ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА
(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій
(повне найменування факультету)

Кафедра інформаційних технологій
(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Бакалавр»

«Розробка мобільного застосунку для автоматизації інвентаризації матеріальних цінностей підприємства»
(тема кваліфікаційної роботи українською мовою)

«Development of a Mobile Application for the Enterprise Assets Inventory Automating»
(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання
спеціальності 122 Комп'ютерні науки
(код, назва спеціальності)
Освітня програма Комп'ютерні науки
(назва)
Бурлака Сергій Сергійович
(прізвище, ім'я, по-батькові здобувача)

Керівник д.т.н., професор Казакова Н.Ф.
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент к.т.н., Домаскін О.М.
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій .
№ 1 від 09 червня 2024 р.

Завідувачка кафедри
(підпис) КАЗАКОВА Надія .
(прізвище,ім'я)

Захищено на засіданні ЕК № 13 .
протокол № 11 від 20 червня 2024 р.

Оцінка добре / С / 75 .
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК
(підпис) КОПИЧЕНКО Іван .
(прізвище,ім'я)

Одеса 2024

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ОС – операційна система ПЗ – програмне забезпечення

ADT – Android Development Tools – Інструменти розробки в ОС Android

API – Application Programming Interface – програмний інтерфейс програми

IDE – Integrated Development Environment – інтегроване середовище розробки

SDK – Software Development Kit – набір засобів розробки

ВСТУП

У світі, який переживає війну, все більше популярності набувають види діяльності, що дозволяють виконувати завдання та отримувати інформацію з мінімальним особистим контактом з іншими людьми або через Інтернет за допомогою електронних пристроїв. Не варто забувати і про ефективне управління інвентарем. Під час війни багато бізнесів змушені працювати в безпечному режимі та впроваджувати нові технології для підтримки своєї діяльності.

Як швидко і безпечно отримати детальну інформацію про товар, який знаходиться перед очима? Можна звернутися до співробітника або спробувати знайти інформацію в Інтернеті. Однак, чи не буде зручніше просто навести камеру свого пристрою на штрих-код товару і відразу дізнатися всю необхідну інформацію? Саме тому розробка програми для сканування штрих-кодів і управління інвентарем є актуальною та необхідною.

Практично у кожної людини є смартфон, яким можна користуватися в будь-який час протягом дня. Тому розробка саме мобільного застосунку для управління інвентарем є актуальним завданням.

Метою кваліфікаційної роботи було створення мобільного застосунку "InventoryTrackerApp" для управління інвентарем з використанням технології сканування штрих-кодів.

Також у процесі розробки та ретельного аналізу було вирішено зробити WhiteLabel рішення з базовою програмою, яку буде кастомізовано під потреби користувача. Розробка базового мобільного додатку і є метою цього дипломного проекту.

Для досягнення поставленої мети були сформульовані наступні завдання:

– обґрунтувати вибір програмних засобів розробки та технологій;

– провести проектування системи з використанням мови UML;

– реалізувати функціонал аутентифікації користувачів за допомогою Firebase Authentication;

– розробити бізнес-логіку для управління інвентарем з використанням Firebase Firestore;

– інтегрувати бібліотеку Picasso для завантаження та відображення зображень продуктів;

– оптимізувати додаток для підвищення його продуктивності та зручності використання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПРОГРАМНИХ СИСТЕМ

1.1 Аналіз предметної області

Управління інвентарем є критично важливим аспектом діяльності будь-якого підприємства, що має справу з фізичними товарами. Від ефективного управління інвентарем залежить здатність компанії задовольнити попит клієнтів, зменшити витрати та оптимізувати операційні процеси. У цьому контексті проект InventoryTrackerApp спрямований на створення мобільного додатку, який допоможе малому та середньому бізнесу ефективно управляти своїм інвентарем. Було вирішено розробити InventoryTrackerApp як WhiteLabel рішення зі своєю звітною базою, що забезпечить керування доступами та використання як внутрішнього продукту компанії для ведення звітності не тільки товарів, а й інвентарю.

Основні компоненти управління інвентарем:

1. Облік запасів:

– Ведення точного обліку кількості товарів на складі.

– Відстеження руху товарів, включаючи надходження та витрати.

– Звітність не тільки товарів, а й інвентарю компанії.

2. Управління складом:

– Організація простору на складі для оптимального зберігання товарів.

– Відстеження місцезнаходження кожного товару та інвентарю.

3. Замовлення та поповнення запасів**:**

– Планування та здійснення замовлень на поповнення товарів та інвентарю.

– Взаємодія з постачальниками для своєчасної доставки товарів та інвентарю.

4. Відстеження продажів та використання інвентарю**:**

– Реєстрація продажів та оновлення запасів у реальному часі.

– Аналіз продажів для прогнозування попиту та планування закупівель.

– Відстеження використання інвентарю для внутрішніх потреб компанії.

5. Звітність та аналітика**:**

– Генерація звітів про стан запасів, продажі, використання інвентарю та інші ключові показники.

– Використання аналітики для прийняття обґрунтованих рішень щодо управління інвентарем та інвентарем.

Проблеми та виклики в управлінні інвентарем**:**

1. Точність даних:

– Недостатня точність даних про запаси та інвентар може призвести до перевитрат або дефіциту товарів та ресурсів.

– Помилки у введенні даних та людський фактор.

2. Своєчасне поповнення запасів:

– Несвоєчасне поповнення може призвести до втрати продажів та незадоволеності клієнтів.

– Відсутність ефективних інструментів для прогнозування попиту.

3. Відстеження руху товарів та інвентарю:

– Складність відстеження руху товарів та інвентарю у великих складах.

– Недостатня автоматизація процесів.

4. Інтеграція з іншими системами:

– Необхідність інтеграції з системами бухгалтерського обліку, продажів, логістики та управління інвентарем.

– Висока вартість та складність інтеграційних рішень.

5. Потреби користувачів:

Малий та середній бізнес:

– Простота використання та швидке налаштування системи.

– Доступність та прийнятна вартість рішення.

– Можливість масштабування та адаптації до змін потреб бізнесу.

6. Великі компанії:

– Необхідність у комплексному рішенні для управління не тільки товарами, а й інвентарем.

– Забезпечення безпеки даних та керування доступами.

– Інтеграція з існуючими внутрішніми системами та процесами.

7. Переваги WhiteLabel рішення:

– Брендування та кастомізація:

– Можливість адаптації під потреби різних компаній та брендів.

– Кастомізація інтерфейсу користувача та функціональності.

8. Контроль доступів:

– Гнучке налаштування прав доступу для різних ролей користувачів. Налаштування прав доступу реалізує Firebase Authentication.

– Забезпечення безпеки даних та захист від несанкціонованого доступу.

9. Звітність та аналітика:

– Розширені можливості для створення кастомних звітів та аналітики.

– Використання даних для прийняття стратегічних рішень.

InventoryTrackerApp як WhiteLabel рішення має значний потенціал для покращення управління інвентарем та інвентарем в малому та середньому бізнесі. Використання Firebase для управління даними та автентифікації, а також вибір Android Studio як основного середовища розробки, забезпечує ефективність, безпеку та гнучкість. Комплексний підхід до розробки, включаючи

управління доступами, кастомізацію та розширену звітність, робить InventoryTrackerApp ефективним інструментом для сучасного бізнесу.

1.2 Аналіз існуючих програмних систем

У сучасному світі ефективне управління інвентарем є критично важливим для багатьох підприємств, особливо для малого та середнього бізнесу. З розвитком технологій і мобільних додатків, можливості для поліпшення управління інвентарем стали більш доступними. У цьому контексті важливо розглянути існуючі системи, які пропонують аналогічні функції до нашого проекту InventoryTrackerApp, щоб зрозуміти їхні переваги та недоліки, а також обґрунтувати вибір конкретних технологій для розробки нашого додатку.

Аналіз систем:

1. Sortly

Опис: Sortly – це гнучка система управління інвентарем, призначена для малого та середнього бізнесу. Вона надає можливості для створення детальних каталогів продуктів, ведення обліку запасів і генерації звітів. Інтерфейс програми Sortly відображено на рисунку 1.1

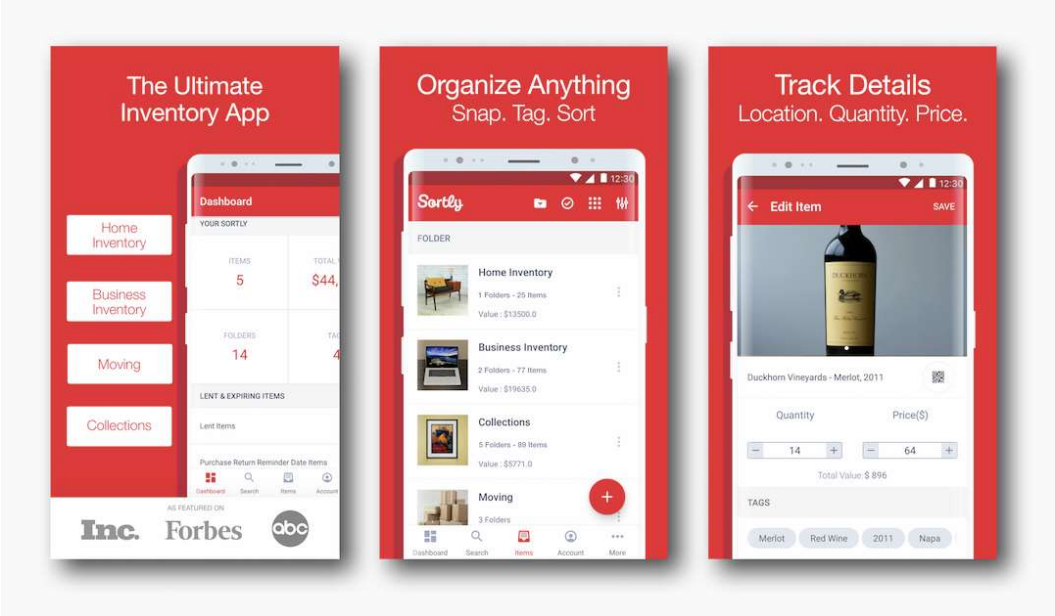


Рисунок 1.1 – Інтерфейс програми Sortly

Переваги:

– Інтуїтивний інтерфейс користувача
– Можливість додавання фотографій продуктів
– QR-коди для швидкого доступу до інформації про продукти
– Відстеження руху інвентаря

Недоліки:

– Обмежені функції безкоштовної версії

Відсутність глибокої інтеграції з іншими сервісами, такими як Firebase

2. Zoho Inventory

Опис: Zoho Inventory – це комплексна система управління інвентарем, що входить до екосистеми Zoho. Вона забезпечує повний набір інструментів для управління запасами, включаючи відстеження замовлень, управління складом і інтеграцію з іншими продуктами Zoho. Інтерфейс програми Zoho Inventory відображено на малюнку 1.2.

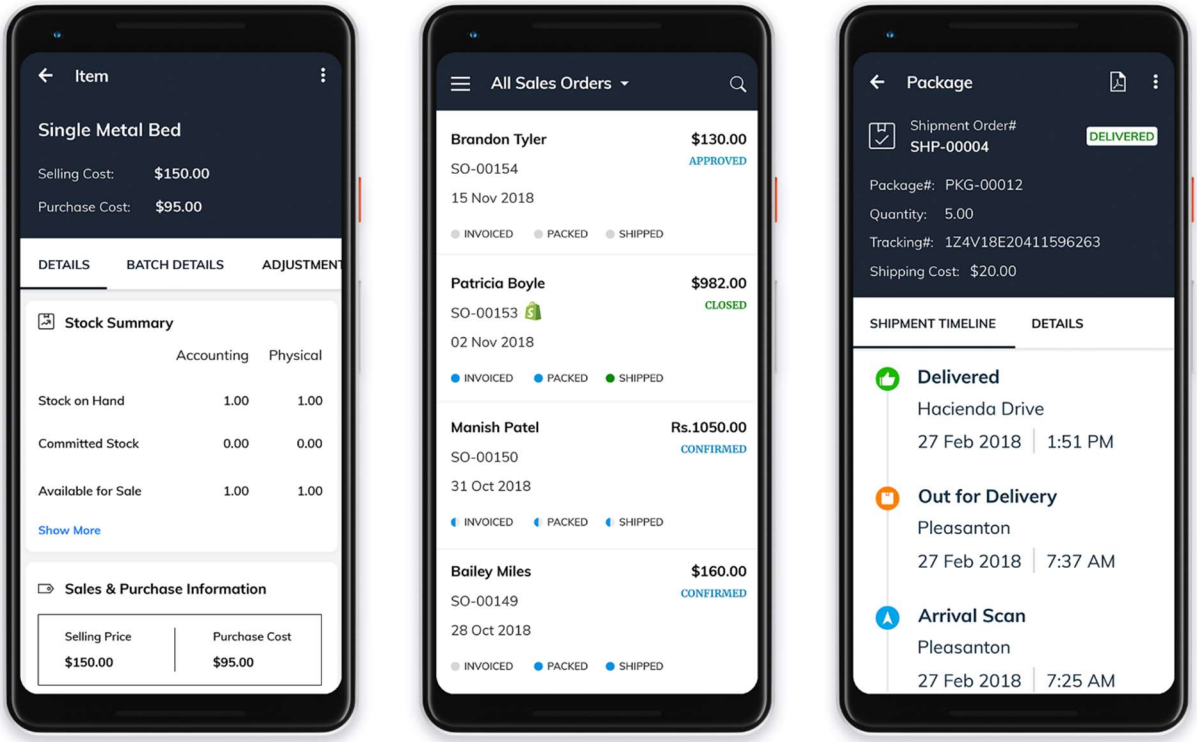


Рисунок 1.2 – Інтерфейс програми Zoho Inventory

Переваги:

– Інтеграція з іншими продуктами Zoho
– Підтримка мультиканального продажу
– Висока гнучкість і масштабованість
– Автоматизація бізнес-процесів

Недоліки:

– Складність налаштування для новачків
– Висока вартість для малого бізнесу
– Відсутність мобільної оптимізації для деяких функцій

3. Stock Control

Опис: Stock Control – це проста мобільна програма для управління запасами, що дозволяє користувачам створювати каталоги продуктів і відстежувати запаси. Інтерфейс програми Stock Control відображено на рисунку 1.3.

