

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Прогнозування поведінки користувачів на базі
застосунку для ОС Android»**

Виконав студент групи ІПЗм-21-1
ХІМІЧ Владислав Олегович

Керівник роботи:
ЧИЖМОТРЯ Олексій Володимирович

Рецензент:
ЛОКТИКОВА Тамара Миколаївна

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії програмного
забезпечення

Тетяна ВАКАЛЮК

«18» вересня 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ХІМІЧ Владислав Олегович**

Керівник роботи: **ЧИЖМОТРЯ Олексій Володимирович**

Тема роботи: **«Прогнозування поведінки користувачів на базі застосунку для ОС Android»**,

затверджена Наказом закладу вищої освіти від **«18» вересня 2023 р. № 479/с**

Вихідні дані для роботи: Об'єктом дослідження є методи прогнозування поведінки користувача при користуванні програмними продуктами. Предметом дослідження є застосування методів прогнозування поведінки користувача на базі застосунку для ОС Android.

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.В.	22.09.2023р.	22.09.2023р.
2	Чижмотря О.В.	10.10.2023р.	10.10.2023р.
3	Чижмотря О.В..	25.10.2023р.	25.10.2023р.
4	Чижмотря О.В.	17.11.2023р.	17.11.2023р.

РЕФЕРАТ

Індивідуальне завдання кваліфікаційної магістерської роботи полягає в дослідженні методів, способів та технологій прогнозування поведінки користувачів у застосунку для ОС Android, а саме використання отриманих аналітичних даних користувачів для подальшого покращення наявного програмного продукту.

Кваліфікаційна магістерська робота на тему «Прогнозування поведінки користувачів на базі застосунку для ОС Android» складається з теоретичного та програмного комплексу системи та пояснювальної записки. Пояснювальна записка до кваліфікаційної магістерської роботи складається зі вступу, чотирьох розділів, списку літератури з 30 найменувань, 29 рисунків, 2 таблиць. Повний обсяг роботи становить 56 сторінок.

КЛЮЧОВІ СЛОВА: ЗАСТОСУНОК, ОС ANDROID, АНАЛІТИКА, ПРОГНОЗУВАННЯ, АНАЛІЗ ДАНИХ, ІНТЕРПРЕТАЦІЯ.

ABSTRACT

The individual task of the qualifying master's work consists in researching the methods, methods and technologies of predicting user behavior in the application for the Android OS, namely the use of the received analytical data of users for further improvement of the existing software product.

The qualifying master's thesis on the topic "Prediction of user behavior based on the application for the Android OS" consists of a theoretical and software complex of the system and an explanatory note. The explanatory note to the qualifying master's thesis consists of an introduction, four chapters, a bibliography of 30 titles, 29 figures, 2 tables. The full volume of the work is 56 pages.

KEY WORDS: APPLICATION, ANDROID OS, ANALYTICS, FORECASTING, DATA ANALYSIS, INTERPRETATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ ПРОЦЕСУ ЗБОРУ ДАНИХ КОРИСТУВАЧІВ У ЗАСТОСУНКАХ ОС ANDROID	10
1.1. ВАЖЛИВІСТЬ ЗБОРУ ДАНИХ ДЛЯ ПРОГНОЗУВАННЯ ПОВЕДІНКИ КОРИСТУВАЧІВ.....	10
1.2. АНАЛІЗ ЗАСОБІВ ЗБОРУ ДАНИХ ДЛЯ ЗАСТОСУНКІВ ОС ANDROID.....	11
1.3. АНАЛІЗ СПОСОБІВ ПРОГНОЗУВАННЯ ПОВЕДІНКИ КОРИСТУВАЧІВ	13
Висновки до розділу 1.....	15
РОЗДІЛ 2. ПРОЄКТУВАННЯ ЗАСТОСУНКУ ДЛЯ ЛІЧИЛЬНИКІВ	16
2.1. АНАЛІЗ АНАЛОГІВ ПРОГРАМНОГО ПРОДУКТУ.....	16
2.2. ВИБІР АРХІТЕКТУРИ ЗАСТОСУНКУ	20
2.3. ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ ЗАСТОСУНКУ ДЛЯ ОС ANDROID.....	22
2.4. РОЗРОБКА БАЗИ ДАНИХ ЗАСТОСУНКУ	24
2.5. РОЗРОБКА ВНУТРІШНЬОЇ АНАЛІТИКИ ЗАСТОСУНКУ	28
Висновки до розділу 2.....	31
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ЗАСТОСУНКОМ ДЛЯ ЛІЧИЛЬНИКІВ	32
3.1. ПОРЯДОК ВСТАНОВЛЕННЯ ЗАСТОСУНКУ	32
3.2. СТРУКТУРА ІНТЕРФЕЙСУ. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ЗАСТОСУНКОМ ДЛЯ ЛІЧИЛЬНИКІВ.	33
3.3. ТЕСТУВАННЯ РОБОТИ СИСТЕМИ	43
Висновки до розділу 3.....	45
РОЗДІЛ 4. АНАЛІЗ ТА ІНТЕРПРЕТАЦІЯ ОТРИМАНИХ АНАЛІТИЧНИХ ДАНИХ	46
4.1. АНАЛІЗ АНАЛІТИЧНИХ ДАНИХ ЗГЕНЕРОВАНИХ КОРИСТУВАЧАМИ.....	46

4.2. СПОСОБИ ПОКРАЩЕННЯ ЗАСТОСУНКУ НА ОСНОВІ ОТРИМАНИХ АНАЛІТИЧНИХ ДАНИХ.....	50
Висновки до розділу 4.....	51
ВИСНОВКИ	52

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ОС – Операційна система.

SDK – Software Development Kit – набір інструментів для розробки програмного забезпечення

API – Application Programming Interface – прикладний програмний інтерфейс

ВСТУП

Актуальність. Смартфон у його сучасному розумінні було винайдено не так давно, близько 15-ти років тому, але за цей короткий відтинок часу він став невід'ємною частиною життя практично будь-якої людини. Нині саме смартфони є гаджетами, котрими середньостатистична особа користується найчастіше в особистих потребах, що обумовлює бурхливий розвиток застосунків для мобільних ОС. Однак для успішного розвитку додатків необхідно розуміти, як користувач взаємодіє із застосунком, які він має патерни поведінки та які дії є найбільш важливими для його досвіду використання. Саме на дослідженні цього питання зосереджено увагу в цій роботі. Основою для дослідження є власноруч створений застосунок для лічильників для ОС Android, що викликано, як і популярністю цієї ОС у всьому світі, так і важливістю інтеграції аналітики в мобільні застосунки.

Метою дослідження є аналіз методів прогнозування поведінки користувача на базі застосунку для ОС Android. Встановлена мета обумовлює наступні завдання:

- аналіз засобів та способів прогнозування поведінки користувачів;
- аналіз аналогічних програмних продуктів;
- створення макету дизайну застосунку;
- визначення архітектури та узагальненої структури застосунку;
- обґрунтування та вибір засобів реалізації застосунку;
- проєктування структурних складових застосунку;
- реалізація програмного комплексу застосунку;
- публікація застосунку в Google Play;
- збір та аналіз аналітичних даних користувачів програмного продукту;
- інтерпретація та подальше формування варіантів покращення користувацького досвіду та популяризації застосунку

Об'єктом дослідження є методи прогнозування поведінки користувача при користуванні програмними продуктами.

Предметом дослідження є застосування методів прогнозування поведінки користувача на базі застосування для ОС Android.

Практичне значення цієї роботи полягає у тому, що розроблений застосунок та доданий до нього аналітичний модуль надають дані про поведінку користувачів, що дозволяє знайти варіанти покращення користувацького досвіду при подальшій розробці та віднайти способи популяризації програмного продукту.

Матеріали кваліфікаційної роботи були включені до матеріалів VI Всеукраїнської науково-технічної конференції: "Комп'ютерні технології: інновації, проблеми, рішення" 29-30 листопада 2023 р. [1].

РОЗДІЛ 1. АНАЛІЗ ПРОЦЕСУ ЗБОРУ ДАНИХ КОРИСТУВАЧІВ У ЗАСТОСУНКАХ ОС ANDROID

1.1. Важливість збору даних для прогнозування поведінки користувачів

Процес збору даних про поведінку користувачів в застосунку є надзвичайно важливим для розуміння та вдосконалення продукту. Отримання відомостей про те, як користувачі взаємодіють з застосунком, їхні дії, вподобання та звички, дозволяє зрозуміти, як поліпшити користувацький досвід, підвищити задоволення клієнтів та залучити нових користувачів.

Ось деякі ключові аспекти, що підкреслюють важливість процесу збору даних про поведінку користувачів в застосунку:

Розуміння потреб користувачів: Збір даних про поведінку дозволяє зрозуміти, як користувачі використовують застосунок та які їхні потреби. Це надає можливість адаптувати функціонал, вдосконалювати інтерфейс та забезпечувати більш зручний та задоволений досвід для користувачів.

Вдосконалення продукту: Дані про поведінку користувачів дають розробникам можливість зрозуміти, як їхній продукт використовується в реальних умовах. Вони можуть виявити слабкі місця, помилки та недоліки, що допомагає вдосконалити функціонал, оптимізувати продукт та забезпечити його більшу ефективність.

Персоналізація та адаптація: Збір даних про поведінку дозволяє створювати персоналізований досвід для користувачів. Інформація про їхні вподобання та звички може бути використана для надання рекомендацій, пропозицій та індивідуальних налаштувань, що покращують користувацький досвід та збільшують задоволення від використання застосунку.

Прийняття ділових рішень: Дані про поведінку користувачів надають цінну інформацію для прийняття ділових рішень. Вони допомагають визначити ефективність рекламних кампаній, оцінити вплив нових функцій та змін на

користувачів, а також розробити стратегії для покращення конверсії та збільшення доходів.

Виявлення проблем та уразливостей: Збір даних про поведінку користувачів також допомагає виявляти проблеми безпеки, баги та уразливості в застосунку. При ретельному аналізі даних можна виявити аномалії, незвичайні активності та спроби злому, що дозволяє приймати заходи для забезпечення безпеки та надійності продукту.

Загалом, процес збору даних про поведінку користувачів в застосунку є незамінним інструментом для розуміння та вдосконалення продукту. Використання цих даних допомагає компаніям створювати більш привабливі та корисні застосунки, задовольняти потреби своїх користувачів та досягати успіху на ринку.

1.2. Аналіз засобів збору даних для застосунків ОС Android

Компанія Google, яка є основним розробником ОС Android надає власні технічні засоби збору даних з застосунків, які легко інтегруються при розробці додатків та є доволі гнучкими для зміни під власні потреби.

Основні технічні засоби для збору даних про поведінку користувачів в операційній системі Android включають:

Google Analytics for Mobile Apps: Це потужний безкоштовний сервіс, розроблений Google, який надає розширені засоби аналізу для мобільних додатків Android. За допомогою цього інструменту розробники можуть отримати відомості про кількість встановлень, активність користувачів та інші метрики. Однак на цей момент він вже є застарілим та не удосконалюється компанією Google.

Firebase Analytics: Цей інструмент, який є частиною платформи розробки мобільних додатків Firebase, надає комплексну аналітичну систему для додатків Android. Він дозволяє збирати дані про користувачів, взаємодію з додатком,

конверсії та інші метрики, що допомагає розробникам краще розуміти поведінку своїх користувачів і вдосконалювати додаток.

