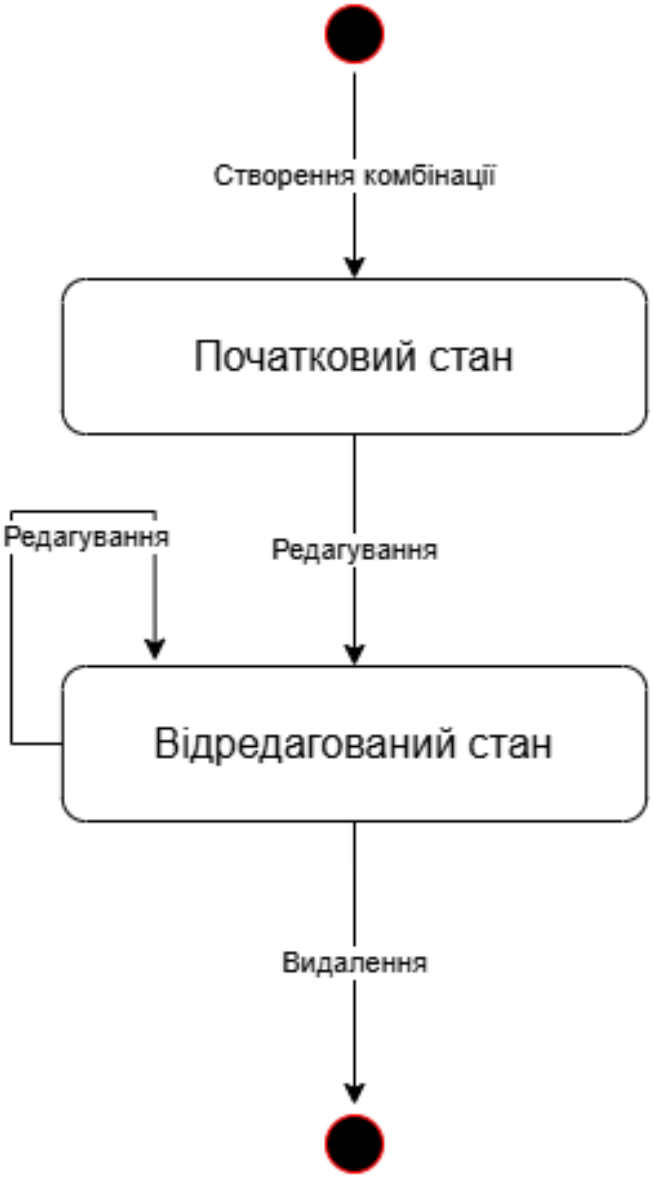

Рисунок 3.9 – UML діаграма станів для додавання та редагування наборів (рисунок виконаний самостійно)

3.2 Проєктування архітектури ПЗ

Для створення мобільного додатку "Управління гардеробом" було обрано сучасні технології, які забезпечують надійність та зручність розробки. Клієнтська частина для Android розробляється переважно на мові Kotlin з використанням офіційних інструментів від Google. Таких, як середа розробки мобільних додатків Android Studio, інструмент для створення UI Jetpack Compose [6]. В якості ORM для взаємодії з локальною БД використовується технологія Room Database та для взаємодії з REST API – бібліотеки Retrofit2, OkHttp3 та Coil (для завантаження і відображення зображень).

Для роботи з даними застосовується архітектура MVVM [7] (Model View ViewModel) (рис. 3.10), що дозволяє ефективно розділяти логіку та інтерфейс.

Архітектура додатку (рис. 3.10) побудована за принципами чистої архітектури, що дозволяє чітко розділити відповідальність між різними частинами системи. Вона складається з трьох основних рівнів.

Рівень даних відповідає за отримання та зберігання інформації. Він включає локальну базу даних на пристрої користувача (Room Database [8]) та зв'язок із сервером через REST API. Репозиторії та сервіси виконують роль посередників між різними джерелами даних.

Рівень доменної логіки містить основні бізнес-правила додатку. Тут знаходяться моделі даних. Цей рівень не залежить від конкретних технологій реалізації, що полегшує майбутні зміни.

Рівень презентації відповідає за відображення інформації та взаємодію з користувачем. Він складається з екранів (Screen), UI компонентів, ViewModel для керування станом інтерфейсу та механізмів керування станом додатку.

У додатку для реалізації UI в Jetpack Comose використовується патерн State, який особливо ефективний для обробки різних режимів роботи інтерфейсу. Цей поведінковий патерн дозволяє графічному об'єкту змінювати свою поведінку при зміні внутрішнього стану, що ідеально підходить для динамічних інтерфейсів.

Для управління гардеробом також було застосувано патерн Composite, який ідеально підходить для організації ієрархічних структур інтерфейсу. Цей

структурний патерн дозволяє єдинообразно працювати як з окремими елементами, так і з їх групами.