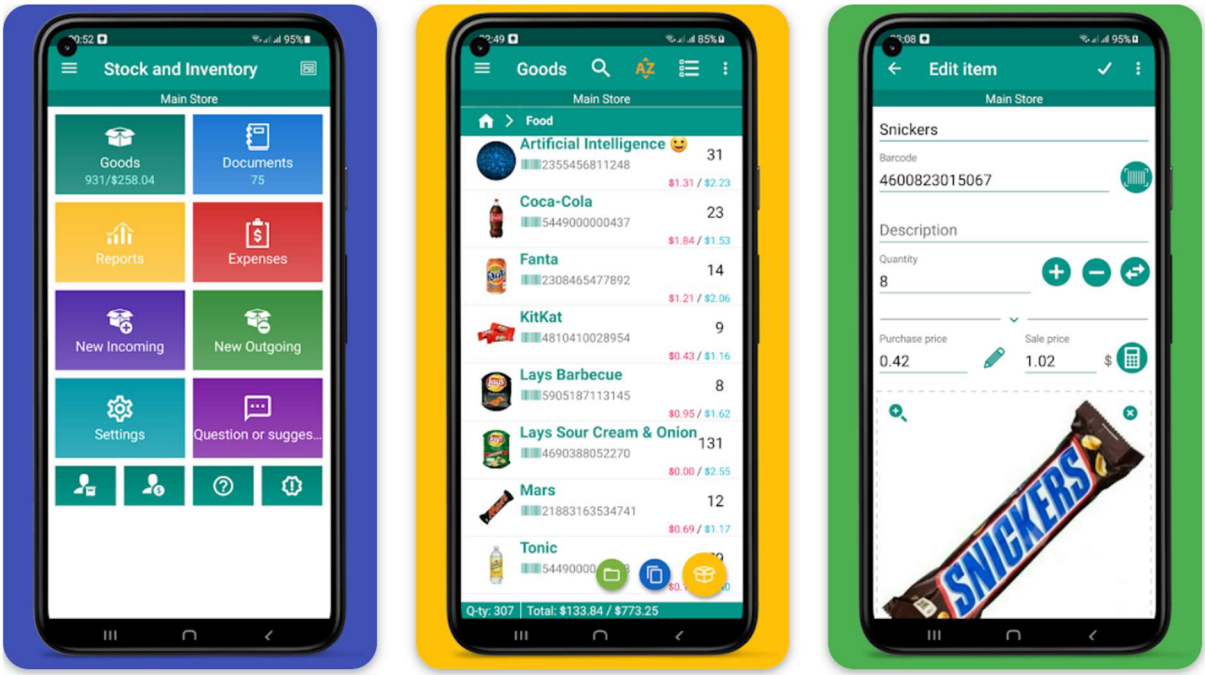


Рисунок 1.3 – Інтерфейс програми Stock Control

Переваги:

– Простий і зрозумілий інтерфейс
– Можливість створення декількох каталогів
– Підходить для особистого використання та малого бізнесу

Недоліки:

– Обмежені функції
– Відсутність інтеграції з зовнішніми сервісами
– Обмежені можливості для масштабування

4. Delivrd

Опис: Delivrd – це веб-додаток для управління інвентарем, який пропонує інструменти для відстеження запасів, управління замовленнями та створення звітів. Інтерфейс веб-додатку відображено на малюнку 1.4.

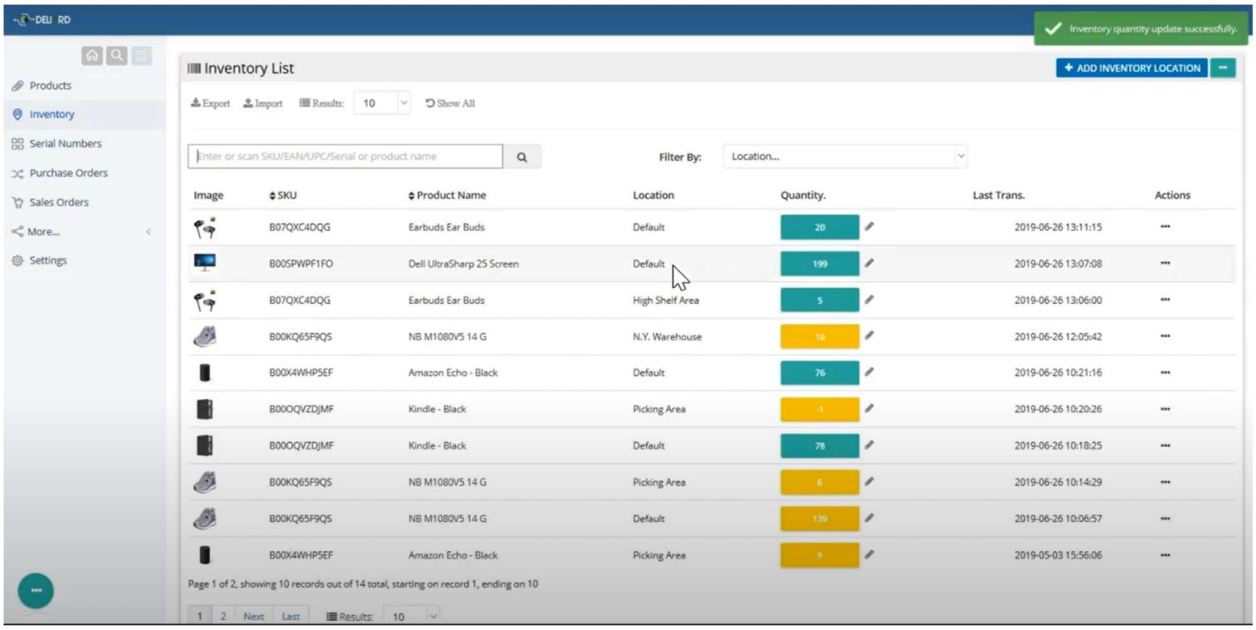


Рисунок 1.4 – Інтерфейс програми Delivrd

Переваги:

– Хмарна платформа з доступом з будь-якого пристрою
– Інтеграція з різними платформами електронної комерції

– Підтримка штрих-кодів для управління інвентарем

Недоліки:

– Відсутність мобільного додатку

– Обмежена функціональність у безкоштовній версії

– Залежність від інтернет-з’єднання

5. Inventory Now

Опис: Inventory Now – це проста програма для управління запасами, яка дозволяє користувачам відстежувати кількість продуктів та їхнє місцезнаходження. Інтерфейс програми Inventory Now відображено на малюнку 1.5.

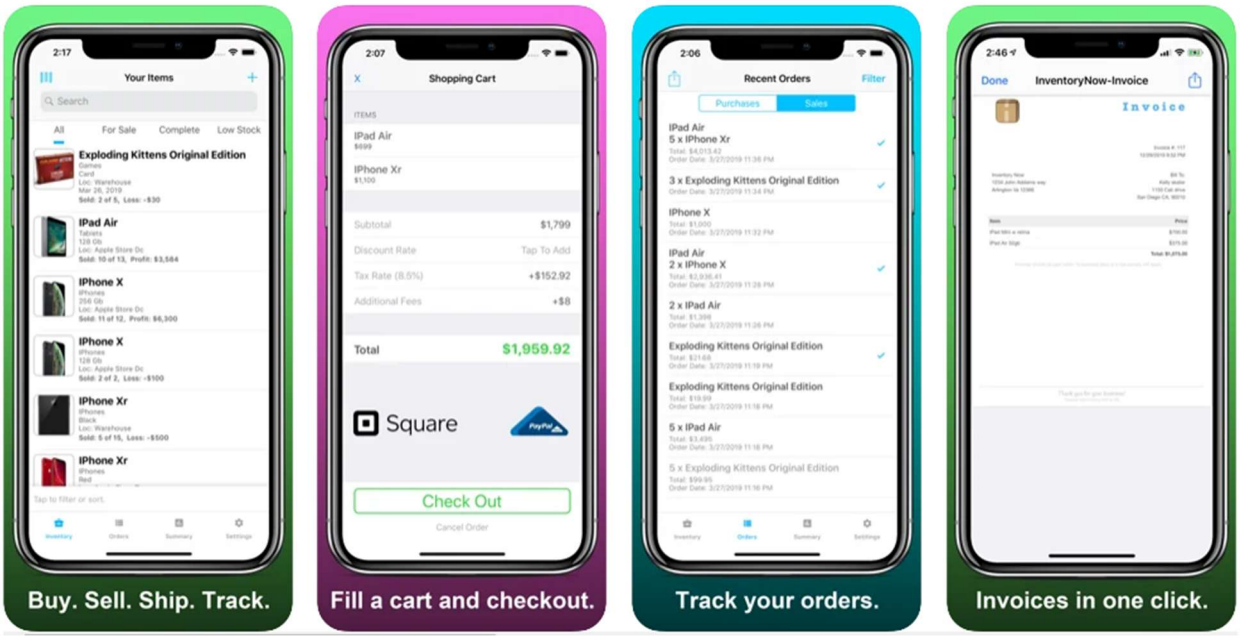


Рисунок 1.5 – Інтерфейс програми Inventory Now

Переваги:

– Легкий у використанні інтерфейс

– Мобільна оптимізація

– Підходить для малого бізнесу та особистого використання

Недоліки:

– Обмежені функції для великого бізнесу
– Відсутність інтеграції з іншими сервісами
– Обмежена аналітика та звітність

Схематичний підсумок можна побачити у таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз мобільних застосунків

Система	Інтерфейс користувача	Функціональність	Вартість
Sortly	Інтуїтивний	Середня	Дорога для преміум
Zoho Inventory	Складний для новачків	Висока	Висока
Stock Control	Простий	Обмежена	Низька
Delivrd	Середній	Середня	Середня
Inventory Now	Легкий	Обмежена	Низька

При розробці InventoryTrackerApp я врахував декілька ключових факторів, щоб забезпечити оптимальну функціональність, зручність використання та масштабованість нашої системи. Розглянувши існуючі системи, я вирішив використати бібліотеку Zxing для сканування штрих-кодів і сервіси Firebase (Firebase Firestore та Firebase Authentication) для зберігання даних та автентифікації користувачів.

Вибір Zxing:

– Простота інтеграції: Zxing легко інтегрується в мобільні додатки Android, що дозволяє швидко реалізувати функцію сканування штрих-кодів.

– Відкрите ПЗ: Zxing є відкритим програмним забезпеченням, що дозволяє нам модифікувати код для специфічних потреб нашого додатку.

– Широка підтримка: Zxing підтримує різні формати штрих-кодів, що забезпечує гнучкість у роботі з різними типами продуктів.

Вибір Firebase:

– Firebase Firestore: Забезпечує масштабоване та швидке сховище для даних з можливістю реального часу, що ідеально підходить для управління інвентарем.

– Firebase Authentication: Пропонує простий і безпечний спосіб автентифікації користувачів, що підвищує безпеку нашого додатку.

Ретельний аналіз існуючих систем управління інвентарем показав, що наш проект InventoryTrackerApp може вигідно відрізнятися від конкурентів завдяки вибору оптимальних технологій. Використання бібліотеки Zxing для сканування штрих-кодів забезпечує швидку та точну роботу з інвентарем, а інтеграція з Firebase Firestore та Firebase Authentication надає потужні можливості для зберігання даних та безпечної автентифікації користувачів. Це робить InventoryTrackerApp ефективним інструментом для малого та середнього бізнесу, який шукає надійне рішення для управління інвентарем у сучасному світі.

2. ОГЛЯД ІСНУЮЧИХ ТЕХНОЛОГІЙ ТА ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ

2.1 Функціональні та нефункціональні вимоги

До початку розробки застосунку були встановлені його функціональні та нефункціональні вимоги.