Crashlytics: Цей сервіс, також входить до складу Firebase, надає засоби виявлення і відстеження збоїв і помилок в додатках Android. Він дозволяє отримувати докладну інформацію про причини збоїв, стек викликів, дані пристрою та інші важливі параметри. Завдяки Crashlytics розробники можуть швидко виправляти проблеми і забезпечувати більш стабільну роботу свого додатку.

User Interaction Tracking: Розробники можуть використовувати різні техніки для відстеження взаємодії користувачів з додатком, такі як відслідковування подій та жестів. Це дозволяє збирати дані про кліки, прокрутку, наведення курсору та інші взаємодії користувачів, що стає основою для аналізу та вдосконалення інтерфейсу.

Ці технічні засоби дозволяють розробникам отримати вичерпну інформацію про користувачів, їхню взаємодію з додатком, виявляти проблеми та покращувати продукт.

Для ефективного збору великої кількості інформації не варто зосереджуватися лише на одному засобі збору даних, а краще використати найкращі сторони кожного з них, щоб зібрати різноманітні дані при цьому зручно розділивши їх за типами. Саме тому для збору інформації про небажану поведінку застосунку, таку як вильоти чи ANR (Application Not Responding) буде використовуватися Crashlytics, який є найзручнішим інструментом для трекінгу таких проблем для застосунків на ОС Android. В той же час для власних подій, що будуть викликатися при певних діях користувача, буде використовуватися Firebase Analytics, однією з переваг якого є те, що він легко інтегрується та підлаштовується під інші сервіси Firebase, такі як – Remote Config, Cloud Functions та інші.

Також при важливості отримання більше даних саме про взаємодію з інтерфейсом буде використано User Interaction Tracking, однак на перших етапах розробки застосунку така імплементація не передбачається.

В підсумку для збору даних в застосунку для ОС Android будуть використовуватися Firebase Analytics для власних подій та Crashlytics для збору інформації про вильоти та інші технічні проблеми застосунку.

1.3. Аналіз способів прогнозування поведінки користувачів

Прогнозування поведінки користувачів за допомогою великої кількості аналітичних даних, зібраних в застосунку під ОС Android, є складним завданням, але з використанням різних технік та методів, є можливим. Ось кілька доступних способів прогнозування:

Машинне навчання: Використання методів машинного навчання дозволяє побудувати прогнозні моделі, які засновані на аналізі великої кількості даних користувачів. Ці моделі можуть передбачати дії користувачів, їхні вподобання, звички, реакції на певні події та інше. Використовуючи алгоритми класифікації, кластеризації, регресії та інші, можна побудувати прогнозні моделі, які надають цінні інсайти.

Рекомендаційні системи: Це технології, що базуються на аналізі даних про поведінку користувачів для надання персоналізованих рекомендацій. Застосовуючи методи колаборативного та контент-базованого фільтрування, рекомендаційні системи можуть передбачати, які товари, послуги, контент або функції будуть цікаві конкретному користувачу.

Когнітивний аналіз: Це підхід до аналізу даних, який заснований на розумінні людських психологічних процесів. Застосовуючи когнітивний аналіз, можна отримати інсайти про мотивації користувачів, їхні вподобання, ставлення до продукту та інші фактори, що впливають на їх поведінку.

Аналіз використання додатків: Зібрані дані про взаємодію користувачів з додатками дозволяють виявити патерни використання, аномальні дії та слабкі місця. Це допомагає розробникам вдосконалити інтерфейс, функціонал та процеси, забезпечуючи зручний та задоволений досвід для користувачів.

Сегментація користувачів: Аналіз даних дозволяє виокремити різні групи користувачів за певними характеристиками. Сегментація допомагає краще розуміти потреби та вподобання різних категорій користувачів та розробляти спеціалізовані стратегії для залучення і утримання кожної групи.

Аналіз відмов: Виявлення та аналіз поведінки користувачів, які припинили використання додатка (так звані "відмови"), допомагає з'ясувати причини невдоволення та недоліків продукту. Це дозволяє покращити додаток, враховуючи причини відмов та підвищити задоволення клієнтів.

Для того, щоб визначити найбільш підходящі способи для аналізу аналітичних даних, зібраних в застосунку, необхідно взяти до уваги засоби збору даних, які були обрані для цього проєкту. Так, найкращими рішеннями для аналізу будуть: аналіз використання додатків, сегментація користувачів та аналіз відмов, адже машинне навчання є занадто складним та комплексним підходом для застосунку невеликих розмірів. Також рекомендаційні системи не є дуже доречними для саме цього проєкту для ОС Android, адже він не має функціоналу рекомендації користувачу певного контенту.

В той же час загальний аналіз використання додатків є чи не найважливішим для успішного підтримання та розвитку застосунків, адже дозволяє комплексно оцінити загальну картину якості застосунку, патерни використання та слабкі місця, що надає хороше підґрунтя для покращення користувацького досвіду. В самому проєкті ці аналітичні дані представлені у формі окремих подій з власноруч визначеними та створеними параметрами. Для такої мети Firebase Analytics є ідеальним інструментом, адже надає зручний інтерфейс, багатий функціонал та, мабуть, найкращу інтеграцію з застосунками для ОС Android.

Однак не завжди користувача можна затримати надовго у застосунку з різних причин, тому для кращого відсотку утримання в додатку потрібно застосувати спосіб аналізу відмов. Збір інформації про цей процес відрізняється на кожній з платформ і ОС Android не є виключенням, адже найпростішим та досить ефективним способом аналізу відмов є аналіз відгуків на сторінці

застосунку в Google Play. Саме ці відгуки надають корисний фідбек на пряму від людей, що користуються застосунком. Далі проблеми, що описують користувачі, можна категоризувати та намагатися віднайти найбільш ефективні рішення.

Отже, проаналізувавши можливі способи прогнозування поведінки користувачів, було обрано найбільш підходящі для цього застосунку під ОС Android, а саме – аналіз використання додатків та аналіз відмов.

Висновки до розділу 1

У цьому розділі дипломної роботи було обґрунтовано важливість процесу збору даних для прогнозування поведінки користувачів та подальшого покращення програмного продукту для ОС Android.

Далі було проаналізовано засоби збору даних для застосунків для ОС Android та обрано ті, що найкраще підходять для розроблюваного програмного продукту, а саме:

- Firebase Analytics для збору аналітичних даних
- Crashlytics для збору інформації про технічні проблеми застосунку

В останньому пункті першого розділу було проаналізовано способи прогнозування поведінки користувачів та виокремлено ті, що використовуватимуться у цьому застосунку: аналіз використання додатків та аналіз відмов.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ЗАСТОСУНКУ ДЛЯ ЛІЧИЛЬНИКІВ

2.1. Аналіз аналогів програмного продукту

Тепер буде доречним розглянути аналоги програмного продукту, котрі вже доступні в Google Play. Такий тип застосунків є доволі поширеним на операційній системі Android, тому знайти аналоги буде просто.

Першим застосунком для аналізу буде найпопулярніший з них, що має близько мільйона завантажень, - Counter.

При вході в застосунок для нас відразу відкривається екран з пустим лічильником, хоча ми не обирали цю дію, і це може бути як плюсом, так і мінусом, бо з одного ми відразу можемо почати рахувати, а з іншого, можливо, нам це непотрібно з першого моменту заходу в застосунок. Далі ми можемо побачити, що дизайн застосунку є непоганим, але при цьому виглядає трохи застаріло та моментами не інтуїтивно, як, наприклад, у випадку зі списком лічильників. Сам функціонал лічильника є непоганим в плані варіативності, адже корегування значення лічильника можна здійснювати натисканням кнопок, збільшення на одиницю при натисканні на поточне значення і навіть клавішами гучності самого смартфона. Але і в цьому функціоналі є один мінус: не можна змінювати крок лічильника. Також варто відмітити, що в цьому застосунку немає можливості створити груповий лічильник та синхронізувати дані.

Отже, з першого і найпопулярнішого програмного продукту для аналізу можна зробити наступні висновки: потрібно приділити увагу зовнішньому вигляду власного застосунку та надати певну тактильну індикацію користувачу про зміну значення лічильника.

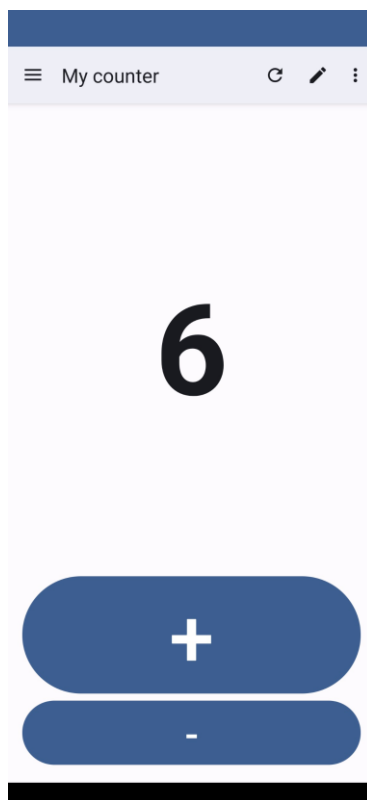


Рис. 2.1. Скріншот застосунку Counter

Наступним програмним продуктом для порівняння буде “Counter: Click Counter App”, який має меншу кількість завантажень, але також є дуже популярним. Цей застосунок має вже дійсно приємніший дизайн за попередній, варіативність відображення лічильників, плитками чи в стовпчик, та багатий функціонал з можливістю точного налаштування лічильника. Наприклад, можна налаштувати окремо значення інкременту та декременту, мінімальні та максимальні значення та інші параметри, що допомагають точно сконфігурувати необхідний лічильник, однак, як на мене, таке налаштування є надлишковим і занадто нагромаджує інтерфейс при редагуванні. Окремо хочеться виділити повноекранний режим, де на екрані відображається лише значення лічильника великим шрифтом та методи індикації збільшення значення лічильника такі, як: вібрація, звук та озвучування голосовим асистентом. Загалом цей застосунок значно кращий за попередній та має багатий і корисний функціонал, однак з мінусів можна виділити часткове нагромадження інтерфейсу та присутність реклами, яку можна відключити лише, придбавши платну підписку.

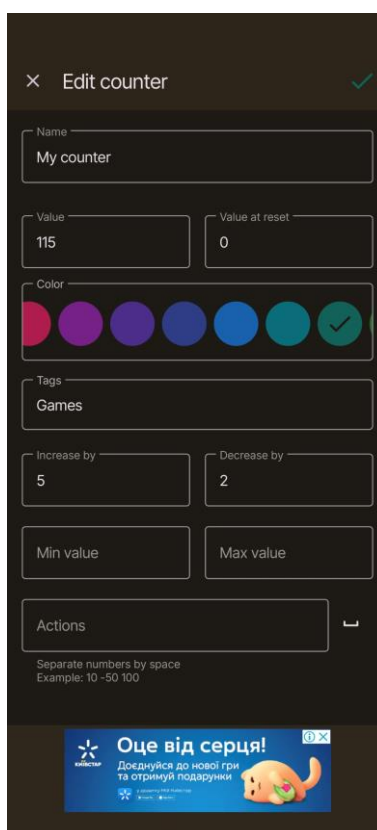


Рис. 2.2. Екран редагування лічильника в “Counter: Click Counter App”



Рис. 2.3. Повноекранний режим лічильника в “Counter: Click Counter App”

Останнім програмним продуктом для розгляду буде лічильник “Score Counter” від українського розробника. Якщо попередні застосунки були більш

універсальні, то саме цей більш орієнтований на допомогу в процесі гри в настільні ігри. Однак саме цей застосунок чудово поєднує привабливий інтерфейс та простий користувацький досвід. Також особливістю цього застосунку є можливість згенерувати число від 1 до 6 імітуючи підкидання грального кубика. Звісно, ця функція не буде доречною у власному програмному продукті, але цей момент реалізовано цікаво. Загалом можна сказати, що цей застосунок є найкращим з сьогоднішнього порівняння, адже добре поєднує у собі зручність, дизайн, функціональність та не нагромаджує користувача рекламою та платними підписками.