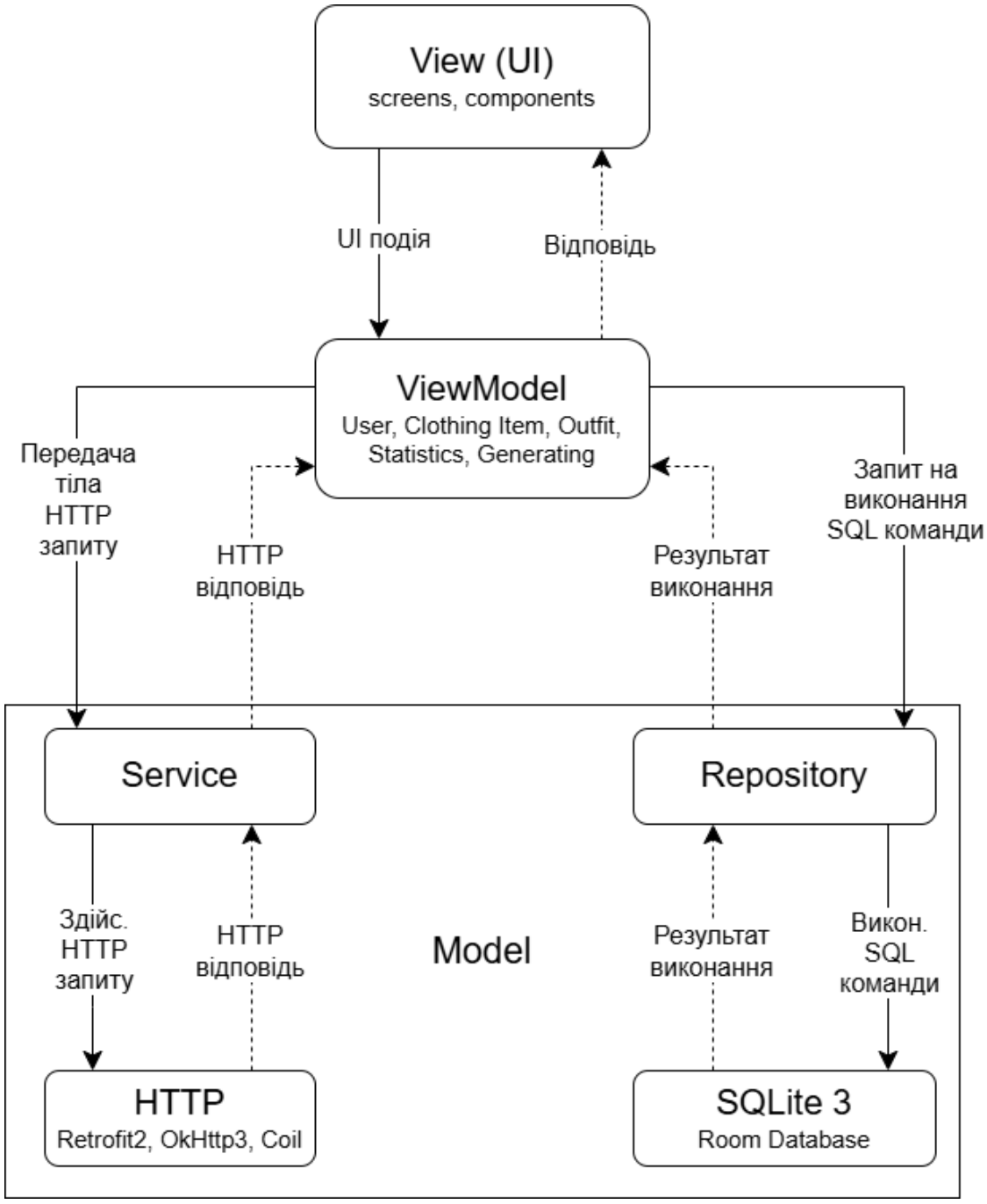

Рисунок 3.10 – Схема архітектури MVVM, яка використовується у додатку (рисунок виконаний самостійно)

3.3 Проєктування структури зберігання даних

Система зберігання даних для додатку "Управління гардеробом" була розроблена з урахуванням необхідності забезпечення стабільної роботи як в онлайн, так і в офлайн-режимах. Для цього було використано комбінацію локального та хмарного сховищ, що дозволяє користувачам мати доступ до своїх даних у будь-який час.

Основу системи становить локальна база даних на пристрої користувача, яка реалізована за допомогою Room Database на базі SQLite. Вона містить усі необхідні дані для роботи додатку без інтернет-з'єднання. Для синхронізації між різними пристроями використовується віддалене сховище на базі MySQL.

В локальній БД містяться елементи, які представляють собою елемент одягу (clothing_items) та набір одягу (outfits), які мають між собою зв’язок багато-до-багатьох (за який відповідає таблиця clothing_item_outfit_cross).

Варто зазначити, що в таблиці для одягу зберігаються посилання на зображення. Вони можуть посилатись як на ті, що є хмарному середовищі (для їх відображення буде потрібно підключення до інтернету), так і на ті, що зберігаються на самому пристрої. Локально збережені зображення після авторизації за локальними даними будуть автоматично завантажені у хмарне середовище.

Також для підтримки синхронізації даних було реалізовано регулярну перевірку «версій» локальних та серверних даних, які включають в себе список одягу та список наборів, де «версія» виступає у ролі точного часу останнього оновлення серверних даних. Ці «версії» порівнюються, і, якщо вони не співпадають, то відбувається оновлення даних на локальній БД згідно серверних даних (трохи детальніше про це можна почитати в підрозділі 4.3).

Дана система зберігання даних забезпечує:

- надійне зберігання всіх необхідних даних;
- швидкий доступ до інформації;
- ефективну синхронізацію між пристроями;
- оптимальне використання ресурсів пристрою;

– можливість подальшого розширення функціоналу.

Сутності локальної БД:

Елемент одягу (clothing_items):

– id – унікальний ідентифікатор речі;
– filename – шлях до зображення одягу, який може бути http посиланням, або вказувати розташування на локальному пристрої;
– name – назва одягу;
– category – категорія одягу (наприклад головний одяг, взуття);
– season – сезон, для якого призначено одяг (зима, весна, літо, осінь);
– red, green, blue – цілі числа, можливі значення яких від 0 до 255, що представляють собою компоненти кольорової схеми RGB;
– material –матеріал, з якого виготовлено одяг (наприклад, бавовна, шкіра);
– brand – бренд одягу;
– purchase_date – дата придбання даного одягу;
– price – вартість одягу;
– is_favorite – значення, яке вказує, чи є річ улюбленою для користувача.

Елемент одягу (outfits):

– id – унікальний ідентифікатор набору;
– name – назва набору.

Таблиця для зв’язку багато-до-багатьох між clothing_items та outfits (clothing_item_outfit_cross) з зовнішніми ключами: clothing_item_id та outfit_id.

Для демонстрації структури бази даних була спроектована ER-модель даних (рис. 3.11), яка демонструє всі сутності та їхні взаємозв'язки.

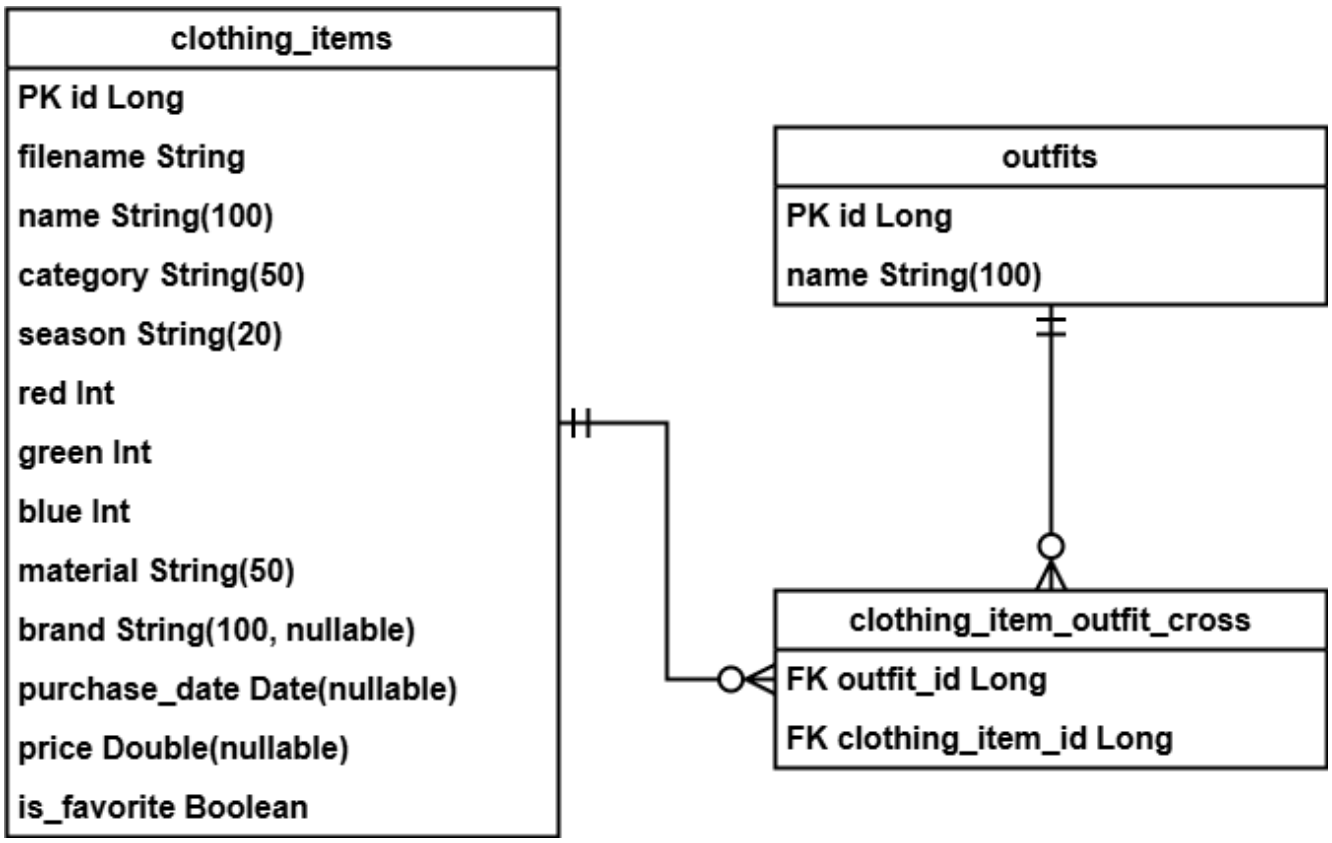

Рисунок 3.11 – ER модель локальної бази даних (рисунок виконаний самостійно)

3.4 Приклади найцікавіших алгоритмів та методів

3.4.1 Кругова діаграма для статистики гардеробу

Для наочності було реалізовано компонент, який займається малюванням кругової діаграми на користувацькому інтерфейсі згідно вказаних даних. На сторінці статистики ця діаграма демонструє співвідношення «улюблених» та не «улюблених» елементів одягу у гардеробі. Вона автоматично оновлюється при зміні цього статусу одягу в гардеробі. Програмний код цього UI-компоненту можна побачити у додатку Г.