Функціональні вимоги:

– Автентифікація користувача – Користувач може увійти до додатку за допомогою облікового запису електронної пошти або Google.

– Реєстрація нового користувача – Користувач може зареєструватися, введучи необхідні особисті дані.

– Перегляд списку товарів – Користувач може переглядати список доступних товарів у додатку.

– Сканування штрих-кодів – Користувач може сканувати штрих-коди товарів для отримання детальної інформації про продукт.

– Огляд деталей товарів – Користувач може переглядати детальну інформацію про обраний товар.

– Оновлення інформації про товар – Користувач може змінювати і оновлювати інформацію про товари у базі даних.

Нефункціональні вимоги:

– Безпека – Додаток повинен забезпечувати безпеку особистих даних користувачів, використовуючи шифрування для зберігання і передачі даних.

– Швидкодія – Відгуки користувача на дії в додатку (наприклад, вхід у систему, сканування штрих-коду) повинні відбуватися без помітних затримок.

– Надійність – Додаток повинен бути стабільним і несприйнятливим до відмов, забезпечуючи неперервну роботу для користувачів.

– Сумісність – Додаток повинен підтримувати різні версії операційних систем Android і пристроїв, що підтримують ці версії.

– Локалізація – Додаток повинен підтримувати українську мову і мати інтерфейс, придатний для використання україномовними користувачами.

– Масштабованість – Система повинна бути готовою до масштабування, здатною обробляти зростаючу кількість користувачів і товарів без втрати продуктивності.

– Доступність – Додаток повинен бути доступним для використання з різних мережевих з'єднань і забезпечувати коректну роботу при втраті зв'язку.

Ці вимоги формують базову основу для розробки і впровадження Android додатку з використанням Firebase для автентифікації, управління продуктами та сканування штрих-кодів.

2.2 Мови програмування для розробки

На сьогоднішній день існує кілька основних мов програмування, які використовуються для розробки мобільних додатків. Кожна з них має свої особливості, переваги і сфери застосування.

Kotlin є сучасною мовою програмування, розробленою компанією JetBrains і підтримуваною Google для розробки Android-додатків [1]. Основні переваги Kotlin включають лаконічний синтаксис, високу продуктивність і безпеку коду. Вона має повну сумісність з Java, що дозволяє використовувати існуючий Java-код і бібліотеки, а також сприяє швидкій інтеграції нових функцій у додатки.

Java залишається однією з найпоширеніших мов програмування, яка використовується для Android-додатків. Вона відома своєю стабільністю, об'єктно-орієнтованим підходом і великим набором бібліотек. Java підтримується великою спільнотою розробників і залишається перспективною для створення складних додатків з різноманітною функціональністю.

Swift є мовою програмування, створеною Apple для розробки iOS- та macOS-додатків. Вона відрізняється високою продуктивністю, безпекою і простотою використання, що робить її популярною серед розробників. Swift підтримує сучасні концепції програмування і дозволяє створювати надійні та високоякісні додатки.

Objective-C, хоча поступово витісняється Swift, все ще використовується для розробки iOS-додатків, особливо в старих проектах. Вона надає потужні можливості для роботи з об'єктами та інтеграції з низькорівневими бібліотеками, що робить її цінною у певних випадках.

Вибір мови Kotlin для розробки мобільного застосунку "InventoryTrackerApp" був обґрунтований її сучасністю, підтримкою від Google і легкістю інтеграції з існуючим Java-кодом, що сприяє швидкому розвитку функціоналу додатка і забезпечує високу продуктивність [2].

Таким чином, вибір мови програмування визначається потребами проекту, екосистемою платформи та особистим досвідом розробника, забезпечуючи оптимальне поєднання між потужністю і ефективністю розробки.

2.3 Фреймворки та бібліотеки для мобільної розробки

Jetpack Compose є сучасним фреймворком для розробки користувацьких інтерфейсів в Android-додатках, що був розроблений командою Google. Він відзначається декларативним підходом до створення UI, що дозволяє описувати, як повинен виглядати інтерфейс, використовуючи прості композиційні функції, а не імперативно описувати кожен крок побудови інтерфейсу. Такий підхід значно спрощує розробку, поліпшує читабельність коду і зменшує ймовірність помилок.

Що стосується обрання AndroidX, це сучасна бібліотека, яка замінює попередні підходи до підтримки функцій Android, таких як управління життєвим

циклом, фрагменти, архітектурні компоненти тощо. Однією з ключових переваг AndroidX є те, що вона розділяє компоненти на окремі модулі, що полегшує і покращує розробку, тестування і підтримку додатків [3].

Обрання Jetpack Compose і AndroidX має декілька вагомих переваг:

– Сучасність і продуктивність: Jetpack Compose пропонує сучасний підхід до створення інтерфейсів, що підвищує продуктивність розробників і забезпечує більш якісний кінцевий продукт.

– Сумісність із наявним кодом: Jetpack Compose повністю сумісний з існуючим Android-кодом і бібліотеками, що робить його привабливим вибором для оновлення та розвитку існуючих додатків.

– Покращена підтримка функцій Android: AndroidX забезпечує зручну і покращену підтримку функцій Android, що дозволяє розробникам швидше і надійніше реалізовувати функціональність додатків.

– Активна підтримка та спільнота: Як продукти від Google, які активно розвиваються, Jetpack Compose і AndroidX мають активну спільноту розробників, яка допомагає вирішувати проблеми і вдосконалювати фреймворки.

Отже, вибір Jetpack Compose разом з AndroidX для розробки мобільних додатків – це стратегічне рішення, яке підтримує сучасні технології і забезпечує високу якість розробки, підтримку та масштабування додатків на платформі Android.

2.4 Середовища розробки

Розробка мобільних додатків, зокрема InventoryTrackerApp, є складним і багатогранним процесом, який вимагає правильного вибору середовища розробки. У даному випадку, Android Studio виокремлюється як оптимальний вибір, оскільки це офіційне середовище розробки для платформи Android, розроблене спеціально Google [4]. Воно забезпечує повний набір інструментів для

ефективної розробки, тестування та відладки додатків, що спрощує процес створення програмного забезпечення.

Однією з ключових переваг Android Studio є його інтеграція з Android SDK, що дозволяє розробникам легко використовувати всі потужні можливості платформи Android, такі як розширені API для роботи з геолокацією, камерою, базами даних тощо. Також Android Studio підтримує роботу з різними мовами програмування, зокрема Kotlin і Java, що робить його універсальним і гнучким інструментом для розробників з різними потребами та досвідом.

Порівнюючи з іншими середовищами розробки, такими як Eclipse і IntelliJ IDEA, Android Studio виходить на передній план завдяки своїм спеціалізованим інструментам для Android, які ретельно підтримуються Google. Eclipse, хоч і був популярним у минулому, сьогодні має обмежену підтримку і не отримує регулярних оновлень, що робить його менш привабливим для сучасних розробок. IntelliJ IDEA, з іншого боку, є потужним інструментом загального призначення, але не має такого глибокого фокусу на Android, як Android Studio.

Крім того, Android Studio відмінно інтегрується з іншими ключовими сервісами Google, такими як Firebase та Google Cloud Platform, що надає розробникам легкий доступ до хмарних послуг і інструментів для аналізу та моніторингу додатків. Це значно спрощує процес розробки та підтримки додатків після їх випуску.

Отже, вибір Android Studio для розробки InventoryTrackerApp обґрунтовується його спеціалізованими функціями для Android, широкою підтримкою від Google та інтеграцією з ключовими сервісами, що дозволяє зосередитися на створенні якісного і функціонального додатку для мобільних пристроїв.

3 ВИБІР ТА ОБГРУНТУВАННЯ ЗАСОБІВ РОЗРОБКИ

3.1 Технологія баркодів (штрихкодів)

Розробка та імплементація технологій штрихкодів (або баркодів) стали важливим елементом у сфері автоматизації і ідентифікації товарів, процесів та послуг. Штрихкоди використовуються практично в усіх секторах економіки, від роздрібної торгівлі та логістики до медицини і наукових досліджень. Вони спрощують і прискорюють процеси ідентифікації, збору даних та ведення обліку, що робить їх невід'ємною частиною сучасних технологічних рішень.

Історія штрихкодів сягає своїм корінням середини 20-го століття, коли почали активно розроблятися системи автоматичної ідентифікації товарів. Перші штрихкоди були реалізовані як прості чорно-білі смуги на паперовій основі, які зберігали обмежену кількість інформації. З розвитком технологій друку та сканування, штрихкоди стали більш універсальними і ефективними інструментами для автоматизації.

Технологічно штрихкоди складаються з кодування інформації за допомогою різноманітних символів, які можуть бути розпізнані за допомогою спеціалізованого обладнання (сканери штрихкодів).

Основними елементами технології штрих кодів є:

1. Символи штрихкодів: Штрихкоди представляють собою набір різних штрихових і проміжкових елементів, які кодують в собі цифрову або алфавітну інформацію. Існують різні типи символів штрихкодів, такі як штрихкод UPC (Universal Product Code), EAN (European Article Number), Code 39, Code 128 тощо [5].

2. Декодери та сканери: Для зчитування і інтерпретації інформації з штрихкодів використовуються спеціалізовані пристрої – сканери штрихкодів. Вони працюють на основі різних технологій, таких як лазерне сканування, CCD (Charge-Coupled Device) сканування або камери для сканування QR-кодів.

3. Стандарти і специфікації: Штрихкоди відповідають різним міжнародним індустріальним стандартам, які регулюють їхню конструкцію, розміри, типи кодування та розширення. Наприклад, ISO/IEC 15417 визначає стандарти для штрихкодів Code 128, а GS1 EAN/UPC стандартизує EAN і UPC штрихкоди для продуктів.

4. Ранні прототипи: Перші штрихкоди, створені в 1950-х роках, були прототипами сучасних систем автоматичної ідентифікації. Вони були використані в обмеженому обсязі для маркування товарів у магазинах та складах. Розвиток мікроелектроніки та комп'ютерної технології прискорив впровадження штрихкодів у широкий спектр застосувань.

Штрихкоди застосовуються у багатьох галузях, зокрема:

- Роздрібна торгівля: Для ідентифікації товарів, керування запасами і забезпечення точності ціноутворення.
- Логістика і постачання: Для відстеження руху товарів в ланцюжку постачання, управління інвентаризацією та оптимізації доставок.
- Медицина: Для ідентифікації пацієнтів, маркування проб та фармацевтичних продуктів.
- Транспорт і авіація: Для відстеження багажу, квитків та управління документацією.
- Наукові дослідження: Для маркування зразків, управління даними та ідентифікації обладнання.

З розвитком технологій штрихкоди продовжують еволюціонувати. З'являються нові формати, такі як QR-коди, які можуть зберігати більшу кількість інформації і забезпечують більші можливості для інтерактивної інформації та маркетингових кампаній. Також штрихкоди інтегруються з хмарними технологіями для зберігання додаткових даних і взаємодії з іншими системами.

У підсумку, штрихкоди залишаються однією з ключових технологій для автоматизації ідентифікації та збору даних у різних галузях. Вони продовжують еволюціонувати, забезпечуючи нові можливості.

3.2 Порівняльний аналіз бібліотек для сканування штрих-кодів

В сучасному світі використання технологій штрих-кодів є необхідною складовою багатьох сфер бізнесу, включаючи управління запасами, логістику, торгівлю та виробництво. Додаток InventoryTrackerApp, який розробляється для автоматизації обліку запасів, потребує ефективного механізму сканування штрих-кодів для забезпечення точності та ефективності у процесі інвентаризації та відстеження товарів.

При виборі бібліотеки для сканування штрих-кодів у InventoryTrackerApp було враховано кілька ключових критеріїв: продуктивність, точність розпізнавання, підтримка різних типів штрих-кодів, легкість інтеграції та підтримка від спільноти.

Ось порівняльний аналіз трьох популярних бібліотек: Zxing, Barcode Scanner SDK та ML Kit Vision API від Google.

1. Zxing

Опис: Zxing («Zebra Crossing») — це відкрите програмне забезпечення для сканування штрих-кодів та QR-кодів, розроблене на Java [6].

Переваги:

– Відкрите програмне забезпечення: Ліцензоване під Apache License, версія 2.0, що дозволяє безкоштовне використання та модифікацію.

– Підтримка різних форматів: Розпізнає широкий спектр стандартних штрих-кодів та QR-кодів, таких як EAN-8, EAN-13, Code 39, Code 128, ITF і багато інших.

– Легкість інтеграції: Легко інтегрується в Android додатки, має зрозумілу документацію та активну спільноту користувачів.

Недоліки:

– Точність: Хоча Zxing має добру точність для багатьох типів штрих-кодів, він може мати проблеми з розпізнаванням на пошкоджених або нерівномірних етикетках.

2. Barcode Scanner SDK

Опис: Barcode Scanner SDK — це комерційний інструмент, що надається різними виробниками для розпізнавання штрих-кодів.