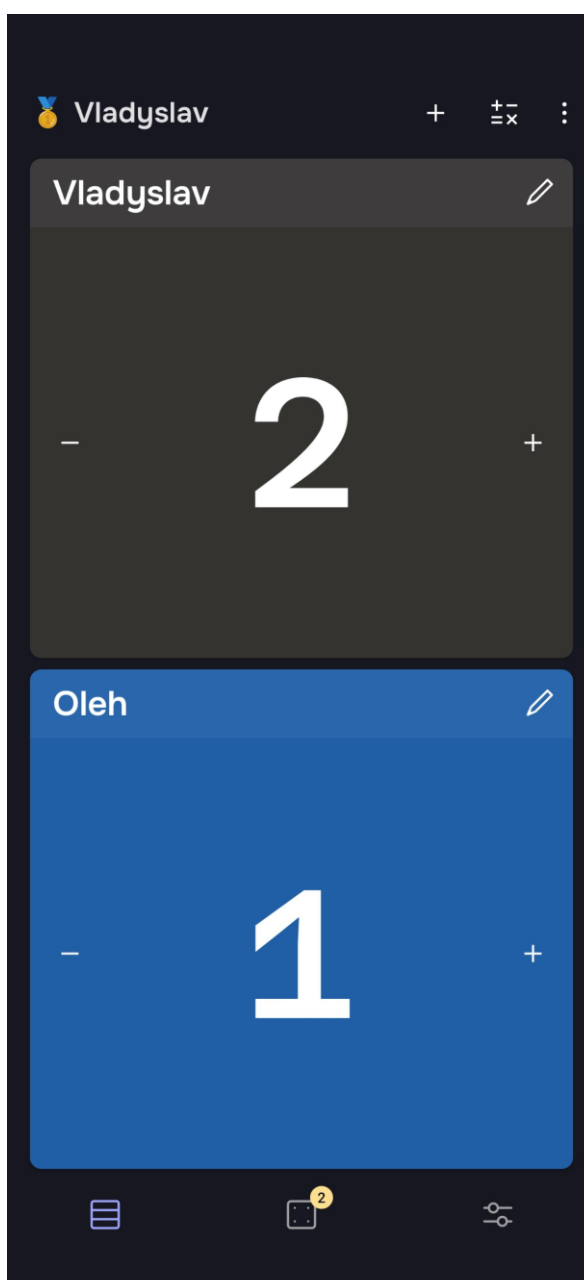


Рис. 2.4. Головний екран “Score Counter”

Таблиця 2.1.

Порівняння аналогів

Назва	Привабливий UI	Зручний та продуманий UX	Функціональність	Можливість кастомізації
Counter	Погано	Середнє	Середнє	Погано
Counter: Click	Середнє	Середнє	Добре	Добре
Score Counter	Добре	Добре	Добре	Середнє

Із проробленого вище аналізу можна зробити наступні висновки для розробки власного застосунку:

- потрібно знайти баланс між функціональністю та простотою користування для кінцевого користувача
- потрібно дотримуватися останніх рекомендацій Google для їхньої мови дизайну Material You, щоб зробити зрозумілий та привабливий інтерфейс
- відсутність реклами буде перевагою, а підписка повинна бути присутньою лише для додаткового функціоналу
- можливість створення групових лічильників та хмарна синхронізація може бути конкурентною перевагою

2.2. Вибір архітектури застосунку

Для розробки Android додатку можна використовувати наступні архітектурні патерни:

- MVC
- MVP
- MVI
- MVVM

Взагалі в одному проєкті можна поєднувати патерни в залежності від спеціальних вимог, але більш доцільним буде застосування одного патерну, щоб уникнути неконсистентності коду. За офіційними рекомендаціями щодо архітектури потрібно звернутися до документації Android, де власне Google описує найкращі підходи до проєктування застосунків під їх операційну систему.

Проаналізувавши документацію, можна зробити рішення, що MVVM (Model-View-ViewModel) буде вибором для загальної архітектури програмного комплексу цієї випускної роботи.

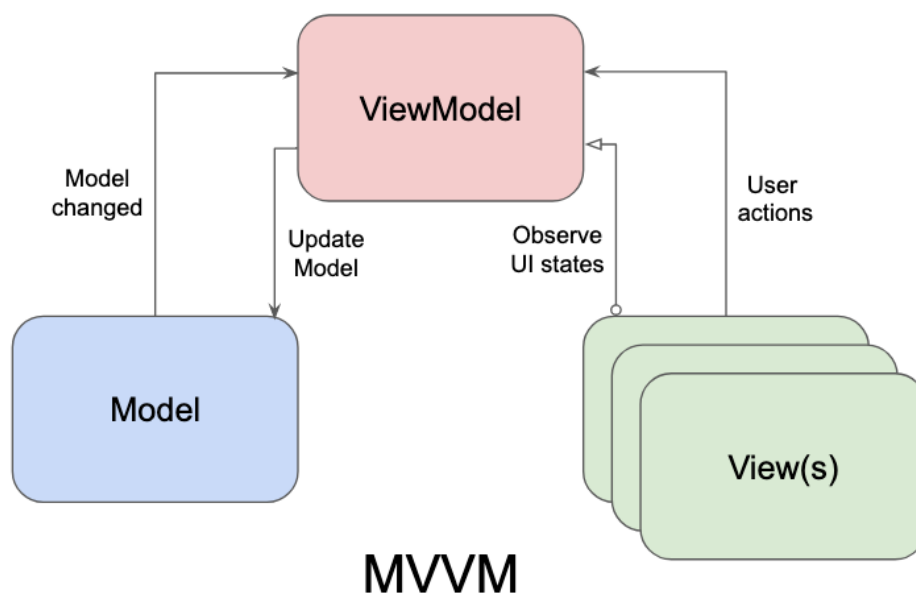


Рис. 2.5. Model-View-ViewModel

Model-View-ViewModel – це доволі поширений патерн проєктування не лише у розробці програмного забезпечення для ОС Android, але і для десктопних додатків з використанням інших технологій, таких як WPF (Windows Presentation Foundation) та інші. Тепер буде доречним розібрати кожен компонент цього архітектурного патерну.

Model – це рівень бізнес-логіки додатку, який не пов’язаний з жодним графічним представленням, і може включати в себе базу даних, репозиторій (Repository pattern) та інші класи керування бізнес-логікою.

View – це те, що бачить користувач, може мати як статичні, так і динамічні дані. За рекомендаціями Google цей шар не має містити логіку додатку, а лише те, що відповідає за відображення UI. Саме в ОС Android це має бути Fragment або Activity.

ViewModel – цей компонент зв’язує View та Model, проте в Android він має ще одну важливу роль: переживає зміни конфігурації UI, тобто, коли Activity

та/або Fragment перестворюються внаслідок дій користувача чи оптимізацій системи. При цьому всі необхідні дані зберігаються у ViewModel без потреби робити повторний запит до мережі чи БД, що покращує оптимізацію та загальну роботу додатку.

Загалом можна сказати, що саме патерн Model-View-ViewModel є стандартом де-факто для розробки програмного забезпечення для ОС Android, що доказує обширний розділ офіційної документації Android та безліч розробників, котрі надають перевагу саме йому. Тому вважаю, що це буде найкращим вибором для цієї роботи.

2.3. Вибір засобів реалізації застосунку для ОС Android

Тепер потрібно визначитися з інструментарієм для розробки застосунку. Перш за все, потрібно обрати мову програмування. Оскільки додатки Android працюють на JVM, серед мов програмування доступні лише два варіанти: Java або Kotlin. Ще з 2018-го року Google оголосив, що розробка під Android стає Kotlin-first і подальший розвиток інструментів для розробників опиратиметься саме на цю мову. З того часу Kotlin дуже розвинувся і зараз важко зустріти новий код, написаний на Java, тому і для цього проєкту буде використано саме Kotlin.

Для роботи з базою даних вибір теж невеликий, адже сам Google в офіційній документації рекомендує та описує роботу з ORM (Object-Relational Mapping) Room, яка нині є стандартом де-факто для роботи з єдиною доступною в Android СУБД – SQLite.

Щодо IDE, то цей вибір є легким, адже сам власник та розробник ОС Android – компанія Google – має своє офіційне середовище для розробки програмного забезпечення – Android Studio. Воно базоване на IntelliJ IDEA від компанії JetBrains – одного з лідерів розробки IDE загалом, отже, має постійну підтримку, швидкі оновлення, до того ж безкоштовне. Також до плюсів можна додати наявність в одному середовищі всіх необхідних інструментів для розробки таких, як: Device Emulator, Device Inspector, App Profiler та інші. Однак

є певний мінус, який варто брати до уваги, а саме вимогливість до характеристик пристрою, а якщо точніше, то до кількості оперативної пам'яті, адже лише одна Android Studio при важкому проєкті та ввімкненому емуляторі може використовувати близько 7 ГБ оперативної пам'яті, проте для моєї робочої машини це не є проблемою, адже на ній її об'єм складає 16 ГБ.

У ролі сторонньої програми для створення дизайну застосунку було обрано Figma, адже вона має найбагатший функціонал, доступна для перегляду з будь-якого пристрою та є легкою для освоєння.

З самого початку користувацькі інтерфейси для застосунку під цю операційну систему мали досить типовий спосіб визначення - опис елементів інтерфейсу через теги в файлах з розширенням .xml, котрі були окремими ресурсами для модулю застосунку. Далі ці файли розмітки мали поєднуватися з програмним кодом через Activity/Fragment в окремих Java-файлах, де писалася логіка для інтерфейсу та обробка подій. Однак це з'єднання двох різних за типом та застосуванням файлів змушувало писати багато повторюваного та надлишкового коду.

Однак у 2019 році на Google I/O було перевинайдено концепцію та інструменти створення інтерфейсу для Android-застосунків, що отримало назву Jetpack Compose. Це декларативний спосіб побудови інтерфейсу, побудований на ідеї модульної розробки інтерфейсів з використанням можливостей головної мови розробки для Android – Kotlin.

З його застосуванням всі елементи інтерфейсу описуються у звичайних функціях з анотацією Composable, що дозволяє ділити інтерфейс на групи елементів, розмір та суть яких обирає сам розробник. Таким чином при роботі з Jetpack Compose розробник по суті створює бібліотеку елементів інтерфейсу, необхідних саме його застосунку та які можна перевикористовувати в тій мірі, яка потрібна. Ще однією перевагою Jetpack Compose є ідея використання принципу Unidirectional Data Flow для оновлення стану UI в самій основі технології, що дозволяє значно зменшити кількість багів та проблем, пов'язаних з оновленням даних в інтерфейсі користувача. Значним покращенням в

порівнянні з .xml є те, що при зміні стану UI відбувається рендеринг лише тих Composable функцій, стан яких було змінено, і таким чином, якщо це зміна стану лише певного, навіть малого, елемента, то вона буде стосуватися лише його, зменшуючи навантаження на процесор мобільного пристрою. Разом зі стислістю коду, потрібного для опису інтерфейсу, та акцентом компанії Google на його використання та розвиток, цей інструмент поступово набирає популярність та великі компанії починають впроваджувати Jetpack Compose у свої продукти.