3.4.2 Функція «піпетка» для вибору кольору з фотографії

Функція "піпетка" дозволяє користувачам точно визначати колір предметів одягу з фотографій. При торканні до зображення система аналізує піксель в середені області дотику, визначає його колір, переводить у формат RGB та зберігає, після чого користувач зможе сам відредагувати вибране RGB значення. Програмний код UI-компоненту, що містить функцію «піпетка», можна побачити у додатку Д.

3.5 Створення UI/UX або іншого дизайну системи

Дизайн додатку розроблявся з акцентом на простоту та інтуїтивність використання. Тобто, щоб навіть нові користувачі могли легко орієнтуватись у функціоналі без необхідності вивчати інструкції.

Нижче показано знімки екрану додатку з використанням основних сторінок. Додаток ділиться на такі основні сторінки: список речей (рис. 3.12), список наборів (рис. 3.13), авто-генерація наборів (рис. 3.14), статистика (рис. 3.15), налаштування користувача (рис. 3.16). На ці сторінки користувач може легко та в будь-який момент перейти, використовуючи нижнє навігаційне меню.

Показано сторінку зі списком речей з можливостями пошуку, сортування та фільтрації (рис. 3.12).

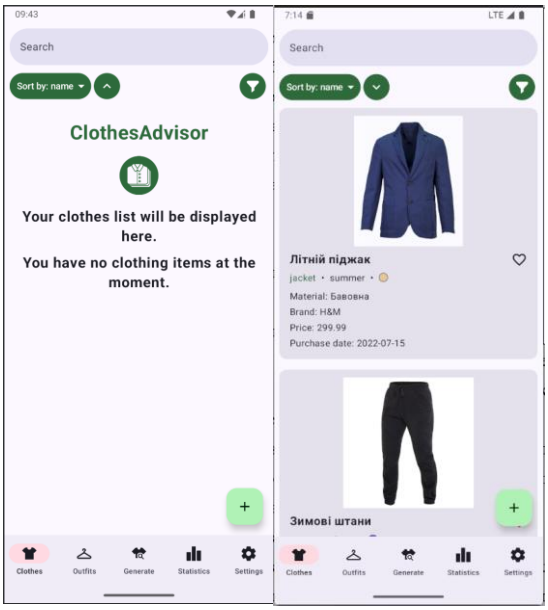

Рисунок 3.12 – Знімок екрану додатку зі сторінкою зі списком речей (рисунок виконаний самостійно)

Показано сторінку зі наборів з можливістю пошуку та перегляду кількості речей в наборі (рис. 3.13).

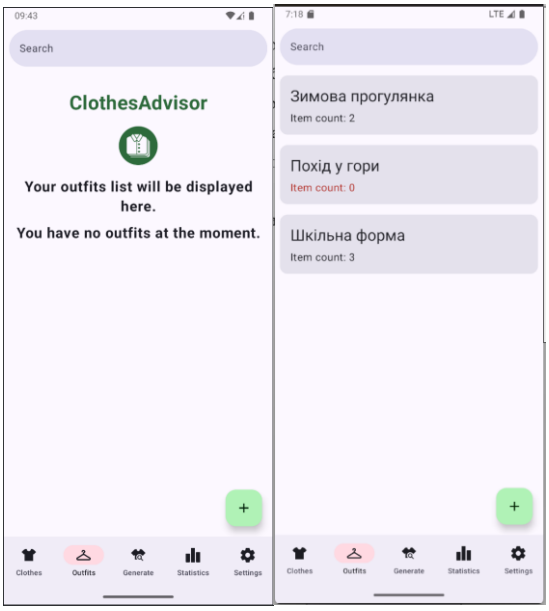

Рисунок 3.13 – Знімок екрану додатку зі сторінкою зі списком наборів (рисунок виконаний самостійно)

Показано сторінку авто-генерації наборів (рис. 3.14), де користувач після формування рекомендацій матиме можливість зберегти згенерований набір.

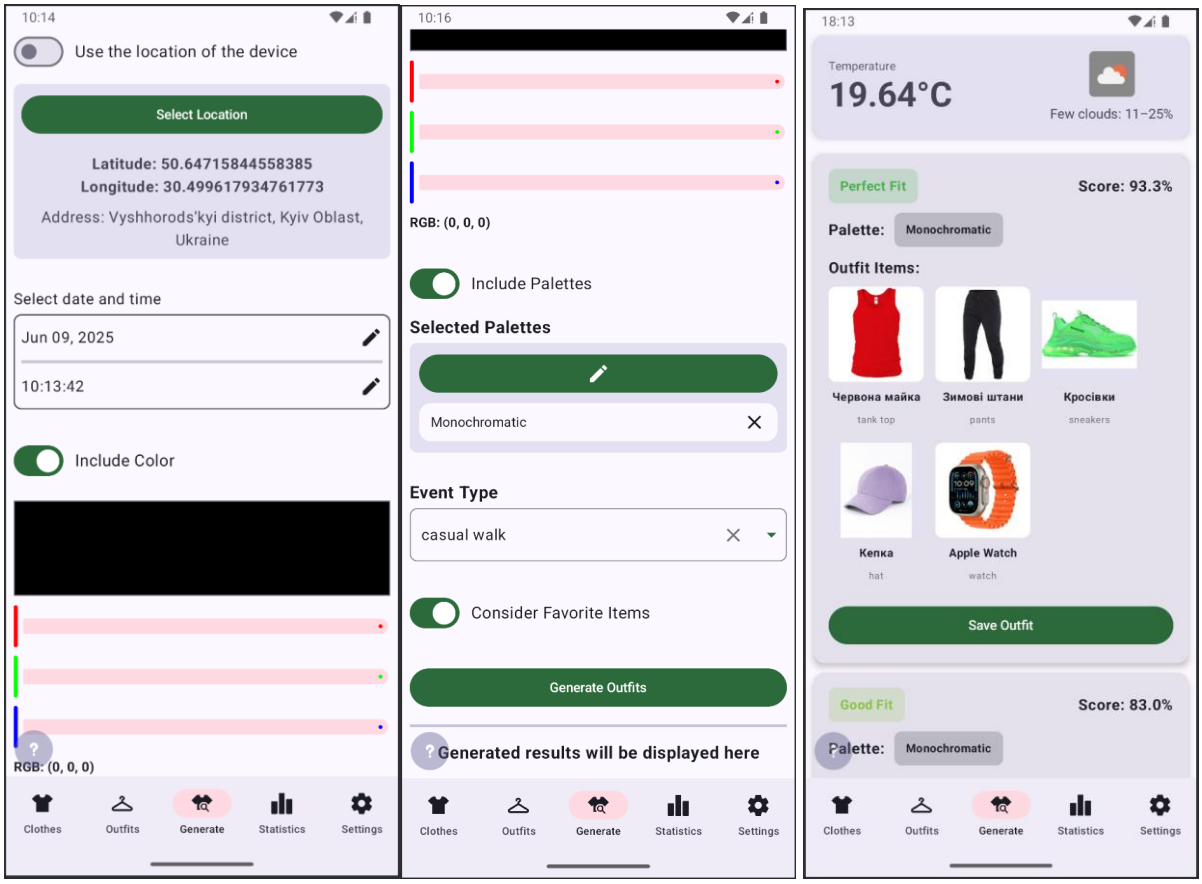

Рисунок 3.14 – Знімок екрану додатку зі сторінкою для авто-генерації наборів (рисунок виконаний самостійно)