Переваги:

– Професійна підтримка: Надається виробником з можливістю отримання технічної підтримки та регулярних оновлень.

– Висока точність: Забезпечує точне розпізнавання штрих-кодів навіть на складних підходах.

Недоліки:

– Вартість: Вимагає покупки ліцензії, що може збільшити загальні витрати на розробку додатків.

3. ML Kit Vision API (Google)

– Опис: Частина ML Kit від Google, яка використовується для розпізнавання штрих-кодів за допомогою машинного навчання.

Переваги:

– Машинне навчання: Застосування моделей машинного навчання дозволяє покращити точність розпізнавання штрих-кодів, включаючи пошкоджені або нерівномірні етикетки.

– Інтеграція з іншими сервісами Google: Легка інтеграція з Firebase для зберігання та обробки даних, що спрощує розробку додатків.

Недоліки:

– Залежність від мережі: Деякі функції можуть вимагати підключення до Інтернету для роботи, що може обмежувати функціональність у місцях з поганим зв'язком.

– Враховуючи потреби InventoryTrackerApp у швидкому та надійному скануванні штрих-кодів для точного ведення обліку запасів, було обрано Zxing.

Ось ключові міркування:

– Відкрите програмне забезпечення: Zxing має відкриту ліцензію, що дозволяє безкоштовне використання та модифікацію, що особливо важливо для проектів з обмеженим бюджетом.

– Підтримка різних форматів: Бібліотека розпізнає більшість стандартних штрих-кодів і QR-кодів, що забезпечує універсальність у застосуванні.

– Легкість інтеграції: Легко інтегрується з Android додатками, має зрозумілу документацію та активну спільноту користувачів, що спрощує процес розробки та підтримки.

– Підтримка спільноти: Активна спільнота розробників та користувачів Zxing забезпечує постійну підтримку та регулярні оновлення, що гарантує стабільну роботу бібліо теки в контексті оновлення платформи Android та забезпечує швидке виправлення помилок.

Хоча існують інші варіанти для сканування штрих-кодів, такі як комерційні рішення Barcode Scanner SDK та інтегровані можливості через ML Kit Vision API від Google, Zxing вирізняється своєю простотою використання, відкритістю програмного забезпечення та високою надійністю. Комерційні рішення можуть бути вибраними у випадках, коли потрібна додаткова підтримка або специфічні функції, які надає тільки платна версія. Однак, у більшості випадків, особливо для проектів з обмеженим бюджетом, відкрите програмне забезпечення, як Zxing, є оптимальним вибором.

Обрання Zxing для реалізації функціоналу сканування штрих-кодів у додатку InventoryTrackerApp є обгрунтованим рішенням, оскільки ця бібліотека відповідає всім ключовим вимогам проекту: вона надійна, швидка, має велику підтримку різних типів штрих-кодів, легко інтегрується з Android додатками та має активну спільноту розробників. Це забезпечить користувачам додатку зручність в роботі з інвентарем і підвищить ефективність ведення обліку запасів.

3.3 Firebase

Обгрунтування вибору Firebase Firestore та Firebase Authentication (Firebase Auth) для додатку InventoryTrackerApp базується на їхній потужності, гнучкості і можливостях, які ідеально відповідають вимогам сучасних мобільних додатків з управлінням інвентарем.

Характеристики та переваги Firestore:

– Firebase Firestore – це реальний час база даних, яка пропонує швидкий доступ до даних з будь-якого місця в Інтернеті. Ось кілька ключових характеристик та переваг Firestore.

– Реальний час оновлення: Firestore забезпечує миттєве оновлення даних між всіма користувачами додатку, що дозволяє реалізувати взаємодію в реальному часі. Це ідеально підходить для додатків, де важлива актуальність даних, таких як ведення інвентаря.

– Структурована база даних: Firestore зберігає дані у вигляді колекцій документів, що дозволяє легко організовувати та структурувати інформацію. Документи можуть містити підколекції, що дозволяє гнучко організовувати дані залежно від потреб додатку.

– Масштабованість: Firestore автоматично масштабується в залежності від потреб вашого додатку, обробляючи як малі обсяги, так і великі навантаження. Це особливо важливо для додатків зі значною кількістю користувачів або великим обсягом даних.

– Офлайн підтримка: Firestore надає можливість роботи офлайн, що означає, що користувачі можуть працювати з додатком із збереженими даними, навіть якщо вони не підключені до Інтернету. Це полегшує використання додатку у зоні поганого зв'язку або без доступу до Інтернету.

Інтеграція з іншими продуктами Firebase: Firestore ідеально інтегрується з іншими сервісами Firebase, такими як Firebase Authentication для автентифікації користувачів, Firebase Cloud Functions для обробки бізнес-логіки, Firebase Storage для зберігання файлів тощо [7].

Характеристики та переваги Firebase Authentication:

– Firebase Authentication – це сервіс, який забезпечує зручний і безпечний спосіб аутентифікації користувачів у вашому додатку. Ось ключові характеристики та переваги Firebase Authentication.

– Різноманітність методів аутентифікації: Firebase Authentication підтримує різні методи аутентифікації, включаючи електронну пошту та пароль, номер телефону, облікові записи Google, Facebook, Twitter тощо. Це дає користувачам можливість обирати зручний спосіб входу до додатку.

– Безпека: Firebase Authentication надає високий рівень безпеки засобами відновлення доступу, двофакторної аутентифікації та інших заходів захисту облікових записів користувачів.

– Простота інтеграції: Firebase Authentication інтегрується з Firestore та іншими сервісами Firebase, що полегшує розробку безпечних і сучасних додатків з можливістю керування доступом користувачів [8].

Вимоги додатку:

– InventoryTrackerApp потребує надійної системи зберігання даних, здатної обробляти як реальний час, так і офлайн сценарії, оскільки користувачі можуть працювати у різних місцях з обмеженим або непостійним Інтернет-з'єднанням. Firestore забезпечує реалізацію цих вимог завдяки реальному часу оновлення та офлайн підтримці.

– Firebase Authentication дозволяє безпечно автентифікувати користувачів за допомогою різних методів, що важливо для забезпечення зручності і безпеки входу у додаток. Інтеграція з Firestore дозволяє керувати доступом до даних, забезпечуючи безпеку і конфіденційність інформації про запаси.

Порівнюючи з іншими рішеннями, такими як власний серверний бекенд або інші бази даних, Firebase пропонує значно менше часу на налагодження та обслуговування, оскільки весь інфраструктурний стек Firebase керується Google і має готові засоби моніторингу, аналізу та масштабування.

Обрання Firebase Firestore та Firebase Authentication для InventoryTrackerApp є стратегічним рішенням, оскільки вони забезпечують потужність, масштабованість, надійність і безпеку, необхідні для сучасного мобільного додатку з управління інвентарем. Ці інструменти дозволяють зосередитися на функціональності додатку, забезпечуючи при цьому ефективне використання ресурсів і швидкий в розгортання нових функцій. Firebase Firestore дозволяє легко організовувати дані в колекції та документи, що ідеально підходить для сховища інформації про товари, їх кількість та інші характеристики, які потрібно відстежувати в InventoryTrackerApp.

Крім того, Firebase Firestore має вбудовану підтримку для реального часу, що означає, що всі зміни даних відразу відображаються для всіх користувачів додатку. Це дозволяє забезпечити актуальність інформації та миттєве оновлення стану інвентаря без затримок чи синхронізацій. Наприклад, якщо один користувач змінює кількість товару, інші користувачі одразу бачать ці зміни, що особливо важливо для колективної роботи або управління групами товарів.

Firebase Authentication в свою чергу забезпечує безпечний доступ користувачів до додатку, використовуючи різноманітні методи аутентифікації. Вона підтримує електронну пошту та пароль, номер телефону, облікові записи Google, Facebook та інші, що дозволяє користувачам вибирати найбільш зручний метод. Це не тільки полегшує вхід до додатку, а й забезпечує високий рівень безпеки за допомогою двофакторної аутентифікації та інших захисних механізмів.

Вибір Firebase Firestore і Firebase Authentication також зумовлений їхньою високою масштабованістю та гнучкістю. Firebase забезпечує автомати-

чне масштабування, що означає, що додаток може легко розширюватися з ростом користувацької бази та обсягів даних. Це дозволяє уникнути проблем зі швидкодією і доступністю, які можуть виникнути при використанні традиційних баз даних.

Наприклад, при підключенні нових складів або збільшенні асортименту товарів, Firestore автоматично масштабується, щоб забезпечити стабільну роботу додатку без необхідності вручну налаштовувати серверні ресурси. Це особливо важливо для мобільних додатків, де прискорення розробки та підтримка швидкодії є критичними факторами успіху.

У контексті InventoryTrackerApp, який має задачу надійного ведення інвентарю, забезпечення швидкого доступу до актуальних даних та високої безпеки обробки інформації про товари, Firebase Firestore та Firebase Authentication виявляються ідеальними рішеннями. Вони дозволяють зосередитися на розробці функціоналу додатку, не витрачаючи час на вирішення інфраструктурних завдань, що робить їх вибір стратегічним для досягнення бізнес-цілей проєкту.

4 ПРОЕКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

Після завершення аналізу предметної області, вивчення програмних продуктів-аналогів та вибору інструментів для розробки, наступним необхідним етапом є проектування інформаційної системи. На цьому етапі слід створити UML-діаграми, що включають діаграми класів, варіантів використання (прецедентів), активності та послідовності. Крім того, необхідно розробити макети інтерфейсу застосунку, які допоможуть визначити структуру і функціональність майбутнього програмного забезпечення.

Діаграми варіантів використання дозволяють візуалізувати взаємодію користувачів із системою, визначаючи основні сценарії її використання. Діаграми активності деталізують робочі процеси та алгоритми, що відбуваються всередині системи, забезпечуючи чітке розуміння послідовності дій та умов переходів між ними. Діаграми послідовності описують обмін повідомленнями між об'єктами в певній послідовності, що дозволяє точно моделювати взаємодію між компонентами системи [9].

Макети інтерфейсу користувача є важливим аспектом проектування, оскільки вони надають візуальне уявлення про зовнішній вигляд та функціональність інтерфейсу. Це допомагає не лише візуалізувати кінцевий продукт, але й проводити попереднє тестування та збирати зворотний зв'язок від користувачів, що сприяє покращенню дизайну та юзабіліті застосунку.

Проектування інформаційної системи є критичним етапом, який закладає фундамент для подальшої розробки та успішного впровадження програмного продукту.

4.1 Спрощена UML – діаграма класів застосунку

Для відображення усіх діаграм цього проєкту я буду використовувати інструмент під назвою PlantUML [10].

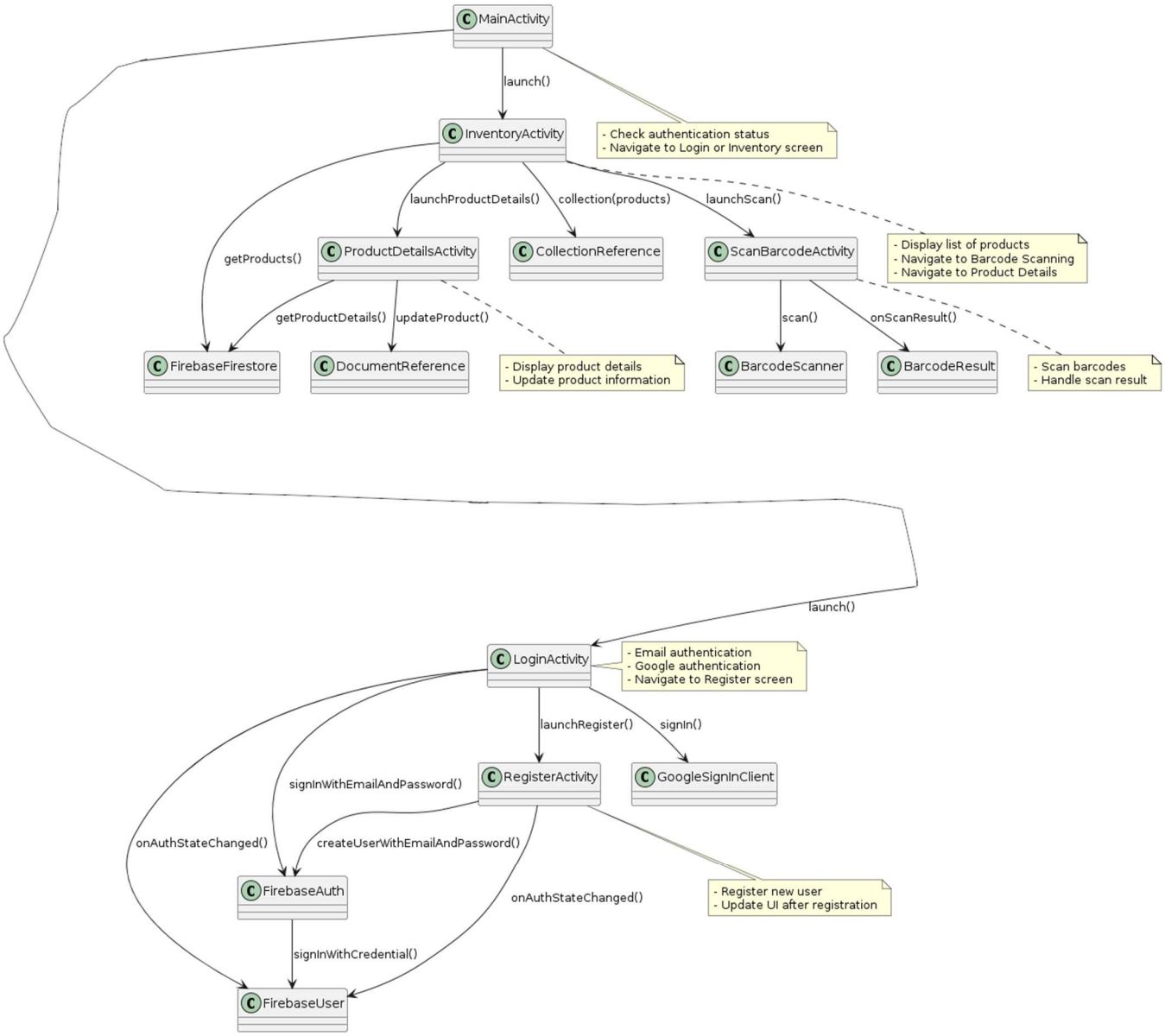


Рисунок 4.1 Спрощена UML – діаграма класів застосунку

На рисунку 4.1 було зроблено базову архітектуру застосунку далі я більш детальніше розповім про кожний елемент системи.