Для реалізації аналітики в застосунку буде використано Firebase Analytics з власними івентами, адже в застосунку вже буде використовуватися частина з Firebase, а саме Crashlytics буде застосовано для відслідковування проблем із застосунком. Firebase Analytics є простим в імплементації, зручним для аналізу даних та безкоштовним до певного об'єму запитів.

Отже, використано такі програмні продукти та середовища розробки:

Нативну розробку з використанням Android SDK, Kotlin, Compose; Firebase Analytics для аналітики; середовище розробки Android Studio; продукт для розробки макету Figma.

2.4. Розробка бази даних застосунку

Для створення схеми бази даних та проєктування таблиць буде використовуватись сервіс SQLDBM.



Рис. 2.6. Схема бази даних

Тепер варто розглянути та пояснити кожну з таблиць бази даних застосунку. Почнімо з таблиці Tag.

Таблиця Tag містить такі поля:

- Tag_Id – ідентифікаційний ключ таблиці
- Name – ім'я тегу
- Is_User_Created – булеве значення, що позначає те, чи тег створений користувачем
- Created_Date – дата створення тегу
- Modified_Date – дата модифікації тегу

Далі розглянемо головну таблицю для застосунку, а саме Counter. Таблиця Counter містить наступні поля:

- Counter_Id – ідентифікаційний ключ таблиці
- Name – ім'я лічильника
- Description – опис лічильника
- Starting_Value – початкове значення лічильника
- Step – крок лічильника
- Value – поточне значення лічильника
- Creation_Date – дата створення лічильника
- Modified_Date – дата останньої модифікації лічильника
- Is_Group – булеве значення, що позначає, чи є цей лічильник груповим
- Tag_Id – зовнішній ключ до таблиці Tag

Таблиця Group_Counter також є важливою для головної логіки лічильників та слугуватиме для частини функціоналу, пов'язаного з груповими лічильниками. Вона складається з наступних полів:

- Group_Counter_Id – ідентифікаційний ключ таблиці
- Name – ім'я групового лічильника
- Value – поточне значення групового лічильника
- Order – порядок групового лічильника
- Created_Date – дата створення групового лічильника
- Modified_Date – дата останньої модифікації групового лічильника
- Counter_Id – зовнішній ключ до таблиці Counter, не може бути пустим

Остання таблиця, котра використовується для збору певної статистики та аналітики для самого застосунку, без використання в Firebase Analytics, це Counter_Action і вона має наступні поля:

- Counter_Action_Id – ідентифікаційний ключ таблиці, для якого буде використовуватися випадково згенерована строка UUID
- Is_Plus – булеве значення, яке позначає додання чи віднімання значення лічильника
- Value – значення, до якого було змінено лічильник

- Action_Date – дата, коли було здійснено дію
- Counter_Id – зовнішній ключ до таблиці Counter, може бути відсутнім
- Group_Counter_Id – зовнішній ключ до таблиці Group_Counter, може бути відсутнім

Також варто визначитися з тим, якою буде структура бази даних в Firebase Firestore, що використовуватиметься для синхронізації даних користувачів, які купили підписку в застосунку. За своєю суттю Firestore це NoSQL база даних, де все зберігається в документах, які формуються у колекції, тому у цьому випадку база даних буде досить ієрархічна, де колекція найвищого рівня матиме назву Users, та зберігатиме дані про всі лічильники користувача у його окремому документі цієї колекції. Сам документ окремого лічильника міститиме всю зібрану інформацію про нього з таблиць локальної БД.

Також вважаю варто визначити логіку синхронізації та єдине джерело даних для застосунку. Як було зазначено раніше, для синхронізації для користувачів з підпискою буде використовуватися Firebase Firestore, який може працювати як в онлайн, так і офлайн режимі. Одним з варіантів є синхронізація в реальному часі з підміною джерела даних за певних умов, однак в цього рішення є певні обмеження, як, наприклад, ліміти по кількості безкоштовних запитів до Firebase Firestore, а такий підхід буде викликати велику кількість операцій з хмарним сервісом. Саме тому, як на мене, найкращим рішенням буде використати локальну базу даних як єдине джерело даних, яке у свою чергу буде наповнюватися даними з Firebase Firestore при необхідності. Щодо самої синхронізації, то буде доречним автоматично її виконувати при вході користувача в застосунок, якщо пройшло більше, ніж година з останньої синхронізації. Звісно ж, в налаштуваннях застосунку користувач зможе самостійно ініціювати синхронізації в будь-який зручний для нього момент.

2.5. Розробка внутрішньої аналітики застосунку

Для розробки аналітики спочатку потрібно визначити ключові події, які допоможуть прогнозувати поведінку користувачів та робити хороші оновлення застосунку, які будуть покращувати досвід користувача. Звісно, ключовим функціоналом, а, отже, і ключовим аспектом для аналітики є самі лічильники, тому варто розробити власні події для Google Analytics саме для цього компоненту. Вся аналітика застосунку буде знаходитися в класі `AppAnalyticsImpl`, який реалізовуватиме інтерфейс `AppAnalytics`, за допомогою якого надаватиметься доступ лише до необхідних змінних та методів.

Лістинг методу для відслідковування дій з лічильниками:

```
override fun trackCounterAction(screenName: ScreenName, actionName:
EventNames, counterJoined: CounterJoined, isSuccess: Boolean) {

    firebaseAnalytics.logEvent(
        actionName.eventName,
        bundleOf(
            Parameters.Id.paramName to counterJoined.counter.id,
            Parameters.CounterName.paramName to counterJoined.counter.name,
            Parameters.StartingValue.paramName to
counterJoined.counter.startingValue,
            Parameters.Value.paramName to counterJoined.counter.value,
            Parameters.Step.paramName to counterJoined.counter.step,
            Parameters.IsGroup.paramName to counterJoined.counter.isGroup,
            Parameters.CreationDate.paramName to
counterJoined.counter.creationDate,

            Parameters.TagId.paramName to (counterJoined.tag?.id ?: 0L),
```

```

        Parameters.TagName.paramName to
counterJoined.tag?.name.orEmpty(),

        Parameters.IsTagUserCreated.paramName to
(counterJoined.tag?.isUserCreated ?: false),

        Parameters.GroupCountersCount.paramName to
(counterJoined.groupCounters?.size ?: 0),

        Parameters.IsSuccess.paramName to isSuccess
    ).addScreenParams(screenName).addCommonParams()
)
}

```

Також з окремими лічильниками пов'язана логіка з тегами, тому дії над ними також варті уваги для аналізу в Firebase Analytics.

Лістинг методу для відслідковування дій з тегами:

```

override fun trackTagAction(screenName: ScreenName, actionName: EventNames,
tag: Tag, isSuccess: Boolean, tagId: Long?) {

    firebaseAnalytics.logEvent(actionName.eventName,

        bundleOf(

            Parameters.TagId.paramName to (tagId ?: tag.id),

            Parameters.TagName.paramName to tag.name,

            Parameters.IsTagUserCreated.paramName to tag.isUserCreated

        ).addScreenParams(screenName).addCommonParams()

    )

}

```

Хоч попередні 2 методи є важливими для аналітики, все одно в застосунку є дії користувача, які теж було б добре відслідковувати. Наприклад, такі як:

відображення екрану, поширення застосунку через опцію в налаштуваннях та самостійне ініціювання користувачем синхронізації даних.

Лістинг методів інших подій аналітики:

```

override fun screenViewed(screenName: ScreenName) {

    firebaseAnalytics.logEvent(

        EventNames.ScreenViewed.eventName,

        Bundle().addScreenParams(screenName).addCommonParams()

    )
}

override fun trackAppShare() {

    firebaseAnalytics.logEvent(

        EventNames.ShareFromApp.eventName,

        Bundle().addScreenParams(ScreenName.Settings).addCommonParams()

    )
}

override fun trackManualSyncingInitiated() {

    firebaseAnalytics.logEvent(

        EventNames.ManualSyncing.eventName,

        Bundle().addScreenParams(ScreenName.Settings).addCommonParams()

    )
}

```

Тепер окремий клас для аналітики готовий і залишилося лише використати його на рівні ViewModel у потрібних місцях. Заради простішого надання залежностей цим класам варто використати підхід Dependency Injection та найпопулярнішу бібліотеку для цього для розробки для ОС Android – Hilt.

Після успішної імплементації в застосунку потрібно буде лише зробити додаткові налаштування в Firebase Analytics і події, що генерують користувачі, будуть доступні до аналізу.

Висновки до розділу 2

У рамках цього розділу було проаналізовано аналоги програмного продукту, доступні в Google Play, виокремлено їх переваги та недоліки. Далі було проаналізовано можливі архітектурні патерни для використання під час розробки та обрано найбільш підходящий з них. Наступним кроком був вибір інструментарію для розробки застосунку для ОС Android, де було розглянуто різні можливі інструменти, виокремлено їх переваги та недоліки, та в результаті обрано ті, що будуть найкращими для цього програмного продукту. Одним з кінцевих етапів цього розділу було проектування бази даних застосунку, а також формування логіки хмарної синхронізації. В останньому розділі було розроблено модуль для відслідковування подій, генерованих користувачами під час користування застосунком.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ЗАСТОСУНКОМ ДЛЯ ЛІЧИЛЬНИКІВ

3.1. Порядок встановлення застосунку

Оскільки для цієї дипломної роботи необхідні аналітичні дані від реальних користувачів, було зроблено рішення щодо публікації застосунку на платформі Google Play. Тому задля встановлення програмного продукту потрібно в пошуку Google Play ввести назву «Countouch» та перейти на відповідний екран.

Щодо вимог самого застосунку:

- мінімальна версія ОС Android 8.0, котрий покриває більше 90 відсотків користувачів цієї операційної системи
- 15 МБ вільного місця на пристрої

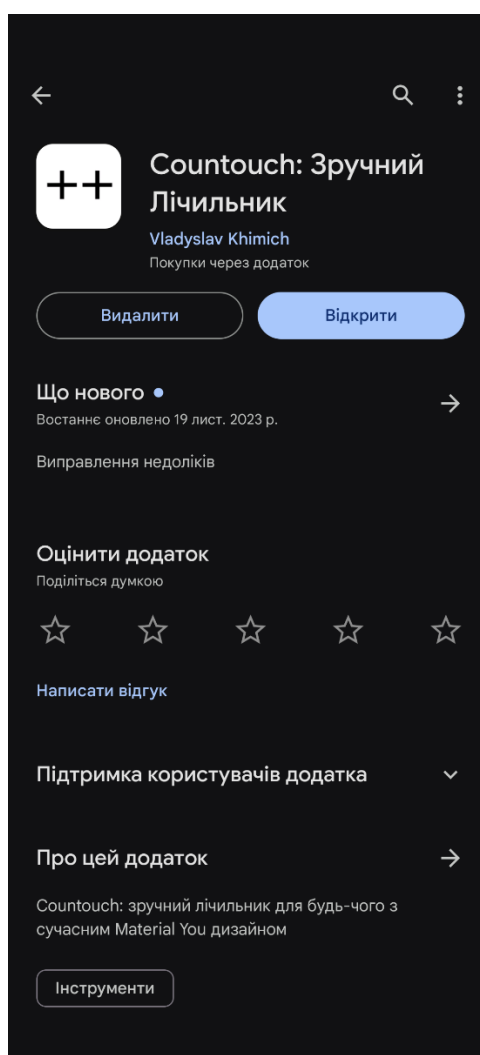


Рис. 3.1. Сторінка застосунку в Google Play

3.2. Структура інтерфейсу. Інтерфейс та порядок роботи з застосунком для лічильників.