Показано сторінку особистої статистики гардеробу користувача (рис. 3.15), яка складається з чотирьох частин:

– статистика загальної кількості речей, на кожен сезон та на кожну категорію;
– 5 найстаріших речей в гардеробі;
– 10 речей по кількості використань в наборах;
– співвідношення улюблених та звичайних речей.

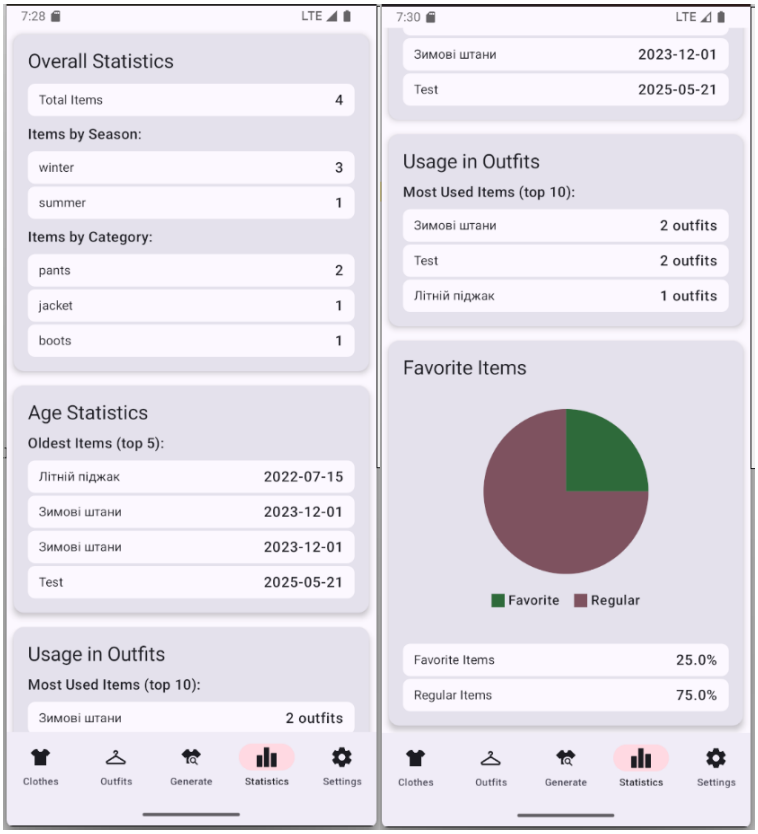

Рисунок 3.15 – Знімок екрану додатку зі сторінкою статистики (рисунок виконаний самостійно)

Показано сторінку особистих налаштувань користувача (рис. 3.16).

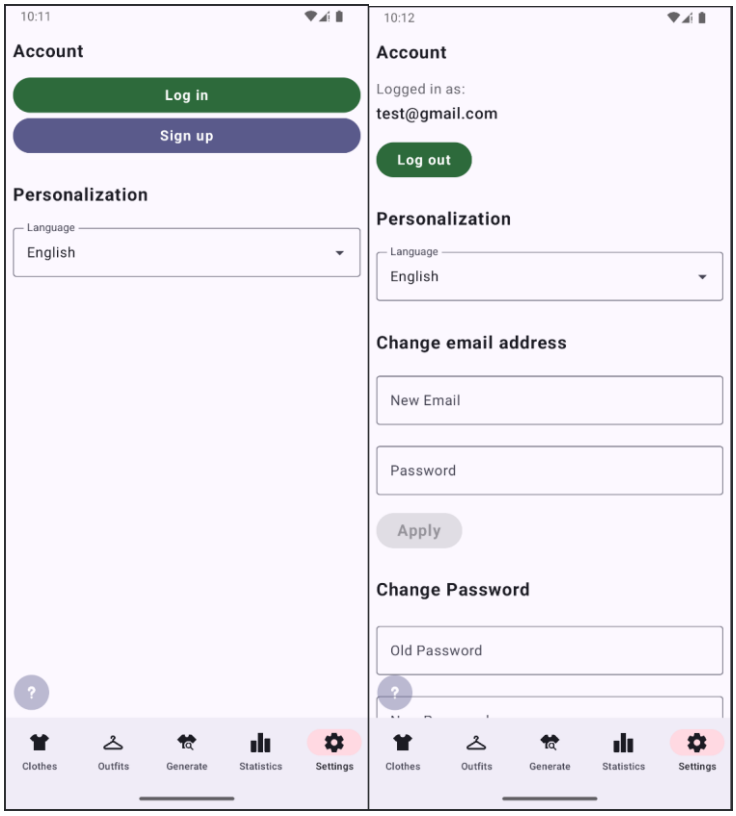

Рисунок 3.16 – Знімок екрану додатку зі сторінкою налаштувань (рисунок виконаний самостійно)

Також додаток містить додаткові сторінки, такі як сторінка для додавання або редагування елементу одягу (рис. 3.17), сторінка для додавання або редагування наборів (рис. 3.18), авторизація та реєстрація (рис. 3.19). Вони розраховані на менш часте використання і до них можна перейти із певних сторінок.

На сторінці для додавання або редагування елементу одягу (рис. 3.17) користувач має додаткову можливість видалення заднього фону зображення, яка доступна тільки для авторизованих користувачів.

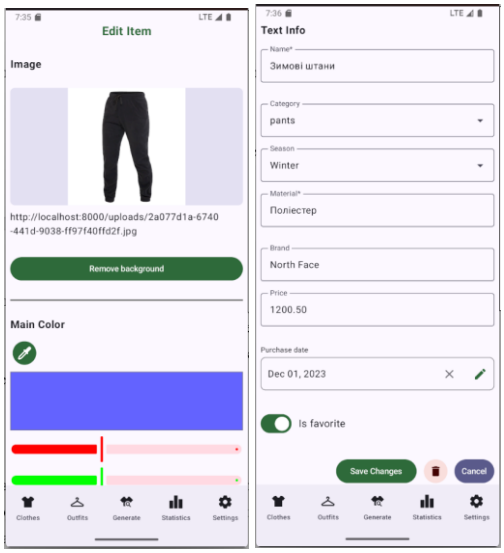

Рисунок 3.17 – Знімок екрану додатку зі сторінкою додавання або редагування елементу одягу (рисунок виконаний самостійно)

Варто зазначити, що на сторінці для додавання або редагування наборів (рис. 3.18) користувач не може додавати два або більше речей з однаковими категоріями.