MainActivity:

– Перевірка статусу аутентифікації.

– Перехід на екран логіна або інвентарю.

LoginActivity:

– Аутентифікація через email і Google.
– Перехід на екран реєстрації.

RegisterActivity:

– Реєстрація нового користувача.
– Оновлення UI після реєстрації.

InventoryActivity:

– Відображення списку товарів.
– Перехід на сканування штрих-коду.
– Перехід до деталей товару.

ScanBarcodeActivity:

– Сканування штрих-кодів.
– Обробка результату сканування.

ProductDetailsActivity:

– Відображення деталей товару.
– Оновлення інформації про товар.

Використовувані класи Firebase:

– FirebaseAuth: для аутентифікації.
– FirebaseUser: для роботи з користувачами.
– GoogleSignInClient: для аутентифікації через Google.
– FirebaseFirestore: для взаємодії з базою даних Firestore.
– DocumentReference: для роботи з документами.
– CollectionReference: для роботи з колекціями.

Класи для сканування баркодів:

– BarcodeScanner: для сканування штрих-кодів.
– BarcodeResult: для обробки результатів сканування.

4.2 Створення UML-діаграми варіантів використання системи

Наступним етапом є Створення UML-діаграми варіантів використання системи я виконав цю задачу на рис 4.2

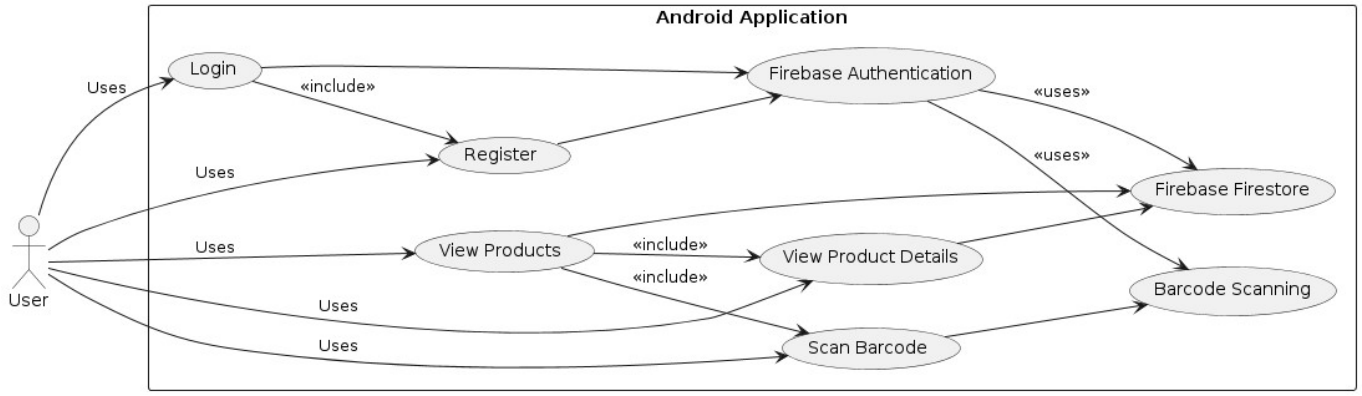


Рисунок 4.2 – Діаграма варіантів використання (прецедентів) системи

Актори:

– User: Користувач вашого Android додатка.

Варіанти використання (Use Cases):

– Login (UC1): Аутентифікація користувача.

– Register (UC2): Реєстрація нового користувача.

– View Products (UC3): Перегляд списку продуктів.

– Scan Barcode (UC4): Сканування штрих-коду.

– View Product Details (UC5): Перегляд деталей продукту.

Зв'язки:

– Uses: Відображає використання кожним варіантом використання актором.

– Include: Показує, що одні варіанти використання включають інші (наприклад, перегляд продуктів включає сканування баркодів і перегляд деталей продукту).

Компоненти системи:

– Firebase Authentication: Компонент для аутентифікації користувачів через Firebase.

– Firebase Firestore: Компонент для зберігання і доступу до даних про продукти.

– Barcode Scanning: Компонент для сканування штрих-кодів.

Ця діаграма дає змогу легко зрозуміти, як користувачі взаємодіятимуть із вашим застосунком, використовуючи різні функції, що надаються Firebase і сканування баркодів.

4.3 Створення діаграми діяльності (активностей)

Для моделювання виконання операцій у мові UML використовуються діаграми активності. Ці діаграми дозволяють візуалізувати послідовність дій або переходів від однієї активності до іншої, зосереджуючи увагу на кожному кроці та його результаті. Результат кожної активності може впливати на зміну стану системи або повернення певного значення, що робить діаграми активності потужним інструментом для аналізу та проектування процесів.

Для проектованої системи була створена діаграма активності, яка детально відображає основний процес – сканування зображення і його подальше розпізнавання. Це включає в себе всі необхідні кроки, від початкового сканування до обробки та аналізу отриманих даних, що сприяє кращому розумінню та оптимізації роботи системи. Я виконав цю задачу на рисунку 4.3.

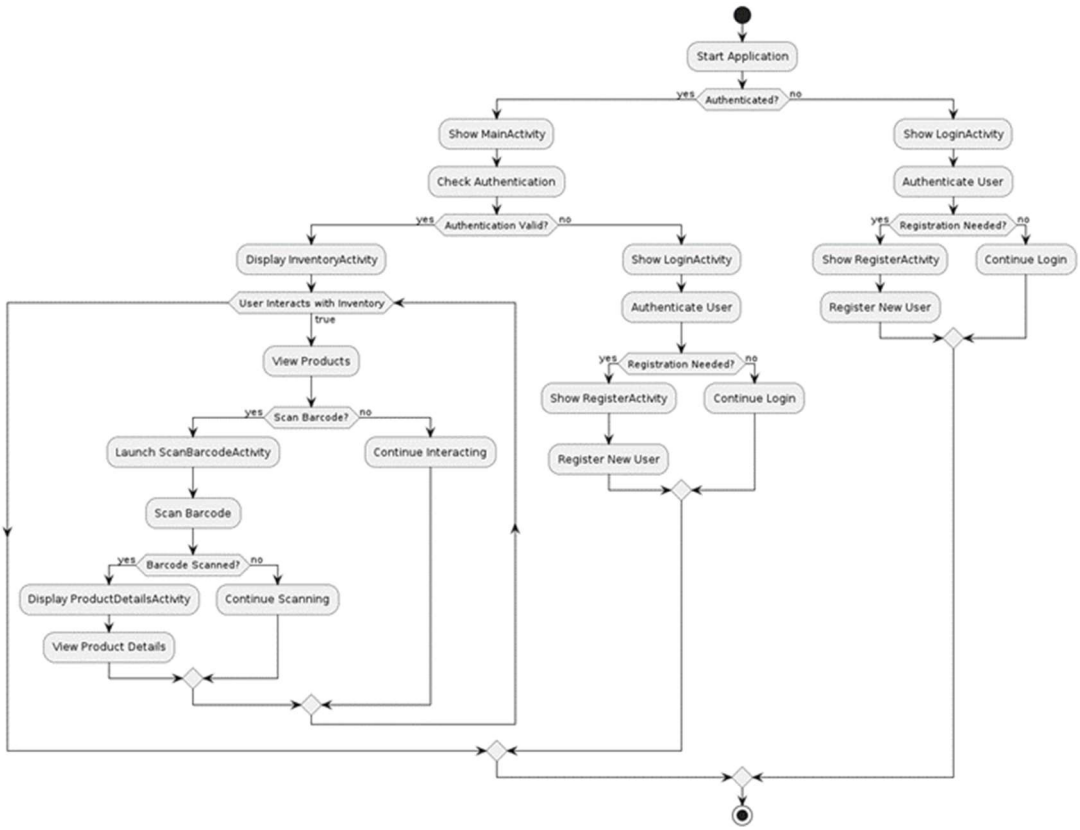


Рисунок 4.3 – Діаграма діяльності (активностей)

Актор:

– User: Користувач вашого Android додатка.

Активності:

– Start Application: Початок програми.

– MainActivity: Основна активність програми, що відображається після аутентифікації.

– InventoryActivity: Активність для перегляду списку товарів і управління ними.

– ScanBarcodeActivity: Активність для сканування штрих-кодів.

– ProductDetailsActivity: Активність для перегляду деталей продукту.

– LoginActivity: Активність для входу користувача.

– RegisterActivity: Активність для реєстрації нового користувача.

Потік дій:

– Користувач починає додаток.

– Якщо користувач аутентифікований, відображається MainActivity.

– У MainActivity перевіряється статус аутентифікації:

– Якщо аутентифікація пройшла успішно, відображається InventoryActivity.

В InventoryActivity користувач може переглядати продукти та взаємодіяти з ними:

– Якщо користувач вирішує сканувати баркод, запускається ScanBarcodeActivity.

– Після сканування штрих-коду відображається ProductDetailsActivity для перегляду деталей продукту.

– Якщо статус аутентифікації не підтверджено, користувач перенаправляється на LoginActivity для аутентифікації.

– За необхідності користувач може зареєструватися через RegisterActivity.

– Якщо користувач не аутентифікований під час старту програми, відкривається LoginActivity.

– За необхідності користувач може зареєструватися через RegisterActivity.

Ця діаграма показує основний потік дій користувачів у вашому застосунку, включно з автентифікацією, переглядом і керуванням продуктами, а також скануванням штрих-кодів.

4.4 Створення діаграми послідовності

Крім того, була створена діаграма послідовності. Ці діаграми використовуються для деталізації діаграм прецедентів і більш точного опису послідовностей дій у сценаріях використання. Вони дозволяють візуально представити взаємодію між об'єктами або компонентами системи в певній послідов-

ності, що допомагає розробникам краще зрозуміти порядок виконання операцій та взаємодії між ними. Таким чином, діаграми послідовності є важливим інструментом для деталізації та оптимізації архітектури програмних систем. Я виконав цю роботу на рисунку 4.4.

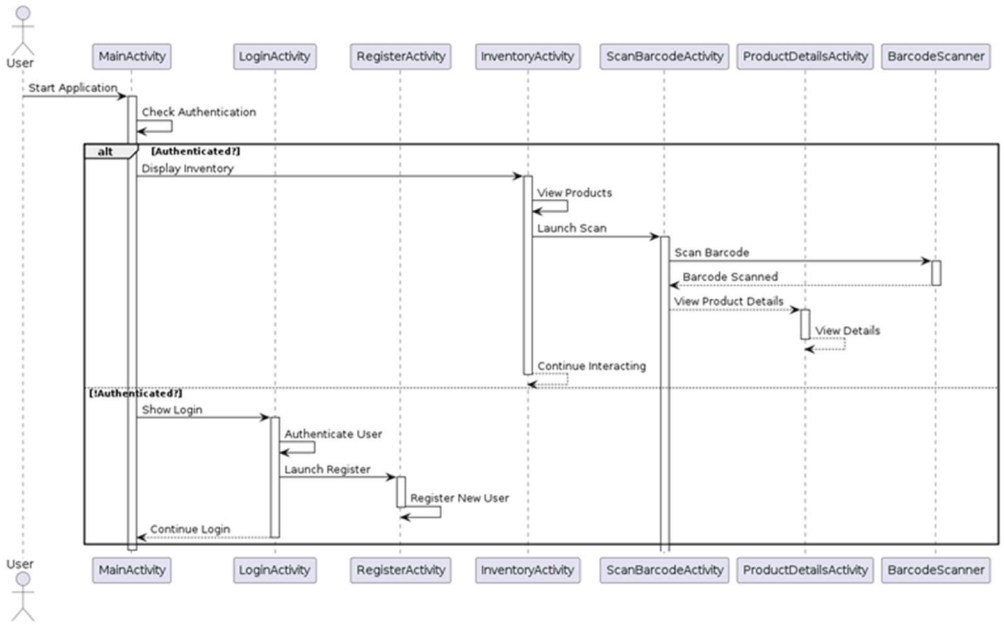


Рисунок 4.4 – Діаграма послідовностей

Актор:

– User: Користувач вашого Android додатка.