При першому запуску користувач опиняється на головному екрані, де при відсутності лічильників відображається плейсхолдер зі вказівками щодо того, як створити новий лічильник. Зверху у Toolbar знаходиться назва застосунку, іконка для відкриття бокового меню та індикатор статусу синхронізації, котра доступна лише за підпискою.

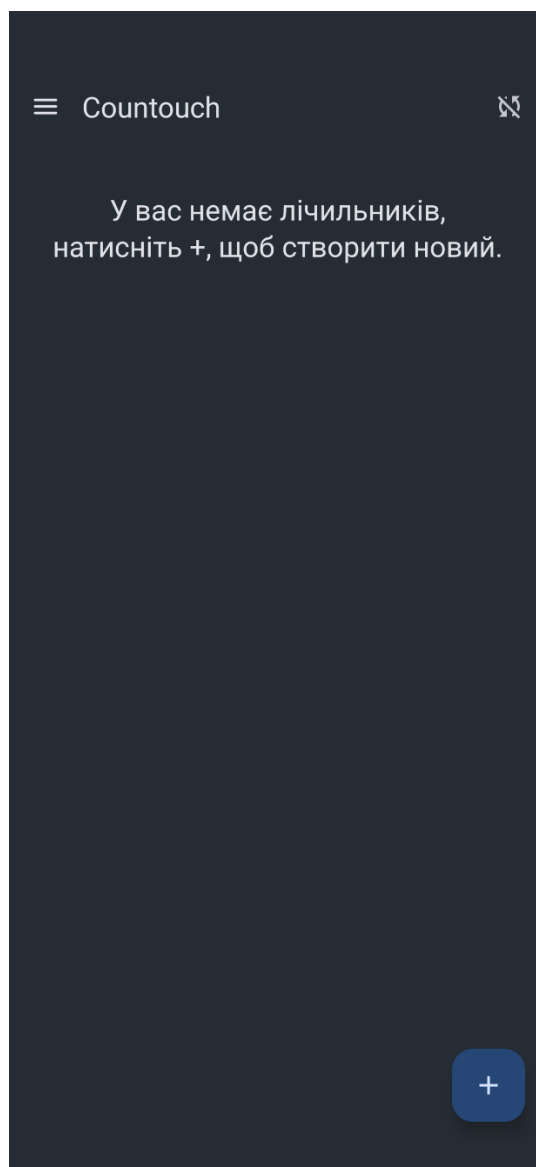


Рис. 3.2. Головний екран застосунку при першому вході

Далі при натисканні на кнопку «+» в нижньому правому куті з'являються 2 опції створення лічильників: окремий та груповий (рис. 3.3 на наступній сторінці).



Рис. 3.3. Опції створення лічильників на головному екрані.

Для першого прикладу оберемо окремий лічильник. Задля простоти цей екран має невелику, але необхідну кількість інформації про лічильник, яку можна додати. Серед них обов'язковими є назва, початкове значення та крок. Також можливо додати опис та прикріпити тег до створюваного лічильника.

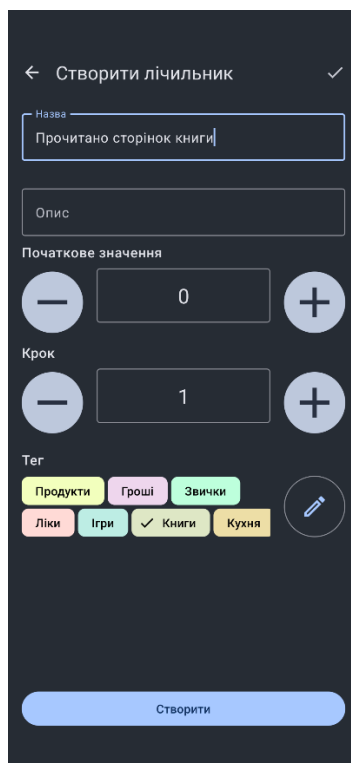


Рис. 3.4. Приклад заповненого екрану окремого лічильника.

Також, натиснувши на кнопку редагування, можна додати новий або видалити власний тег. При натисненні на кнопку з іконкою «+» показується діалог з полем для введення назви нового тегу.

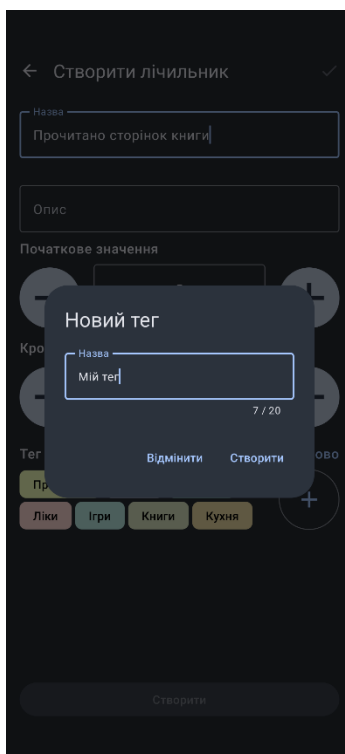


Рис. 3.5. Діалог для введення назви власного тегу.

Після натиснення кнопки «Створити» користувач повертається на головний екран, де можна побачити створений ним лічильник.

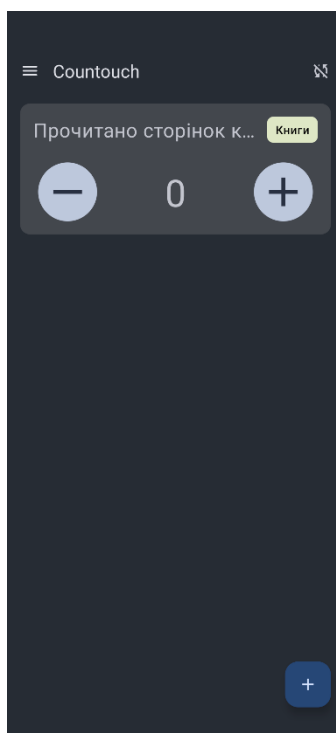


Рис. 3.6. Головний екран зі створеним лічильником.

Наступним прикладом роботи з лічильником буде взаємодія з його груповою версією. На екрані створення групового лічильника присутні майже всі ті самі поля, окрім тегу, а також основною його відмінністю є можливість створювати лічильники в рамках цієї групи, котрі матимуть однакове початкове значення та крок. Також можна змінювати порядок цих лічильників довгим натисканням на кожен з них.

Рис. 3.7. Екран створення групового лічильника.

Після натиснення кнопки «Створити» користувач повертається на головний екран, де до списку всіх лічильників додається новий створений груповий лічильник.

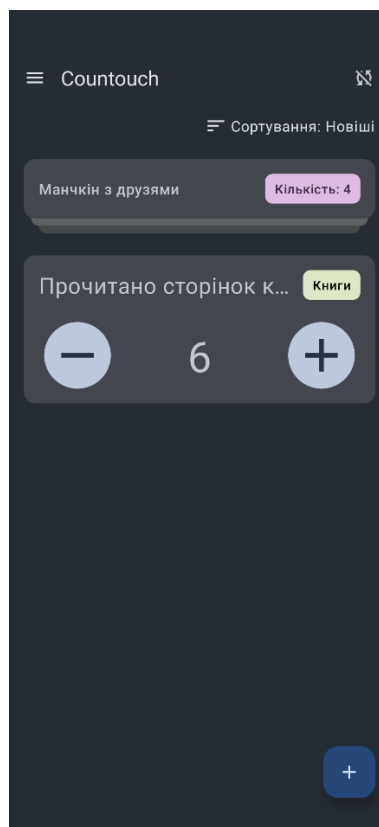


Рис. 3.8. Головний екран зі створеним груповим лічильником.

При натисненні на груповий лічильник відкриється екран з відображенням списку лічильників у цій групі, де можна буде змінювати їх значення.

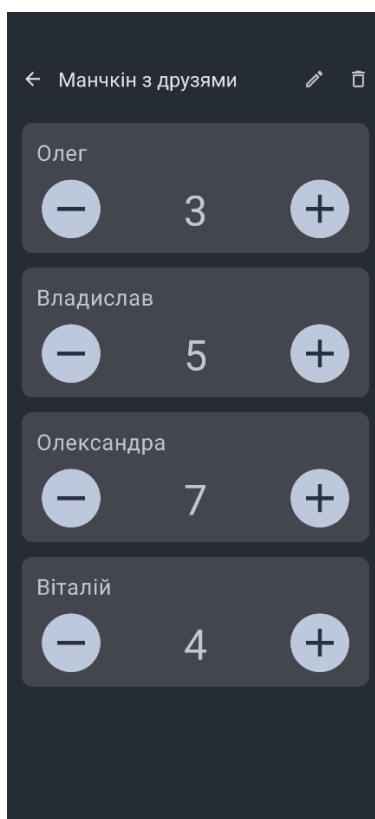


Рис. 3.9. Екран відображення лічильників у групі

Також на головному екрані доступні опції сортування лічильників за різними параметрами.

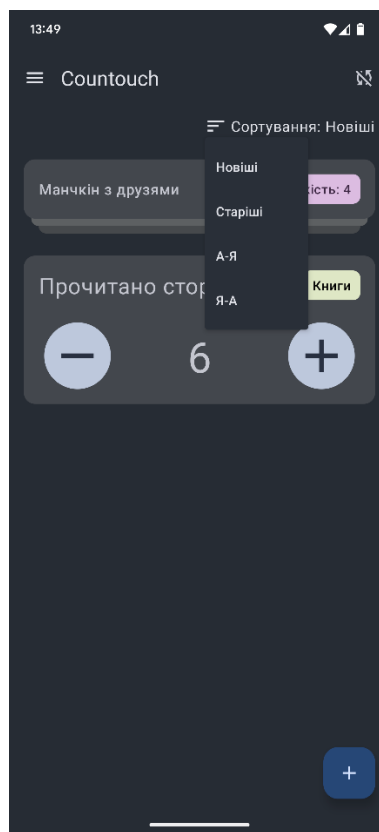


Рис. 3.10. Опції сортування лічильників на головному екрані.
Далі розглянемо інші екрани у боковому меню.

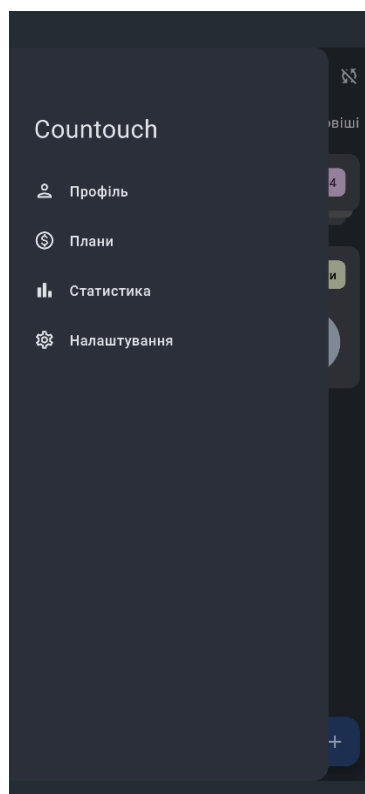


Рис. 3.11. Бокове меню доступне на головному екрані.