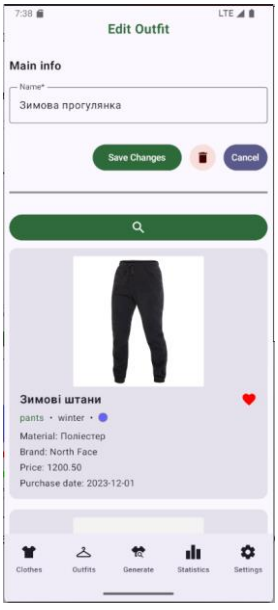

Рисунок 3.18 – Знімок екрану додатку зі сторінкою додавання або редагування набору (рисунок виконаний самостійно)

На сторінці авторизації (рис. 3.19) користувач може почати процедуру перезапису пароля, натиснувши на кнопку «Forgot?» у полі для вводу пароля.

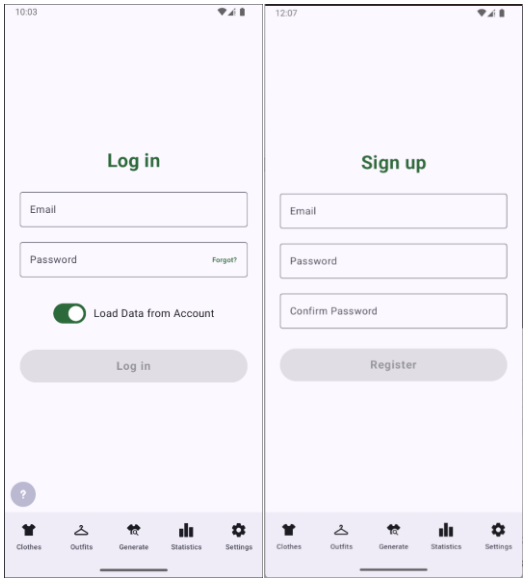

Рисунок 3.19 – Знімок екрану додатку зі сторінок авторизації та реєстрації (рисунок виконаний самостійно)

Дизайн додатку розроблявся з урахуванням сучасних тенденцій Android-розробки, зокрема з використанням вбудованої за замовченням системи динамічних кольорів. Це дозволяє інтерфейсу автоматично адаптуватись до кольорової палітри шпалер пристрою, створюючи гармонійне візуальне середовище.

Розроблений інтерфейс використовує динамічні кольори для ключових елементів інтерфейсу. Кнопки, іконки та акцентні елементи автоматично набувають відтінків, що поєднуються з кольоровою схемою пристрою користувача (рис. 3.20).

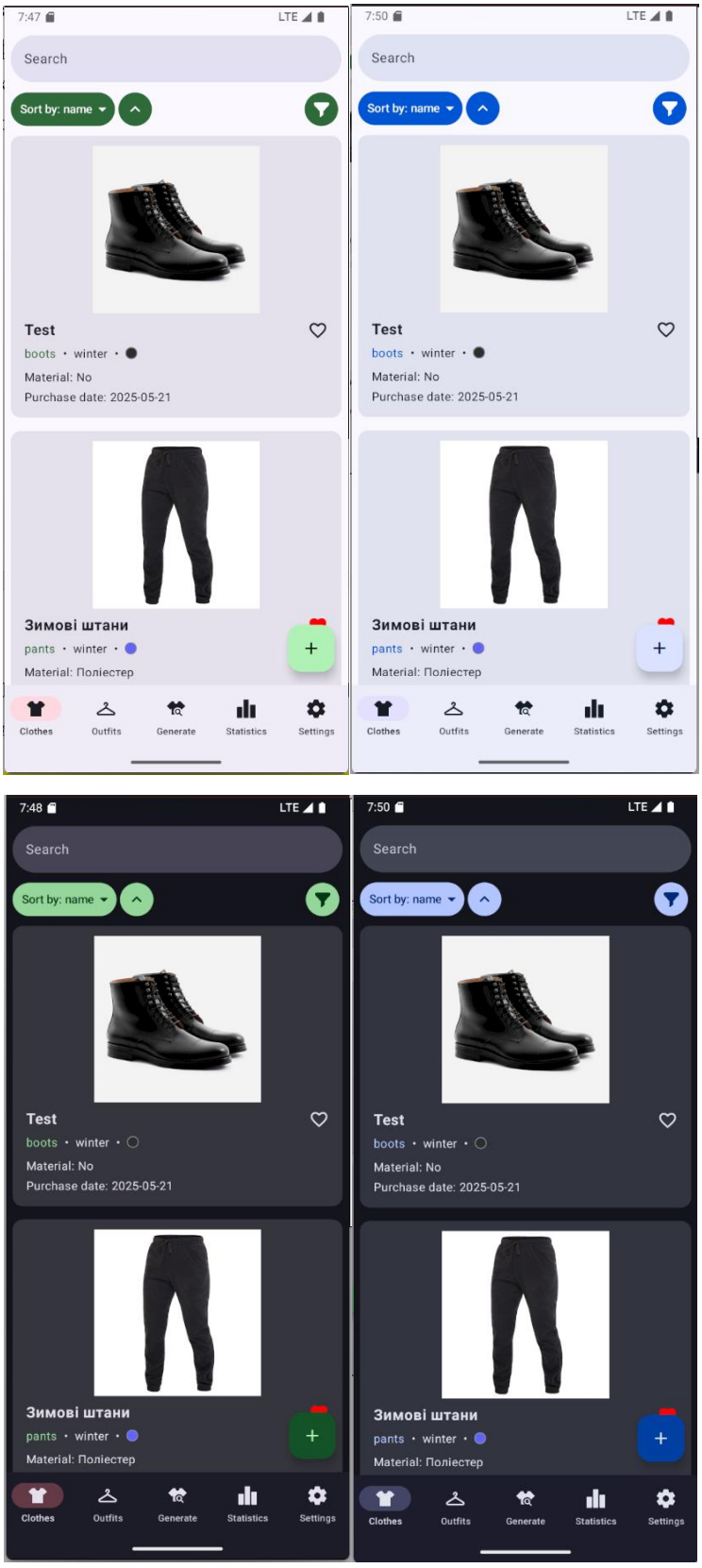

Рисунок 3.20 – Знімок екрану додатку для демонстрації зміни динамічних кольорів інтерфрейсу (рисунок виконаний самостійно)

Однак ця система працює лише для версій Android 12+, а в більш старих версіях використовуються статичні кольори. Приклади з використанням статичних кольорів можна побачити на рисунку 3.21.