Учасники:

– MainActivity: Основна активність програми.

– LoginActivity: Активність для входу користувача.

– RegisterActivity: Активність для реєстрації нового користувача.

– InventoryActivity: Активність для управління інвентарем і продуктами.

– ScanBarcodeActivity: Активність для сканування штрих-кодів.

– ProductDetailsActivity: Активність для перегляду деталей продукту.

Послідовність дій:

– Користувач запускає додаток.

– MainActivity перевіряє статус аутентифікації користувача.

Якщо користувач аутентифікований, відображається InventoryActivity:

– Користувач переглядає список продуктів.

– Користувач запускає сканування штрих-коду в ScanBarcodeActivity.

– Після сканування штрих-коду відображаються деталі продукту в ProductDetailsActivity.

Якщо користувач не аутентифікований, відображається LoginActivity:

– Користувач аутентифікується або реєструється через RegisterActivity.

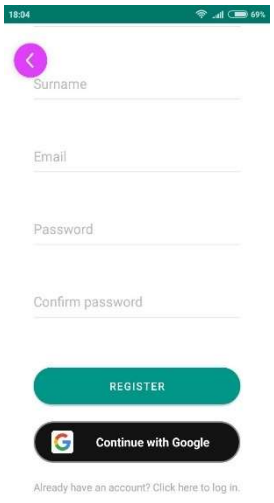
– Взаємодія завершується після виконання потрібних дій.

Ця діаграма допомагає зрозуміти послідовність взаємодії між різними компонентами застосунку залежно від статусу автентифікації користувача.

4.5 Розробка графічного прототипу застосунку

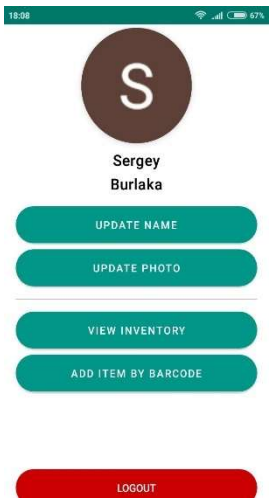
У процесі розробки було розроблено о стек графічних елементів та повний графічний прототип застосунку.

а)

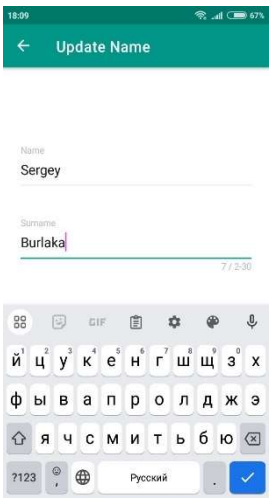


б)

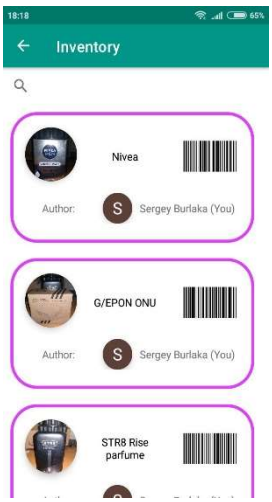
в)



г)



ґ)



д)

е)

є)

Рисунок – 4.5 Графічний прототип застосунку

На рисунку 4.5 відображено графічний прототип застосунку далі я наддам пояснення щодо окремих частин графічного прототипу.

Розбір частин графічного прототипу застосунку:

а) LoginActivity – Активність для входу користувача

На цьому етапі користувач вводить свої дані для входу в додаток, використовуючи електронну пошту або Google аккаунт. Він також має можливість вибору методу входу.

б) RegisterActivity – Активність для реєстрації нового користувача.

Ця активність дозволяє користувачеві створити новий обліковий запис у додатку, введенням необхідних особистих даних, таких як ім'я, електронна пошта і пароль. Після успішної реєстрації користувач отримує можливість використовувати свій обліковий запис для входу.

в) Вибір Google аккаунту користувача

Цей екран дозволяє користувачеві вибрати акаунт Google, з яким він хоче увійти в додаток. Користувач має можливість вибрати один з наявних Google аккаунтів, після чого він буде перенаправлений до авторизації через Google.

г) MainActivity – Основна активність програми

Ця активність відображається після успішного входу користувача в додаток. Основна активність зазвичай містить основний інтерфейс програми, який може включати загальну навігацію, можливість перегляду основних функцій додатку і доступ до інших ключових елементів інтерфейсу.

ґ) RegisterActivity – у режимі оновлення інформації користувача

Ця частина RegisterActivity передбачає можливість оновлення існуючої інформації користувача, такої як ім'я, електронна пошта і пароль. Вона використовується для того, щоб користувач міг змінити свої особисті дані в будь-який момент.

д) InventoryActivity – активність для управління інвентарем і продуктами.

Ця активність призначена для користувача для управління інвентарем та продуктами. Вона надає можливість перегляду списку доступних товарів, їх редагування, видалення і додавання нових товарів. Користувач може також використовувати функціонал для швидкого доступу до інформації про товари і їх характеристики.

е) ScanBarcodeActivity – Активність для сканування штрих-кодів.

Ця активність дозволяє користувачеві сканувати штрих-коди товарів за допомогою камери пристрою. Після успішного сканування система отримує

інформацію з штрих-коду, що використовується для ідентифікації конкретного продукту. Користувач може побачити результати сканування і перейти до перегляду деталей продукту.

є) ProductDetailsActivity: Активність для перегляду деталей продукту.

Ця активність відображає деталі конкретного продукту, який користувач обрав для перегляду. Вона містить інформацію про характеристики товару, такі як назва, опис, ціна, зображення тощо. Користувач може детально ознайомитися з усією доступною інформацією про товар і виконувати дії, наприклад, додавати товар до кошика покупок або редагувати його параметри.

5 РЕАЛІЗАЦІЯ МОБІЛЬНОГО ЗАСТОСУНКУ «InventoryTrackerApp»

Проект «InventoryTrackerApp» розробляється з метою створення мінімального життєздатного продукту (MVP) для ведення інвентаризації товарів. Додаток має на меті допомогти користувачам в управлінні інвентарем шляхом додавання, редагування та видалення продуктів, а також забезпечити можливість сканування штрих-кодів для швидкого доступу до детальної інформації про товари. Враховуючи що проект «InventoryTrackerApp» е базовим WhiteLabel додатком етап інтеграції Firebase Authentication включно з Firebase Firestore будуть виконуватися при реалізації окремо для кожної компанії заказника.

5.1 Інтеграція Firebase в додаток «InventoryTrackerApp»

Далі я опишу процес інтеграції Firebase у свій додаток покроково.

Крок 1: Створення проекту Firebase.

Створення проекту Firebase:

– Увійшов на консоль Firebase та створив новий проект під назвою "InventoryTrackerApp".

Налаштування проекту:

– Вибрав країну/регіон (евро регіон) для розгортання сервера Firebase.

Крок 2: Додавання додатку Android до проекту Firebase.

Додавання додатку до Firebase проекту:

– Додав додаток Android з пакетом com.burlaka.inventory.tacker.app до проекту Firebase.

Налаштування google-services.json:

– Завантажив файл google-services.json та додав його до каталогу app/ в моєму проекті Android Studio.

Крок 3: Налаштування Firebase Authentication.

Активація Firebase Authentication:

– У консолі Firebase налаштував Authentication для додатку "InventoryTrackerApp".

– Включив методи входу Email/Password та Google Sign-In.

Крок 4: Налаштування Firebase Firestore.

Активація Firebase Firestore:

– Створив базу даних Firestore для зберігання даних про інвентар (рисунок 5.1).

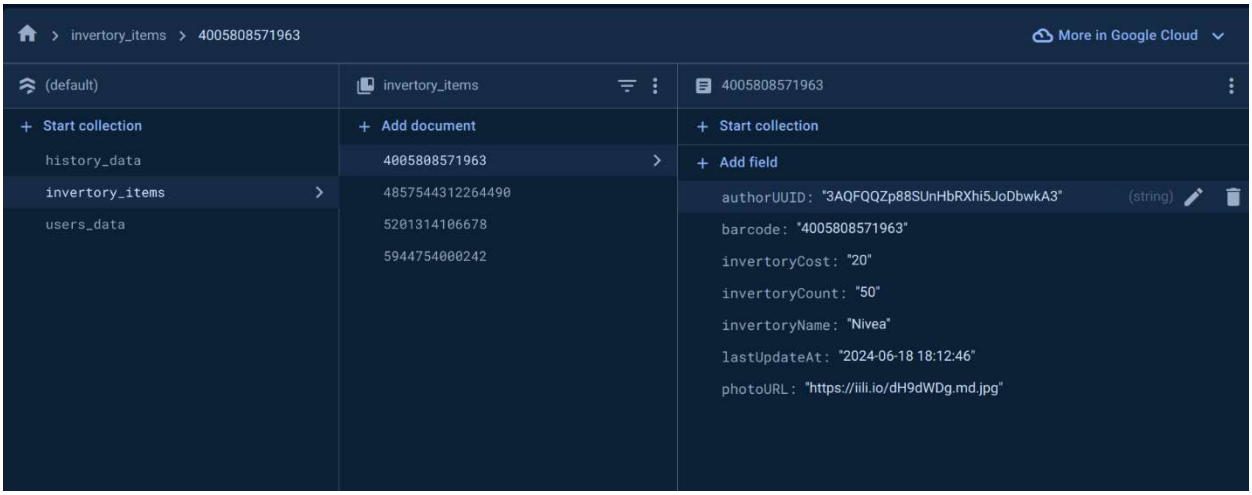


Рисунок 5.1 – База даних Firestore

Налаштування прав доступу:

– Налаштував права доступу до бази даних Firestore для забезпечення безпеки і конфіденційності даних.

Крок 5: Перевірка налаштувань.

Підтвердження налаштувань:

– Переконався, що всі налаштування Firebase (Authentication, Firestore) коректно налаштовані для додатку "InventoryTrackerApp". Я орієнтувався на дані дашборду Firebase (рисунок 5.2).

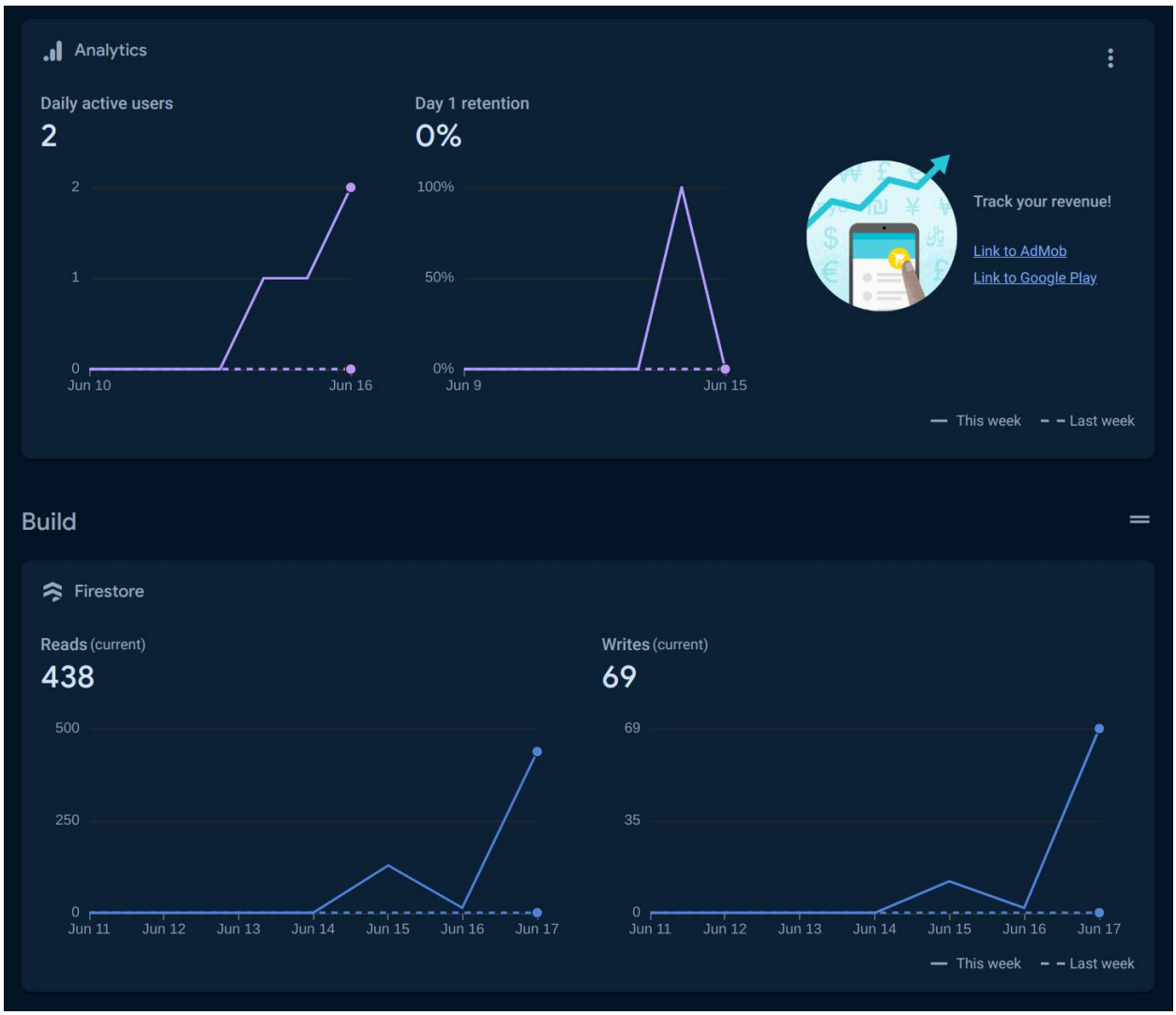


Рисунок 5.2 – дашборд Firebase

Інтеграція Firebase в додаток "InventoryTrackerApp" була успішно завершена. Firebase забезпечує основу для автентифікації користувачів та зберігання даних про інвентар, що дозволяє зосередитися на розвитку функціоналу додатку та забезпечити його стабільність та надійність.