Першим екраном з наявних є екран профілю і він буде заповнений невеликою кількістю інформації тільки після того, як користувач автентифікується за допомогою акаунту Google. Без цього входу на ньому відображається пояснювальний текст та кнопка входу.

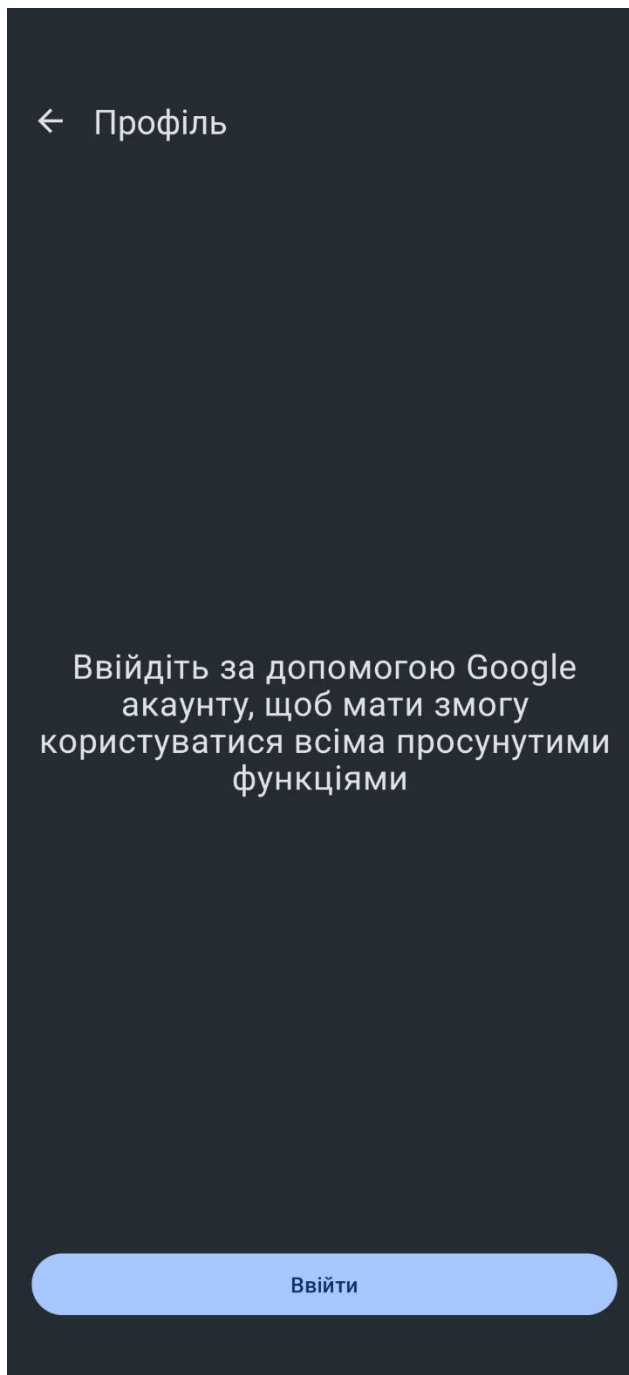


Рис. 3.12. Екран профілю без здійсненого входу.

Наступним екраном з бокового меню є екран «Плани», де відображаються доступні для придбання підписки. Наприклад, підписка потрібна для можливості синхронізації лічильників з хмарним сховищем.

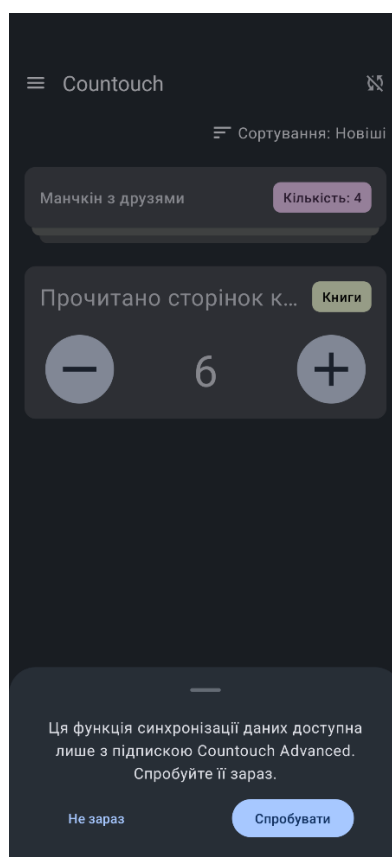


Рис. 3.13. Підказка користувачу щодо доступності функції синхронізації лише з підпискою.

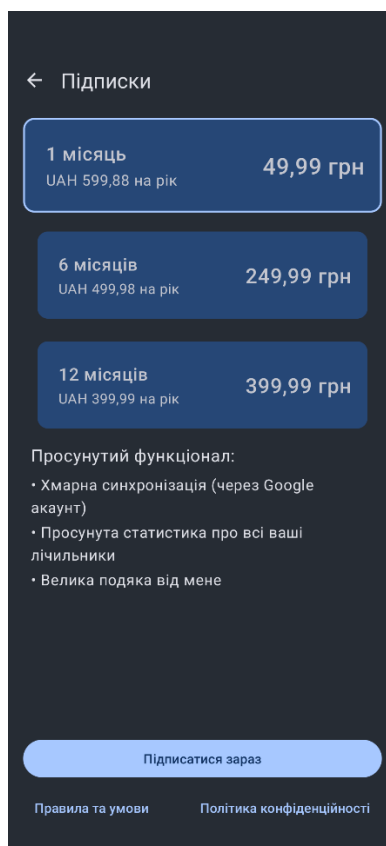


Рис. 3.14. Екран з доступними підписками.

При виборі підписки не автентифікованому користувачу буде показано необхідність входу через акаунт Google, який є обов'язковим для подальшої купівлі підписки. Саме придбання підписки відбувається за допомогою стандартного billing flow від Google Play.

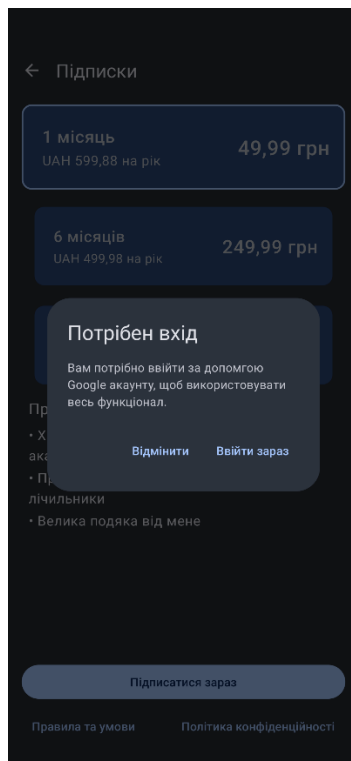


Рис. 3.15. Сповіщення користувача про те, що необхідний вхід.

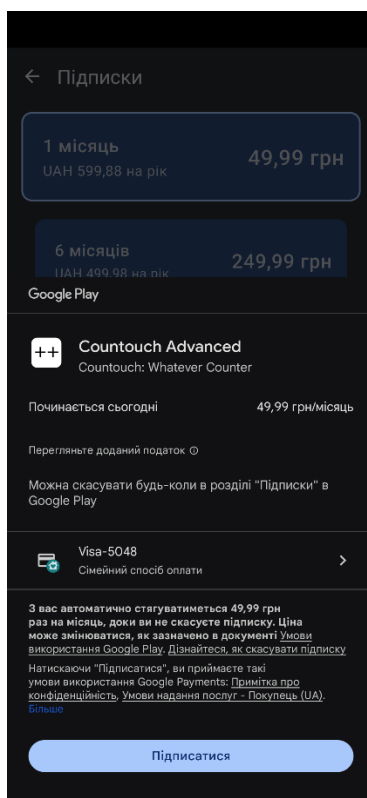


Рис. 3.16. Придбання підписки в застосунку.

Після придбання підписки також стає доступною функція статистики, до котрої можна перейти з бокового меню. Цей екран виглядає наступним чином:

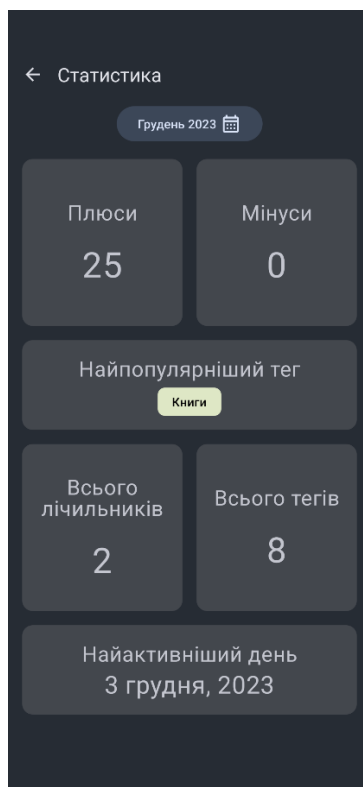


Рис. 3.17. Екран статистики за місяць.

Останнім екраном є екран налаштувань.

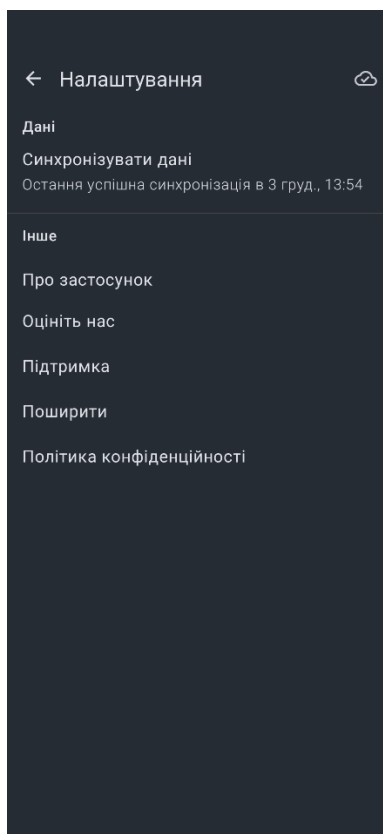


Рис. 3.18. Екран налаштувань застосунку.

Таблиця 3.1.

Опис та призначення налаштувань застосунку

Назва	Опис
Синхронізувати дані	При натисненні буде здійснена спроба синхронізації з хмарним сховищем. Це можливість ініціювати її самостійно, а не чекати на автоматичну синхронізацію.
Про застосунок	Показ інформації про застосунок.
Оцініть нас	Показ діалогу, після вибору оцінки на якому здійснюється перехід в Google Play.
Підтримка	Перехід в поштовий застосунок присутній на пристрої із заздалегідь заповненими даними в листі, такими як: отримувач, тема листа.
Поширити	Відкриття стандартного засобу поширення тексту з посиланням на застосунок.
Політика конфіденційності	Відкриття політики конфіденційності в браузері.

3.3. Тестування роботи системи

Перевірка на працездатність – невід’ємний процес під час розробки будь-якого застосунку.