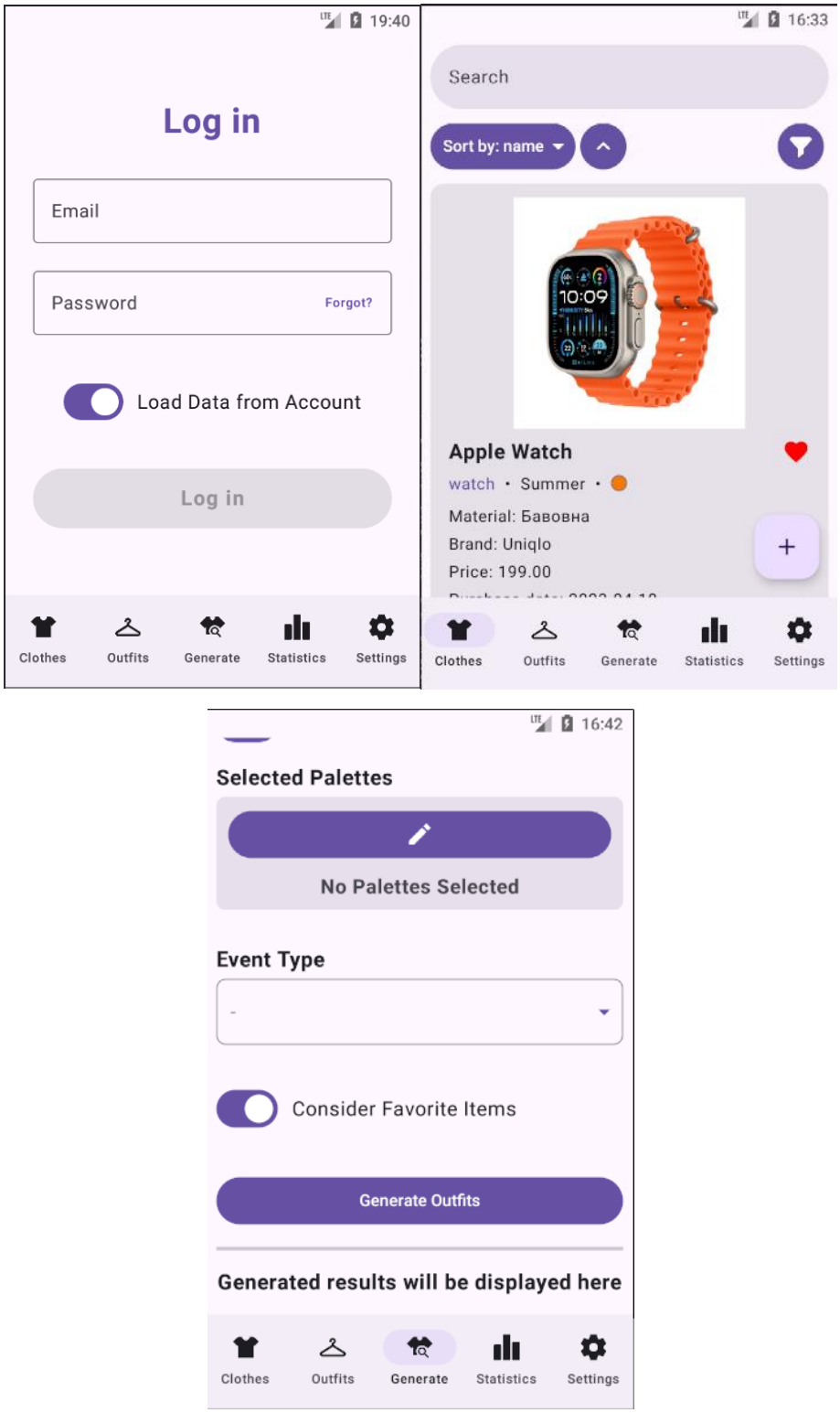

Рисунок 3.21 – Знімок екрану додатку для демонстрації статичних кольорів інтерфрейсу в старих версіях Android (рисунок виконаний самостійно)

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Використані бібліотеки

У розробці мобільного додатка для управління гардеробом було використано низку сучасних бібліотек, які значно прискорили розробку, забезпечили стабільність та додали ключові функціональні можливості. Нижче наведено опис основних бібліотек та їх призначення:

Jetpack Compose:

- призначення: сучасний UI-фреймворк для створення інтерфейсу Android-додатків.
- використання: побудова всіх екранів додатка (авторизація, налаштування, гардероб, генерація образів та інші сторінки), забезпечення адаптивного дизайну для різних розмірів екранів та підтримка темної/світлої теми та анімацій.
- переваги: декларативний підхід до розробки інтерфейсів, висока продуктивність та спрощена підтримка коду.

Navigation Compose:

- призначення: бібліотека для навігації між екранами в Jetpack Compose.
- використання: реалізація переходів між екранами (наприклад, зі списку гардеробу до деталей елемента), обробка глибоких посилань (deep links) для швидкого доступу до певних розділів.
- переваги: простий API для керування стеком навігації та інтеграція з анімаціями переходів.

Retrofit2 + OkHttp3:

- призначення: бібліотеки для роботи з мережевими запитами.
- використання: взаємодія з серверною частиною через REST API (авторизація, синхронізація гардеробу, управління гардеробом та інший функціонал) і надсилання та отримання даних у форматі JSON.
- переваги: проста інтеграція та підтримка асинхронних запитів, вбудована обробка помилок і логування.

Coil:

- призначення: бібліотека для завантаження та кешування зображень.
- використання: відображення фотографій одягу в списках та детальних переглядах і оптимізація завантаження зображень (кешування, масштабування).
- переваги: висока продуктивність та низьке споживання пам’яті.

Room:

- призначення: ORM-бібліотека для локального зберігання даних у SQLite.
- використання: кешування даних у гостьовому режимі та зберігання інформації про елементи гардеробу, комбінації, налаштування.
- переваги: легка інтеграція з Kotlin (підтримка coroutines) та автоматична генерація SQL-запитів.

OpenStreetMap Droid:

- призначення: бібліотека для відображення мап на основі OpenStreetMap.
- використання: вибір потрібної локації для алгоритму генерації наборів.
- переваги: повністю безкоштовне використання необхідних API-функцій без обмежень.

Foundation Layout (Compose Foundation):

- призначення: додаткові компоненти для створення UI у Jetpack Compose.
- використання: побудова адаптивних макетів (рядки, колонки, сітки) та створення кастомних віджетів (кнопки, текстові поля, списки).
- переваги: оптимізована для продуктивності і підтримка Material Design та власних стилів.

JUnit4 + Espresso:

- призначення: бібліотеки для модульного та UI-тестування.
- використання: JUnit 4 - тестування невеличких функцій, Espresso - автоматизація тестування інтерфейсу (кліки, введення тексту, перевірка відображення елементів).
- переваги: стандартні інструменти Android для тестування.

4.2 Система навігації між екранами

Система навігації додатка реалізована за допомогою Navigation Compose, що забезпечує зручне перемикання між екранами та керування стеком навігації. Основними компонентами системи є Router та NavHost.

Router - центральний клас для керування навігацією, який інкапсулює NavController, відповідає за переходи між екранами та передачу даних і інтегрується з ViewModel для керування станом додатка:

```
// NAVIGATION
val navController = rememberNavController()
val router = remember {
    Router(
        navController,
        startDestination,
        listOf(
            userViewModel,
            clothingItemViewModel,
            outfitViewModel,
            statisticsViewModel,
            recommendationViewModel,
            storageViewModel
        )
    )
}
```

NavHost – контейнер, який розташований в елементі Scaffold, для відображення поточного екрану, що використовує rememberNavController() для збереження стану навігації, та здійснення переходу між екранами через (всі екрани зберігаються в мапі screens):

```
NavHost(
    navController = navController,
    startDestination = Screen.ClothesList.name,
    modifier = Modifier.padding(paddingValues)
) {
    screens.forEach { (route, screen) ->
        composable(route.name) {
            when (authState) {
                AuthState.Loading -> LoadingDisplay()
                is AuthState.Error -> ErrorDisplay(
                    authViewModel = authViewModel,
                    message = authState.message
                )
                else -> {
                    Box(
                        modifier = Modifier
                            .testTag("screen__${route.name}")
                    )
                    screen()
                }
```

```
                }
            }
        }
    }
```

Також варто зазначити, що ця система містить в собі систему перевірки авторизації, якщо користувач зареєстрований.