5.2 Реалізація активностей у додатку

1. LoginActivity

Створення інтерфейсу:

– Розроблено активність для входу користувача з полями для введення електронної пошти та пароля, а також кнопками для входу через електронну пошту та Google. Процес створення відображено на рисунку 5.3.

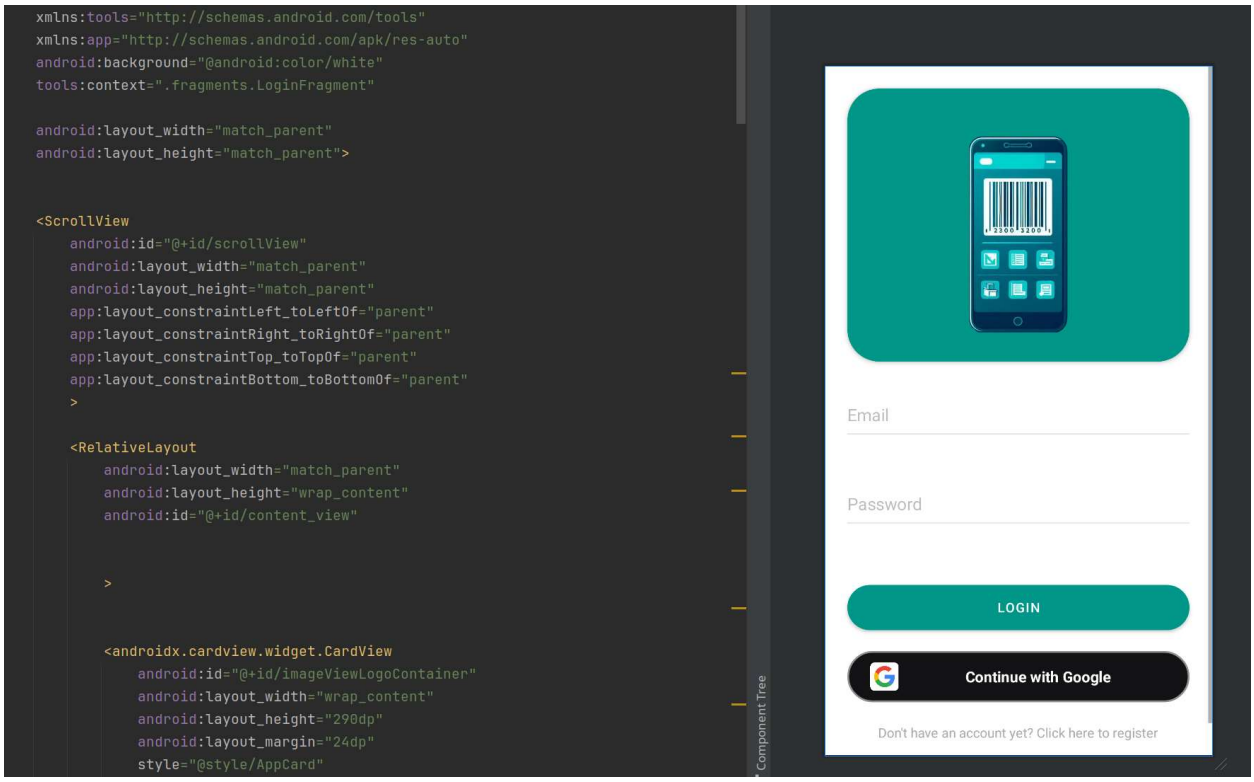


Рисунок 5.3 – створення інтерфейсу LoginActivity

Автентифікація через Firebase:

– Налаштовано взаємодію з Firebase Authentication для перевірки введених даних та обробки авторизаційних запитів.

Логіка обробки помилок:

– Додано перевірку на правильність введених даних і обробку помилок під час автентифікації демонстрацію можно побачити на рисунку 5.4.

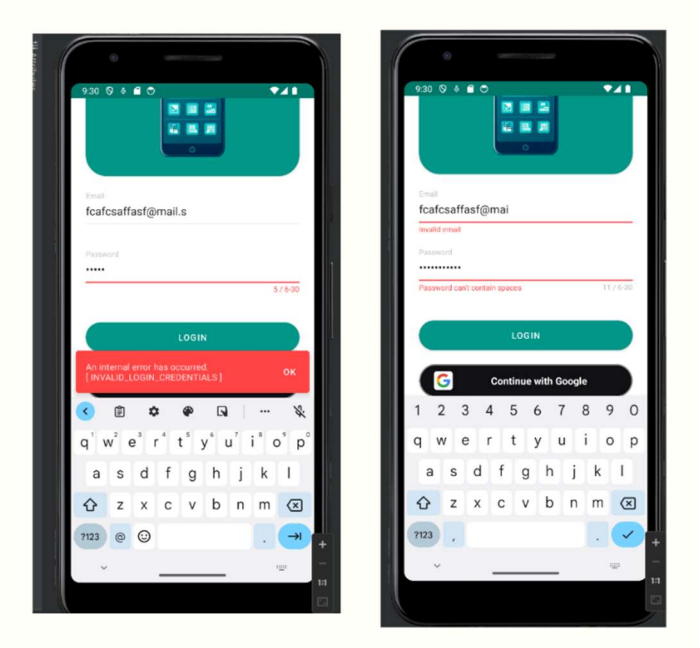


Рисунок 5.4 – обробка помилок автентифікації

2. RegisterActivity

Створення інтерфейсу для реєстрації:

Розроблено активність для реєстрації нового користувача з полями для введення електронної пошти та пароля, а також кнопками для підтвер-дження реєстрації. Процес створення відображено на рисунку 5.5.

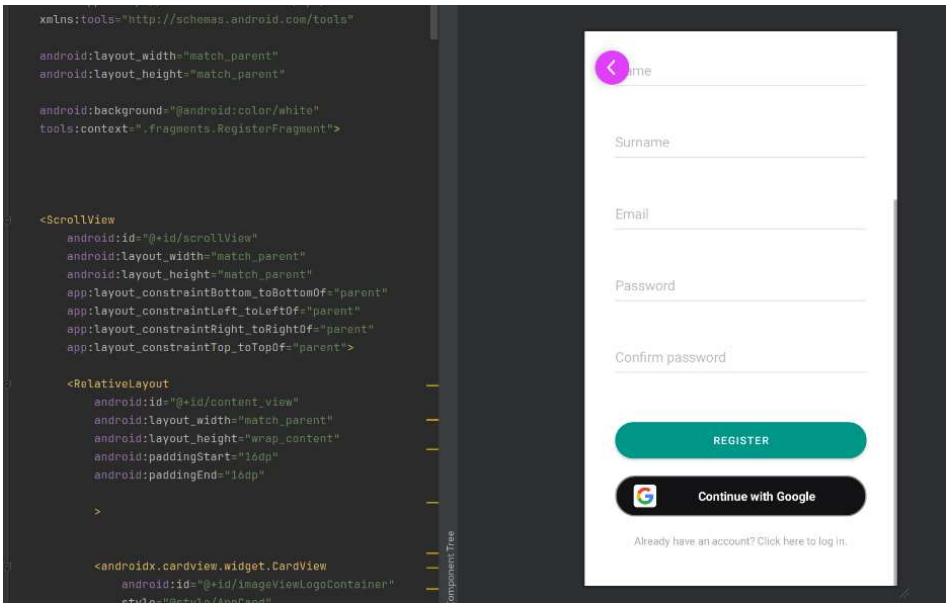


Рисунок 5.5 – створення інтерфейсу RegisterActivity

Реєстрація через Firebase Authentication:

– Інтегровано з Firebase для створення нового облікового запису користувача.

Обробка винятків:

– Додано механізм обробки винятків при помилках під час реєстрації користувача.

3. MainActivity

Головна активність додатку (рисунок 5.6).

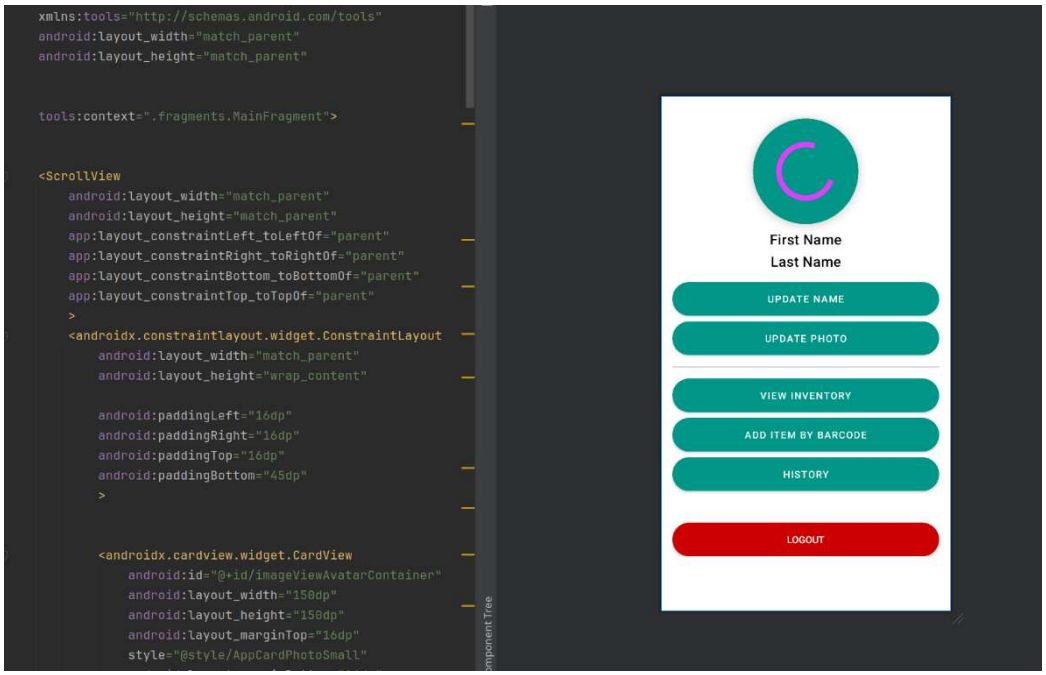


Рисунок 5.6 – створення інтерфейсу MainActivity

– Створено основну активність, яка відображається після успішної автентифікації користувача.

– Додано центральне меню з кнопками для переходу до різних функціональних модулів додатку.

4. InventoryActivity

Управління інвентарем та продуктами (рисунок 5.7):

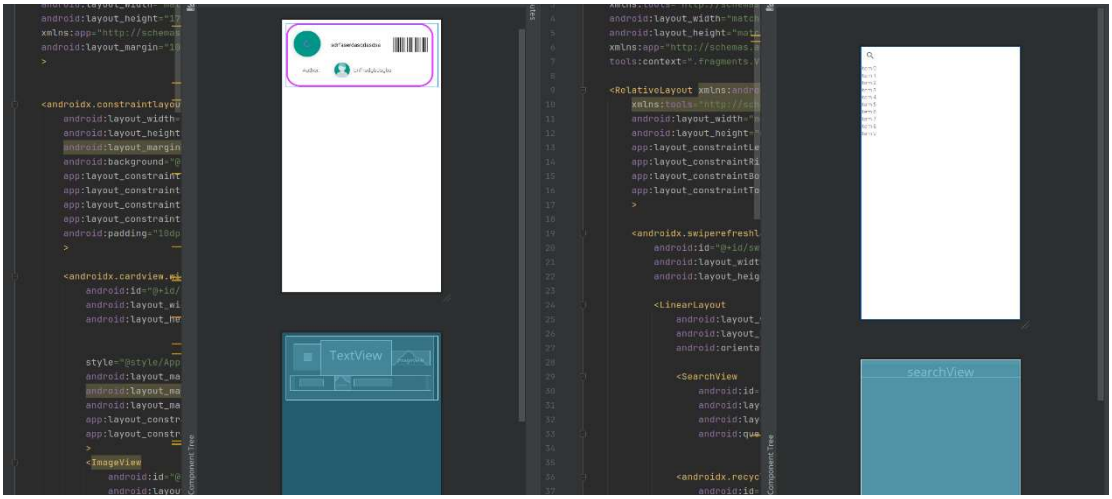


Рисунок 5.7 – створення інтерфейсу InventoryActivity

– Створено активність для управління списком продуктів та інвентарем користувача.