Тестування функціональності. Нижче наведено тест-кейси роботи застосунку.

Кнопка «Створити»:

передумови: користувач повинен знаходитися на вкладці «Підписки» головного екрану застосунку;

кроки: натискання кнопки «Створити»;

очікуваний результат: виїжджаюча анімація знизу з наявними опціями;

фактичний результат: виїжджаюча анімація знизу з наявними опціями.

Одна з кнопок створення лічильника:

Передумови: користувач повинен знаходитися на головному екрані застосунку та кнопка «Створити» має бути натиснута;

Кроки: Натискання однієї з опцій створення;

Очікуваний результат: Перехід на відповідний екран створення лічильника;

Фактичний результат: Перехід на відповідний екран створення лічильника;

Введення некоректних даних при створенні лічильника:

передумови: користувач повинен знаходитися на екрані створення певного типу лічильника.

кроки: введення некоректних даних у валідовані поля;

очікуваний результат: вимкнення можливості створення лічильника, показ помилки біля поля з некоректними даними;

фактичний результат: вимкнення можливості створення лічильника, показ помилки біля поля з некоректними даними.

Створення лічильника:

передумови: користувач повинен ввести всі необхідні дані у правильному форматі на екрані створення певного типу лічильника;

кроки: натискання кнопки «Створити»;

очікуваний результат: додання лічильника, повернення на головний екран та його відображення;

фактичний результат: додання лічильника, повернення на головний екран та його відображення.

Зміна порядку лічильників у групі:

передумови: користувач повинен мати груповий лічильник з мінімум 2 лічильниками;

кроки: перехід на екран редагування групового лічильника, зміна позицій лічильників у групі за допомогою довгого натискання, натискання кнопки зберегти.

очікуваний результат: повернення на екран показу групового лічильника зі зміненою послідовністю лічильників у групі;

фактичний результат: повернення на екран показу групового лічильника зі зміненою послідовністю лічильників у групі.

Придбання підписки:

передумови: користувач через бокове меню повинен перейти на екран «Плани»;

кроки: вибір бажаного плану, натиснення кнопки «Підписатися зараз».

очікуваний результат: показ діалогу про необхідний вхід, запуск придбання підписки через Google Play, при успішному придбанні повернення на головний екран та запуск синхронізації;

фактичний результат: показ діалогу про необхідний вхід, запуск придбання підписки через Google Play, при успішному придбанні повернення на головний екран та запуск синхронізації.

Висновки до розділу 3

У цьому розділі було наведено порядок встановлення та додатку.

Далі було показано вигляд інтерфейсу застосунку та порядок роботи.

В кінці було проведено функціональне тестування застосунку. Застосунок для лічильників працює коректно та виконує поставлені задачі.

РОЗДІЛ 4. АНАЛІЗ ТА ІНТЕРПРЕТАЦІЯ ОТРИМАНИХ АНАЛІТИЧНИХ ДАНИХ

4.1. Аналіз аналітичних даних згенерованих користувачами

Тепер після певного часу як застосунок доступний користувачам ОС Android у магазині застосунків Google Play, можна розглянути зібрані аналітичні дані та зробити припущення щодо покращень аспектів застосунку, що збагатять користувацький досвід.

Першим на розгляді буде власна подія «create_counter», яка, звісно, за суттю є чи не найважливішою подією у програмному продукті, адже стосується її основного функціоналу. Спочатку варто розглянути тип лічильників, які створюють користувачі, в чому допоможе параметр «screen_name» цієї події.

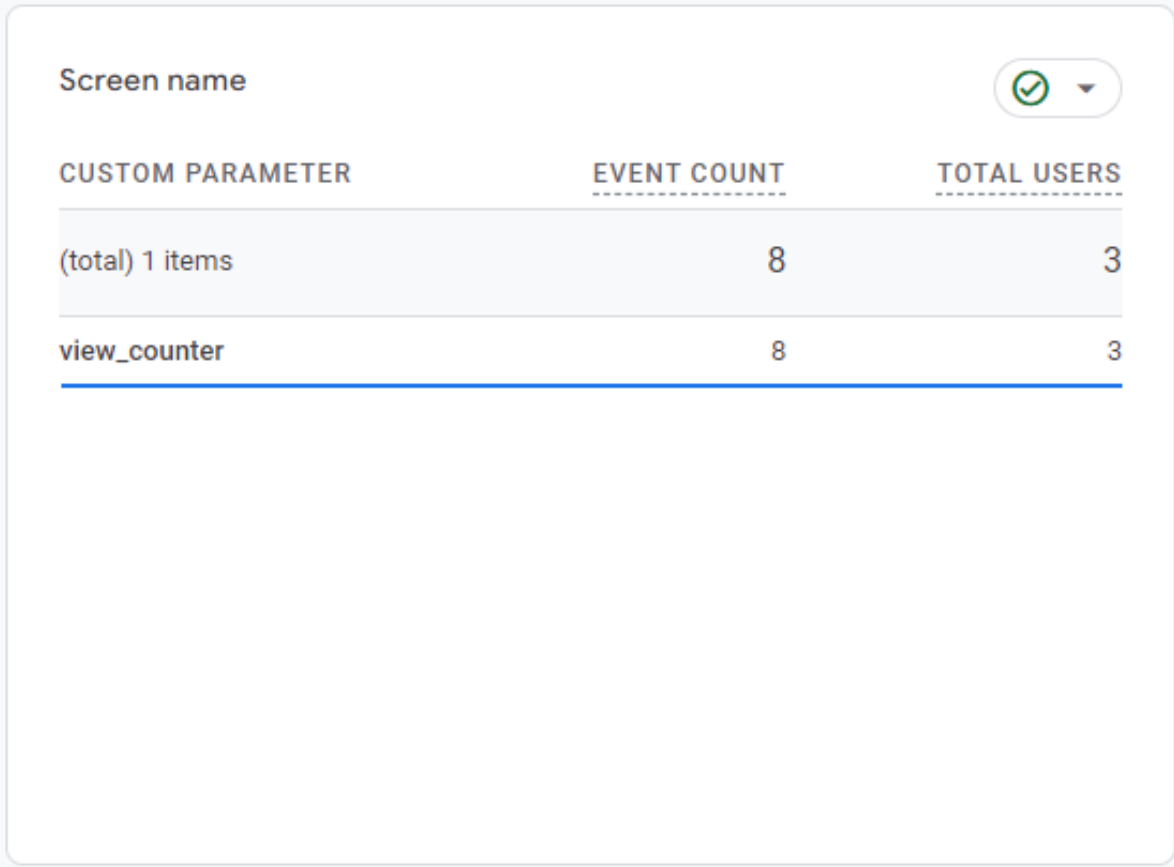
CUSTOM PARAMETER	EVENT COUNT	TOTAL USERS
(total) 3 items	24	12
create_counter	18	11
create_group_counter	5	5
(not set)	1	1

Рис. 4.1. Розподіл створених лічильників за типом.

Як бачимо, в більшості користувачі створюють окремий лічильник, а не груповий, з чого можна зробити наступні висновки: при подальшій розробці та

покращенні варто орієнтуватися на функціонал, що буде покращувати взаємодію з окремим лічильником, або стосуватиметься обох видів лічильників.

Наступна подія варта уваги також стосується лічильника, але на цей раз його видалення, в Firebase Analytics ця подія називається «delete_counter».



CUSTOM PARAMETER	EVENT COUNT	TOTAL USERS
(total) 1 items	8	3
view_counter	8	3

Рис. 4.2. Кількість подій видалення лічильника

Беручи до уваги кількість подій створення лічильника (24) та видалення лічильника (8) можна зробити висновок, що близько третини лічильників в результаті видаляються, що є не дуже хорошим показником, а, отже, потрібно знайти спосіб утримання кількості лічильників у користувача.

Також важливим для розуміння взаємодії користувачів із застосунком є подія «screen_viewed», яка викликається при кожному відкритті кожного екрану програмного продукту.

Screen name ✓ ▾

CUSTOM PARAMETER	EVENT COUNT	TOTAL USERS
(total) 11 items	646	43
main_screen	256	43
view_group_counter	114	7
settings	60	16
(not set)	59	2
plans	49	18
create_counter	32	14
...	...	...

Рис. 4.3. Розподіл подій показу екранів.

Screen name ✓ ▾

CUSTOM PARAMETER	EVENT COUNT	TOTAL USERS
(total) 11 items	646	43
...	...	...
create_counter	32	14
profile	24	9
view_counter	20	6
create_group_counter	15	11
edit_group_counter	15	5
statistics	2	1
...	...	...

Рис. 4.4. Розподіл подій показу екранів.

З наявних даних можна зробити наступні висновки:

- більшість користувачів змінюють значення окремого лічильника з головного екрану, адже подія «screen_viewed» з назвою екрану view_counter була викликана лише 6-ма користувачами з всього 43 (14% користувачів);
- більше половини користувачів не відвідували екрани, що знаходяться в боковому меню на головному екрані. Так, наприклад, найбільшим показником серед 4-х доступних екранів є екран plans, на котрий перейшло 18 користувачів з 43 (41% користувачів);
- Екран перегляду та зміни значень групового лічильника має велику кількість подій, оскільки його значення можна змінювати лише з цього екрану, але при цьому кількість користувачів, що взагалі використовують групові лічильники, не надто велика і складає близько 16%. Це ще раз свідчить про те, що групові лічильники є другорядною функцією для користувачів, однак все ж має свою певну аудиторію.

Останнім аспектом внутрішньої аналітики для розгляду буде не конкретна подія, а розподілення користувачів за країнами.

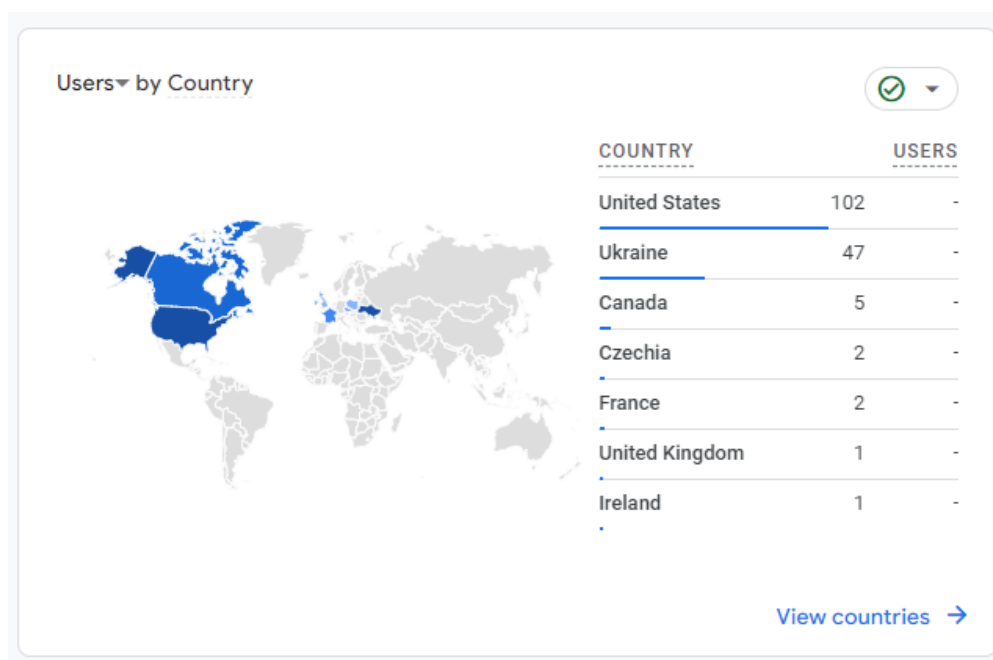


Рис. 4.5. Розподіл користувачів за країнами.