Система автоматично перевіряє стан авторизації (authState) при кожній зміні екрану:

```
LaunchedEffect(Unit) {
    navController.addOnDestinationChangedListener { _, _, _ ->
        authViewModel.profile(
            context = context
        )
    }
}
```

Під час процесу перевірки стану авторизації відображається індкатор завантаження (LoadingDisplay). При помилці перевірки автентифікації (немає підключення до інтернету, неможливість підключення до сервера або інші причини) показує екран помилки (ErrorDisplay), в якому додаток пропонує спробувати перевірити знову або вийти з облікового запису (точніше, видалити токен авторизації). Ці елементи вже можна було побачити в одному з показаних фрагментів коду в цьому підрозділі.

Користувач, який знаходиться у ролі гостя, не потребує перевірки на авторизацію.

Також всі екрани мають тегові ідентифікатори для UI-тестування:

```
Box(
    modifier = Modifier
        .testTag("screen__${route.name}")
)
```

4.3 Система синхронізації

Система синхронізації додатка реалізована для забезпечення узгодженості даних між локальним сховищем (SQLite через Room) та серверною частиною (REST API через Retrofit). Вона відповідає за стабільну роботу для авторизованих користувачів.

Для перевірки даних у кожного користувача в БД на сервері зберігається дата та час (у точності до мікросекунд), що являється часом останнього оновлення користувацьких даних гардеробу на сервері, що включає список речей та список наборів. Тож це значення часу грає роль «версії» даних.

Це часове значення змінюється при оновленні даних (додавання одягу або набору, оновлення характеристик одягу або набору, видалення одягу або набору), тож після виконання подібних операцій сервер оновлює «версію» та передає її на мобільний пристрій для локального зберігання для подальших перевірок на «актуальність».

При кожній перевірці стану авторизації користувача також порівнюються «версії» серверних та локальних даних. При їх незбігані додаток робить HTTP-запит на отримання теперішніх даних та теперішньої «версії», після чого додаток замінює їх в локальній БД.

Процес синхронізації може бути двох типів:

- завантаження даних (зображення, елементи гардеробу та набори) з мобільного пристрою на сервер, який може виконатись тільки при виборі користувача при авторизації, коли йому дається вибір між синхронізацією за локальними або серверними даними;
- завантаження даних з сервера на мобільний пристрій, який відбувається кожний раз при потребі оновлення локальних даних згідно серверних у випадку, коли при перевірці «актуальності» локальних даних система дізнається, що вони вже застарілі, і також, при виборі користувача при авторизації.

Варто зазначити, що для правильної роботи цієї системи було зроблено так, щоб авторизований користувач не міг редагувати дані без підключення до інтернету. Тож у цьому випадку йому потрібно буде або відновити підключення, або увійти у гостьовому режимі, після чого користувач зможе змінювати дані офлайн, але дані не будуть змінюватись та зберігатись на сервері в обліковому записі доки користувач знову не авторизується.

Синхронізація дуже важлива та корисна у випадках, коли обліковий запис використовується на більше, ніж одному пристрої, або при перенесенні даних з одного пристрою на інший.

4.4 Інтерактивна карта на основі OpenStreetMap та вибір локації

У додатку для управління гардеробом реалізовано інтерактивну мапу на основі OpenStreetMap Droid [9] (OSMDroid) – бібліотеки, призначеної для відображення та роботи з картами. Цей інструмент інтегрований у систему для покращення персоналізації генерації комбінацій одягу.

У цьому проєкті інтерактивна карта (рис. 4.1) відіграє роль інструменту для вибору локації, координати якої необхідні для повноцінного виконання генерації наборів, так як на сервері вони потрібні для визначення погоди в поточній локації в поточному часі.

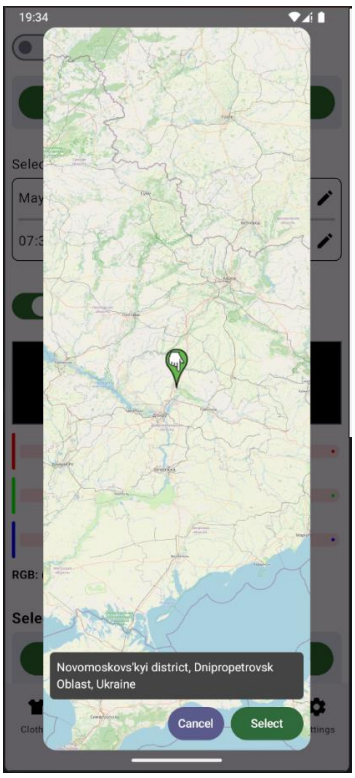

Рисунок 4.1 – Знімок екрану додатку для демонстрації використання інтерактивної карти (рисунок виконаний самостійно)

Дана функція являється необов’язковою, так як вона потрібна у випадку, якщо користувач відмовиться використовувати локацію свого мобільного пристрою для генерації.

Для зручної та гнучкої інтеграції інтерактивної карти в проєкт було створено елемент на основі об’єкту Composable з бібліотеки Jetpack Compose, програмний код якого зазначено в додатку Е.

Якщо ж користувач дозволить використовувати локацію пристрою, то буде використано GPS та інші вбудовані сервіси для знаходження приблизної локації:

```
val geoData = if (useCurrentLocation) {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
        val currentLocation =
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER)
            ?:
locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER
)
        if (currentLocation != null)
            GeoPoint(currentLocation.latitude,
currentLocation.longitude)
        else {
            okDialogTitle = errorMessageTitle
            okDialogMessage = locationNotFoundMessage
```

```
                return@Button
            }
        } else {
            val geoResult =
recommendationViewModel.getDeviceLocation(context)
            if (geoResult != null) geoResult
            else {
                okDialogTitle = errorMessageTitle
                okDialogMessage = locationNotFoundMessage
                return@Button
            }
        }
    } else location!!
```

4.5 Реалізація можливості зміни мови UI в самому додатку

Для забезпечення зручної зміни мови інтерфейсу безпосередньо в додатку було розроблено спеціальний механізм, принцип роботи якого зазначено нижче.

Система зміни мови реалізована через клас LanguageManager, який відповідає за зберігання обраної мови та її застосування. Основний функціонал включає:

- збереження вибору мови у SharedPreferences для забезпечення стійкості між запусками додатка;
- автоматичне визначення системної мови при першому запуску;
- підтримку як ручної зміни мови, так і автоматичного слідування за системними налаштуваннями.

Для роботи з сучасними версіями Android (API 33+) використовується вбудований LocaleManager, тоді як для старіших версій застосовується механізм оновлення конфігурації вручну. Це забезпечує коректну роботу на всіх підтримуваних версіях ОС. Нижче преведено код одного з методів класу LanguageManager, який призначений зміни мови додатку:

```
fun setAppLanguage(languageCode: String) {
    sharedPrefs.edit { putString(KEY_SELECTED_LANGUAGE, languageCode)
}

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
        context.getSystemService(
            LocaleManager::class.java).applicationLocales =
            if (languageCode == SYSTEM_DEFAULT_LANGUAGE) {
                LocaleList.getEmptyLocaleList()
            } else {
                LocaleList.forLanguageTags(languageCode)
```

```
            }
        } else {
            applyLegacyLanguage(languageCode)
        }

        if (context is Activity) {
            context.recreate()
        }
    }
```

Інтерфейсна частина реалізована у вигляді випадаючого меню з підтримкою Material Design 3. Користувач може обрати мову зі списку доступних варіантів, включаючи опцію "Системна мова". Після зміни мови автоматично відбувається перезавантаження активності для застосування змін до всього інтерфейсу.

Особлива увага приділена обробці крайових випадків:

- відновлення останньої обраної мови після перезапуску додатка;
- коректна реакція на зміну мови в системних налаштуваннях;
- плавна зміна інтерфейсу без видимих артефактів.