Інтеграція з Firebase Firestore:

– Реалізовано додавання, редагування та видалення продуктів з використанням бази даних Firestore.

5. ScanBarcodeActivity

Сканування штрих-кодів:

– Розроблено активність для сканування штрих-кодів за допомогою камери пристрою.

– Інтеграція з бібліотекою ZXing:

– Використано бібліотеку ZXing для реалізації функціоналу сканування штрих-кодів.

```
val integrator = IntentIntegrator(ActivityHolder.instance.getAc-
tivity()!!)
        integrator.setDesiredBarcodeFormats(IntentIntegra-
tor.ONE_D_CODE_TYPES + IntentIntegrator.PRODUCT_CODE_TYPES)
        integrator.setPrompt("Scan a barcode")
        integrator.setCameraId(0) // 0 for back camera, 1 for
front camera
        integrator.setBeepEnabled(true)
        integrator.setOrientationLocked(true)
```

```
integrator.setBarcodeImageEnabled(true)
integrator.initiateScan().
```

6. ProductDetailsActivity

Відображення деталей продукту (рисунок 5.8):

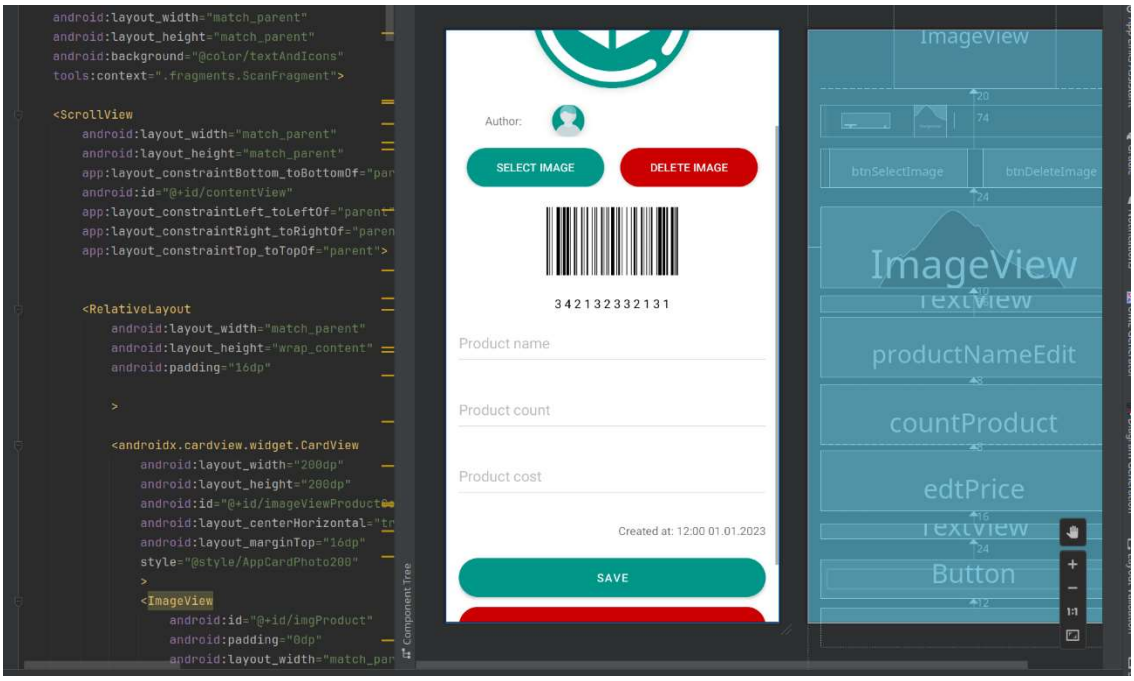


Рисунок 5.8 – створення інтерфейсу ProductDetailsActivity

– Створено активність для відображення детальної інформації про обраний продукт.

Показ даних з Firebase Firestore:

– Інтегровано з Firestore для завантаження та відображення інформації про продукт.

Реалізація активностей у додатку "InventoryTrackerApp" відповідає специфікації і вимогам до зручності використання та надійності. Кожна активність має чітко визначену функціональність, інтегровану з Firebase для забезпечення потрібних функцій, таких як автентифікація, управління інвентарем та сканування штрих-кодів.

5.3 Реалізація бізнес-логіки додатку

1. Реалізація бізнес-логіки

Управління користувачами та автентифікація:

– Налаштовано систему автентифікації через Firebase Authentication для входу та реєстрації користувачів.

Управління інвентарем:

– Реалізовано функціонал для додавання, редагування та видалення продуктів в базі даних Firestore.

– Розроблено інтерфейс для відображення списків продуктів та їх характеристик.

Інтеграція сканера штрих-кодів:

– Впроваджено сканер штрих-кодів для ідентифікації та обробки продуктів з використанням камери пристрою.

Робота зі зображеннями продуктів:

– Реалізовано механізм отримання та відображення зображень продуктів з використанням бібліотеки Picasso.

2. Використання бібліотеки Picasso

Інтеграція Picasso в додаток:

– Додано залежність на Picasso у файл build.gradle проекту [11].

```
dependencies {
    implementation("com.squareup.picasso:picasso:2.8")
    implementation("com.squareup.okhttp3:okhttp:4.9.3")}
```

Отримання зображень з FreeImage Host Hosting (приклад використання самописного кліенту):

```
startLoading()
                    val client = FreeImageHostClient()
                    client.uploadImage(ActivityHolder.in-
stance.getActivity()!!, photoURI!!, object : FreeImageHostCli-
ent.UploadCallback {
                        override fun onSuccess(imageUrl:
String) {
                            var uInfo = AuthManager.in-
stance.currentUserInfo!!
```

```
                                var newUInfo = User(uInfo.uuid,
uInfo.firstName, uInfo.lastName, imageUrl, uInfo.registerAt)//
                                UserManager.instance.updateUse-
rAsync(newUInfo) {
                                    mainHandler.post {
                                        stopLoading()
                                        if (it.isOk) {
                                            AuthManager.in-
stance.currentUserInfo = newUInfo
                                            findNavControl-
ler().popBackStack()
                                        } else {
                                            ActivityHolder.in-
stance.showError(it.errorMessage ?: "Unknown error")
                                        }
                                    }
                                }
                            }
                            override fun onFailure(error: String) {
                                mainHandler.post {
                                    stopLoading()
                                    ActivityHolder.instance.showEr-
ror(error)
                                }
                            }
                        })
```

Зареєстровано та отримано ключ API для FreeImage Host Hosting, яке використовується як сховище зображень продуктів [12].

При створенні або редагуванні продукту, URL зображення зберігається в базі даних Firestore разом з іншими даними про продукт.

Завантаження та відображення зображень за допомогою Picasso:

При завантаженні даних про продукт з Firestore, URL зображення використовується для завантаження зображення за допомогою Picasso.

```
    with(Picasso.get()) {
                        load(Uri.parse(imageURL))
                            .into(binding.imgProduct, object :
Callback {
                                override fun onSuccess() {
                                    binding.imgProduct.visibility =
View.VISIBLE
                                    productImageStopLoading()

                                }
                                override fun onError(e: Exception?)
{
                                    if (e != null) {

ActivityHolder.instance.showError(e!!.message.toString())
                                    } else {

ActivityHolder.instance.showError("Unknown error")
```

```
                    }
                    productImageStopLoading()
                }
            })
        }
```

Кешування та оптимізація завантаження зображень:

– Picasso автоматично кешує завантажені зображення, що зменшує час завантаження при повторних запитах та оптимізує використання мережевих ресурсів.

Реалізація бізнес-логіки у додатку "InventoryTrackerApp" інтегрує сучасні практики розробки для забезпечення ефективного управління інвентарем та взаємодії з користувачами. Використання бібліотеки Picasso спрощує процес завантаження та відображення зображень продуктів, забезпечуючи швидкий доступ до важливих даних для користувачів додатку.

5.4 Налагодження та оптимізація додатку

Розробка та оптимізація додатку "InventoryTrackerApp" виявилася важкою, але захоплюючою задачею для мене, як новача в області розробки мобільних додатків. Цей звіт охоплює мої найважливіші кроки у впровадженні функціональності, використанні бібліотеки Picasso для роботи з зображеннями та процесі налагодження та оптимізації додатку.

Для ефективної відладки додатку я використовував логування. Введення рівня логування на рівні бізнес-логіки та інтерфейсу дозволило мені відстежувати дії програми та виявляти потенційні проблеми.

Android Profiler використовувався для моніторингу використання ресурсів, таких як CPU, пам'ять та мережа. Це дозволило мені виявляти проблеми з продуктивністю додатку та шукати шляхи їх вирішення.

Під час налагодження я перевіряв роботу додатку на різних моделях пристроїв та версіях Android, щоб переконатися в його стабільності та коректній роботі в різних умовах.

Для оптимізації швидкодії завантаження зображень я використовував кешування зображень на стороні клієнта та налаштував політики кешування та мережевих запитів у бібліотеці Picasso.

Для зменшення кількості даних, що передаються, та підвищення швидкості реакції інтерфейсу я використовував фільтрацію даних та обмеження на кількість запитів до бази даних Firebase.

Моя робота також включала оптимізацію інтерфейсу користувача для забезпечення плавного та ефективного взаємодії з додатком, мінімізацію кількості запитів до сервера та підвищення загальної продуктивності.

Процес налагодження та оптимізації додатку "InventoryTrackerApp" був важкою, але надзвичайно цікавою досвідом для мене, як новача в мобільній розробці. Ці кроки дозволили підвищити якість додатку, зробити його більш ефективним та зручним для користувачів.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра був створений мобільний застосунок "InventoryTrackerApp" для управління інвентарем з використанням технології сканування штрих-кодів. Для цього був проведений аналіз предметної області та дослідження стеку доступних мені технологій та засобів. В якості програмного засобу розробки було обрано середовище Android Studio з огляду на те, що воно є стандартом для розробки додатків під Android.

Інструментом для роботи з базою даних було обрано Firebase Firestore, оскільки він забезпечує зручне зберігання та синхронізацію даних в реальному часі. Для автентифікації користувачів використовувалася Firebase Authentication, що підтримує різні методи входу, включаючи електронну пошту та Google.

На етапі проектування були створені UML-діаграми варіантів використання (прецедентів), активності та послідовності, а також розроблені макети інтерфейсу застосунку. З огляду на необхідність зберігання зображень продуктів та їх подальшого відображення у додатку, було вирішено використовувати хмарне сховище FreeImage Host та бібліотеку Picasso для завантаження зображень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Official Android Documentation. URL: https://developer.android.com/develop (дата звернення 5.05.2024)
2. Android Development with Kotlin by Google. URL: https://developer.android.com/kotlin (дата звернення 7.05.2024)
3. AndroidX overview. URL: https://developer.android.com/jetpack/androidx (дата звернення 7.05.2024)
4. Modern Android development. URL: https://developer.android.com/modern-android-development (дата звернення 9.05.2024)
5. Barcodes Defined—How They Work, Benefits & Uses. URL: https://www.netsuite.com/portal/resource/articles/inventory-management/barcodes.html (дата звернення 10.05.2024)
6. ZXing ("Zebra Crossing") GitHub Repository. URL: https://github.com/zxing/zxing (дата звернення 10.05.2024)
7. Офіційний сайт з документацією Firebase Firestore. URL: https://firebase.google.com/docs/reference/android/com/google/firebase/firestore/FirebaseFirestore (дата звернення 12.05.2024)
8. Firebase Authentication. URL: https://firebase.google.com/products/auth#:~:text=Firebase%20Authentication%20aims%20to%20make,and%20GitHub%20login%2C%20and%20more (дата звернення 13.05.2024)
9. Unified Modeling Language (UML) Diagrams. URL: https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/ (дата звернення 14.05.2024)
10. PlantUML Documentation. URL: https://plantuml.com/en/ (дата звернення 14.05.2024)
11. Picasso GitHub Repository. URL: https://github.com/square/picasso (дата звернення 18.05.2024)

12. Офіційний сайт Freeimagehost. URL: https://freeimage.host/uk (дата звернення 24.05.2024)