Як бачимо з наведеної вище статистики, ключова аудиторія застосунку знаходиться у двох країнах: США та Україна, що підказує, що подальший функціонал, який додаватиметься у програмний продукт, можна орієнтувати саме на ці ринки.

4.2. Способи покращення застосунку на основі отриманих аналітичних даних

Першим аспектом для покращення згідно аналізу проведеного в попередньому пункті буде запобігання видалення користувачем окремого лічильника. Для цієї мети можна використати наступне рішення: додати для всіх видів лічильників можливість їх архівації. Заархівовані лічильники будуть приховані у списку всіх лічильників та показуватимуться у вигляді списку лише після натиснення на таку категорію лічильників. Такий підхід допоможе користувачам перевикористовувати лічильники, а також покращить користувацький досвід при пошуку певної інформації про старі лічильники, яка може знадобитися користувачу у певний момент. Також це дотримується висновку, зробленому раніше, що для покращення користувацького досвіду необхідно доповнювати функціонал, що стосується окремого або обох типів лічильників.

Ще однією частиною застосунку, яку варто покращити, це інформування користувачів щодо наявності певного функціоналу. Так, наприклад, лише максимум 41% користувачів здійснив перехід на екрани, що знаходяться у боковому меню на головному екрані. Ключовим з них є екран з підписками, однак певна кількість користувачів не знає про нього, що підтверджується власним досвідом з відгуком реального користувача. Як же можна покращити цю ситуацію?

Для інформованості щодо наявності цих екранів у боковому меню варто при першому запуску застосунку зробити користувачу певний показ того, що доступно на головному екрані. Багато застосунків для ОС Android

використовують такий підхід, використовуючи стрілки та текст з поясненням разом з привабливими анімаціями.

В той же час для кращого спонукання користувачів до переходу на екран підписок можна використати ненав'язливий банер на головному екрані, котрий можна приховати, натиснувши на хрестик. І хоч цей банер можна буде приховати, користувач все одно з більшою вірогідністю прочитає повідомлення цього банеру.

Останнім аспектом, що потребує розгляду у цьому розділі, буде просування програмного продукту на існуючих та нових ринках. З минулого розділу бачимо, що основними країнами поширення є Україна та США. Першою можливою зміною для збільшення кількості придбаних підписок в обох регіонах є зміна цін в меншу сторону на певні державні свята, наприклад, в США це може бути День подяки, а в Україні День праці. Також знижки на підписки можуть бути приурочені загальним подіям у світі продажів, як-от Чорна п'ятниця чи День шопінгу.

Ще однією складовою, що потенційно допоможе розширити аудиторію застосунку це додання інших мов у програмний продукт. Наприклад, іспанська мова є однією з найпопулярніших мов світу і налічує 534 мільйони носіїв у 22-х країнах. Додання цієї мови до строкових ресурсів програмного застосунку допоможе розширити географію користувачів, які до того ж будуть різнитися за рівнем платоспроможності, адже іспанська поширена як і в слабо, так і сильно розвинених економічно країнах.

Висновки до розділу 4

У цьому розділі було проаналізовано аналітичні дані, отримані від користувачів, та зроблено висновки щодо отриманих даних.

Далі на основі аналізу було запропоновано варіанти покращення застосунку для покращення користувацького досвіду та збільшення кількості придбаних підписок.

ВИСНОВКИ

Під час виконання випускної роботи магістра було спроектовано, реалізовано та опубліковано в Google Play застосунок для лічильників для ОС Android. Далі було здійснено аналіз отриманих аналітичних даних користувачів під час користування програмним продуктом, зроблено висновки щодо наявних даних та запропоновано варіанти подальшого покращення застосунку. Для досягнення цього було виконано наступні завдання:

1. Було здійснено аналіз засобів збору даних та було обрано засоби для використання у власному застосунку для ОС Android, а саме: Firebase Analytics для власних подій та Crashlytics для збору інформації про вильоти та інші технічні проблеми застосунку.

2. Також на основі засобів збору даних, що використовуватимуться у застосунку, було обрано способи прогнозування поведінки користувачів: аналіз використання додатків та аналіз відмов.

3. Далі було оглянуто наявні програмні аналоги доступні в Google Play та віднайдено аспекти, на які потрібно зробити акцент при розробці власного застосунку для лічильників.

Цими аспектами є:

- знаходження балансу між функціональністю та простотою користування для кінцевого користувача
- дотримання останніх рекомендацій Google для їхньої мови дизайну Material You для розробки зрозумілого та привабливого інтерфейсу
- відсутність реклами є перевагою, а підписка повинна бути присутньою лише для додаткового функціоналу
- можливість створення групових лічильників та хмарна синхронізація може є конкурентною перевагою

4. Було обрано головний архітектурний патерн для застосунку – Model-View-ViewModel. Також при аналізі було виділено необхідний набір технологій та програмних продуктів для створення застосунку: нативну розробку з

використанням Android SDK, Kotlin, Compose; Firebase Analytics для аналітики; середовище розробки Android Studio; продукт для розробки макету Figma.

5. Для збереження даних користувачів було спроектовано базу даних та описано поля таблиць. Також була сформована загальна логіка збереження користувацьких даних у хмарному сховищі.

6. Далі було здійснено процес розробки застосунку, разом з аналітичним модулем, процес імплементації якого описано у цьому звіті. Також було проведено бета-тестування, виправлення недоліків та подальша публікація застосунку для всіх користувачів ОС Android через Google Play.

7. Також було продемонстровано функціонал при роботі кінцевого користувача з програмним продуктом.

8. Після цього було проведено функціональне тестування застосунку. Застосунок для лічильників працює коректно та виконує поставлені задачі.

9. Було зібрано події аналітики від реальних користувачів за певний проміжок часу, після чого здійснено їх аналіз та запропоновано варіанти покращення користувацького досвіду та популяризації застосунку.

Реалізований програмний продукт є готовим до використання. Аналітичні дані, що збираються в застосунку на основі користування реальними користувачами, можуть використовуватися для їх аналізу та формуванню подальших варіантів покращення користувацького досвіду та популяризації цього програмного продукту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Хіміч В.О. Минуле, теперішнє та майбутнє способів розробки користувацьких інтерфейсів для ОС Android / В.О. Хіміч // VI Всеукраїнська науково-технічна конференція., [Житомир] 29-30 лист. 2023 р.
2. Mateus B. On the adoption, usage and evolution of Kotlin features in Android development / B. Mateus, M. Martinez. // ESEM \20: Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). – 2020. – С. 1–12.
3. Ardito L. Effectiveness of Kotlin vs. Java in android app development tasks / L. Ardito, R. Coppola. // Information and Software Technology Volume 127. – 2020.
4. Saptoto R. A Comparative Study of Java and Kotlin for Android Mobile Application Development / Saptoto R. // 2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI). – 2020.
5. Hecht G. Quantifying the adoption of Kotlin on Android stores: Insight from the bytecode / G. Hecht, A. Bergel. // 2021 IEEE/ACM 8th International Conference on Mobile Software Engineering and Systems (MobileSoft). – 2021. – С. 94–98.
6. Noori Z. UI Performance Comparison of Jetpack Compose and XML in Native Android Applications / Z. Noori, C. Eriksson. – 2023. – С. 78.
7. Szczukin J. Performance analysis of user interface implementation methods in mobile applications / Jakub Szczukin. // Journal of Computer Sciences Institute. – 2023. – №26. – С. 13–17.
8. Vilhunen A. User interface test automation for an Android application / Atte Vilhunen. // Sähkötekniikan korkeakoulu. – 2022. – С. 49–68.
9. Spath P. User Interface / Peter Spath. // Pro Android with Kotlin. – 2022. – С. 257–361.

10. Harty J. Logging Practices with Mobile Analytics: An Empirical Study on Firebase / Julian Harty. // 2021 IEEE/ACM 8th International Conference on Mobile Software Engineering and Systems (MobileSoft). – 2021.
11. Moroney L. Google Analytics for Firebase / Laurence Moroney. // The Definitive Guide to Firebase. – 2017. – C. 251–270.
12. Moroney L. An Introduction to Firebase / Laurence Moroney. // The Definitive Guide to Firebase. – 2017. – C. 1–24.
13. Beschi R. An Efficient Android APP for Attendance Record and Management System / Raja Beschi. // IJETIE. – 2019. – №5.
14. Ranise S. Assessing the Effectiveness of the Shared Responsibility Model for Cloud Databases: the Case of Google’s Firebase / Silvio Ranise. // 2021 IEEE International Conference on Smart Data Services (SMDS). – 2021.
15. Sukmana Y. The Use of Cloud Firestore For Handling Real-time Data Updates: An Empirical Study of Gamified Online Quiz / Yuda Sukmana. // 2021 2nd International Conference on Electronics, Communications and Information Technology (CECIT). – 2021.
16. Tian Y. What and How long: Prediction of Mobile App Engagement / Yuan Tian. // ACM Transactions on Information Systems. – 2021. – №40. – C. 1–38
17. Pan Y. On mining mobile apps usage behavior for predicting apps usage in smartphones / Yi-Chin Pan. // CIKM '13: Proceedings of the 22nd ACM international conference on Information & Knowledge Management. – 2013. – C. 609–618.
18. Prediction of user app usage behavior from geo-spatial data. // GeoRich '16: Proceedings of the Third International ACM SIGMOD Workshop on Managing and Mining Enriched Geo-Spatial Data. – 2016. – C. 1–6.
19. Pierce L. Characterizing and Forecasting User Engagement with In-App Action Graph: A Case Study of Snapchat / Lucas Pierce. // KDD '19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. – 2019. – C. 2023–2031.

20. Wang A. Research on App Usage Behavior Analysis and Prediction Algorithms for Mobile Users / Aihua Wang. // 2019 11th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA). – 2019.
21. Hsu C. What drives purchase intention for paid mobile apps? – An expectation confirmation model with perceived value / Chin-Lung Hsu. // Electronic Commerce Research and Applications. – 2015. – №14. – С. 46–57.
22. Патерни проєктування [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: <https://refactoring.guru/uk/design-patterns>
- 23.Будай А. Дизайн-патерни - просто, як двері / Андрій Будай., 2016.
- 24.Stack Overflow [Електронний ресурс] – Режим доступу до ресурсу: <https://stackoverflow.com/>.
- 25.Офіційна документація Android [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.android.com/docs> .
- 26.Sierra K. Head First Java, 2nd Edition / K. Sierra, B. Bates., 2005.
27. Phillips B. Android Programming: The Big Nerd Ranch Guide / B. Phillips, B. Hardy., 2019.
- 28.Medium. Android Developers. [Електронний ресурс] – Режим доступу до ресурсу: <https://medium.com/androiddevelopers>
- 29.Google Analytics for Firebase. Офіційна документація. [Електронний ресурс] – Режим доступу до ресурсу: <https://firebase.google.com/docs/analytics>.
30. Firebase Crashlytics. Офіційна документація. [Електронний ресурс] – Режим доступу до ресурсу: <https://firebase.google.com/docs/crashlytics>