Реалізація була протестована на різних версіях Android та з різними комбінаціями мовних налаштувань, що підтвердило її стабільність та відсутність пам'яті про попередні стани.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Процес тестування [10] програмного забезпечення був реалізований з метою забезпечення стабільної роботи додатка, виявлення потенційних помилок та підтвердження відповідності системи вимогам користувачів.

Тестування проводилося за допомогою комбінації автоматизованих та ручних методів, що дозволило комплексно оцінити якість продукту. Основна увага приділялася UI-тестуванню для перевірки коректності роботи інтерфейсу і модульному тестуванню окремих компонентів системи.

Для автоматизованого тестування використовувалися такі інструменти:

- JUnit 4 [11] – для модульного тестування бізнес-логіки;
- Espresso [12] – для автоматизації UI-тестів.

Ручне тестування включало перевірку роботи додатка на різних пристроях з відмінними характеристиками (розміри екранів, версії Android від 7 до 15), а також тестування в умовах слабкого інтернет-з'єднання або його повної відсутності.

За допомогою модульного тестування було здійснено перевірку над невеличкими функціями, призначеними для правильного формату відображення інформації на пристрої.

Для перевірки працездатності основних функціональних вимог було використано автоматизоване UI-тестування:

- авторизація та реєстрація;
- управління списком речей;
- управління списком наборів;
- генерація наборів;
- локалізація;
- функції особистого налаштування користувача;
- навігація між сторінками.

Для інших менш важливих функції було використано мануальне тестування:

- перезапис забутого пароля користувача;
- тестування роботи та відображення інформації на різних пристроях.

Для полегшення процесу створення автоматизованих UI-тестів, деякі дії було занесено в функції для їх подальшого використання в тестах, так як вони вимагались для багатьох дій. Наприклад, дія авторизації, яка вимагається для багатьох UI-тестів. Вона включає в себе введення вказаних даних в поля та натискання кнопки:

```
fun authorize(
    email: String,
    password: String
) {
    assertExists("screen__${Screen.LogIn.name}")
    rule.onNodeWithTag(
        "email__input")
        .performTextInput(email)
    rule.onNodeWithTag(
        "password__input")
        .performTextInput(password)
    rule.onNodeWithTag(
        "apply__button")
        .performClick()
}
```

Також для більш чіткого та правильного тестування було створення функції, які враховують затримку (наприклад, час завантаження сторінок та інформації), після виконання деяких дій перед перевіркою отриманих результатів:

```
fun assertExists(
    tag: String
) {
    rule.waitUntil(DEFAULT_TIMEOUT) {
        rule.onAllNodesWithTag(tag)
            .fetchSemanticsNodes().isNotEmpty()
    }
}

fun assertExists(
    matcher: SemanticsMatcher
) {
    rule.waitUntil(DEFAULT_TIMEOUT) {
        rule.onAllNodes(matcher)
            .fetchSemanticsNodes().isNotEmpty()
    }
}

fun assertDoesNotExist(
    tag: String
) {
    rule.waitUntil(DEFAULT_TIMEOUT) {
        rule.onAllNodesWithTag(tag)
            .fetchSemanticsNodes().isEmpty()
    }
}
```

```
fun assertDoesNotExist(
    matcher: SemanticsMatcher
) {
    rule.waitUntil(DEFAULT_TIMEOUT) {
        rule.onAllNodes(matcher)
            .fetchSemanticsNodes().isEmpty()
    }
}
```

В результаті проведених тестів було виявлено та виправлено критичних помилок і близько 10 помилок середнього та низького рівня важливості. Серед найбільш значущих виправлень – оптимізація роботи при завантаженні та відображенні зображень, що зменшало об’єм даних в кеші при роботі з ними. Також тестування допомогло виявити різного роду помилки при роботі додатка на старих версіях Android.

ВИСНОВКИ

У ході виконання даної кваліфікаційної роботи було розроблено мобільний додаток для управління гардеробом, який поєднує у собі сучасні технології розробки та інноваційні підходи до організації персонального стилю.

Головною метою проекту було створення зручного інструменту, що допомагає користувачам ефективно керувати своїм гардеробом, створювати стильні образи та аналізувати свої уподобання.

Розроблений додаток має значний потенціал для подальшого вдосконалення. Перспективними напрямками розвитку є інтеграція з інтернет-магазинами одягу, розширення можливостей соціальної взаємодії (обмін наборами), а також вдосконалення алгоритмів рекомендацій за допомогою глибокого навчання.

Практична цінність роботи полягає у створенні готового до використання продукту, який може бути впроваджений у реальних умовах.

Варто відзначити, що дана робота демонструє ефективне поєднання сучасних технологій розробки мобільних додатків з інноваційними підходами до вирішення повсякденних завдань.

Отримані результати підтверджують, що поставлені цілі були успішно досягнуті, а розроблений додаток має всі необхідні якості для успішного функціонування на ринку мобільних додатків.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Stylebook Closet App [Електронний ресурс] – URL: https://stylebookapp.com/ (дата звернення: 05.04.2025)
2. ClosetSpace - Style Management [Електронний ресурс] – URL: https://closetspace.ua.aptoide.com/app (дата звернення: 05.04.2025)
3. Smart Closet: Your Personal Stylist [Електронний ресурс] – URL: https://smartcloset.me/ (дата звернення: 05.04.2025)
4. Григор’єв О. В. Корекція колірного балансу цифрового зображення на основі статистичних характеристик / О. В. Григор’єв, Т. А. Колесникова, Л. О. Яценко // Поліграфічні, мультимедійні та web-технології: колективна монографія. – Харків: ТОВ «Друкарня Мадрид», 2021. – С. 68-79.
5. Unified Modeling Language (UML) Diagrams | GeeksforGeeks [Електронний ресурс] – URL: https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/ (дата звернення: 13.04.2025)
6. Jetpack Compose UI App Development Toolkit - Android Developers [Електронний ресурс] – URL: https://developer.android.com/compose (дата звернення: 06.04.2025)
7. MVVM (Model View ViewModel) Architecture Pattern in Android | GeeksforGeeks [Електронний ресурс] – URL: https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/ (дата звернення: 08.04.2025)
8. Save data in a local database using Room | Android Developers [Електронний ресурс] – URL: https://developer.android.com/training/data-storage/room/ (дата звернення: 10.04.2025)
9. Android — OpenStreetMap Wiki [Електронний ресурс] – URL: https://wiki.openstreetmap.org/wiki/Android (дата звернення: 15.04.2025)
10. Сидоренко, А. О. Методи покращення якості тестування складних програмних систем : автореф. дис. ... канд. техн. наук : 05.13.06 / А. О. Сидоренко ; НТУУ "КПІ". – Київ, 2021. – 20 с.
11. JUnit – About | Junit [Електронний ресурс] – URL: https://junit.org/junit4

(дата звернення: 20.04.2025)

12. Espresso Android Developers [Електронний ресурс] – URL: https://developer.android.com/training/testing/espresso (дата звернення: 20.04.2025)
13. GitHub - oleh-maliuta/clothes-advisor-client-android [Електронний ресурс] – URL: https://github.com/NureMaliutaOleh/diploma-bachelor/tree/main/2025_%D0%91_%D0%9F%D0%86_%D0%9F%D0%97%D0%9F%D0%86-21-6_%D0%9C%D0%B0%D0%BB%D1%8E%D1%82%D0%B0_%D0%9E_%D0%92