МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет економіки, підприємництва та інформаційних технологій

Кафедра інформаційних технологій

Допущено до захисту

«____»_____2026 р.

Зав. кафедри, к.т.н., доцент

_____Ліщук Р.І.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ДОГЛЯДУ ЗА ДОМАШНІМИ СОБАКАМИ

Виконав – студент 4 курсу, групи 41 з-кн освітньої програми "Комп’ютерні науки" спеціальності 122 Комп’ютерні науки

Мальований Євгеній Михайлович

Науковий керівник – к.е.н., доцент

Концеба Сергій Михайлович

Умань – 2026

Реферат
на кваліфікаційну роботу "Розробка мобільного застосунку для догляду за домашніми собаками"

студента факультету економіки, підприємництва та інформаційних технологій
Уманського національного університету
Мальованого Євгенія Михайловича

Структура та обсяг роботи. Пояснювальна записка кваліфікаційної роботи складається з чотирьох розділів, містить 62 сторінки, 7 таблиць, 15 рисунків, 1 додаток, 22 джерел наукової літератури.

Метою цієї кваліфікаційної роботи бакалавра є проєктування та реалізація інноваційного мобільного застосунку, орієнтованого на комплексний догляд за собаками, який спрощує повсякденні справи їхніх власників.

У першому розділі проаналізовано проблеми розробки та тестування мобільних застосунків, використання цифрових технологій у догляді за собаками, також здійснено аналіз мультиплатформних мов програмування та мов програмування для однієї платформи.

У другому розділі здійснено аналіз фаз створення застосунку: фаза концепції мобільного застосунку, фаза технічного планування, фаза розробки та реалізації, фаза тестування та оптимізації

У третьому розділі реалізовано мобільний застосунок для догляду за домашніми собаками.

У четвертому розділі реалізовано мобільний застосунок для догляду за домашніми собаками.

Анотація

У роботі досліджуються сучасні тенденції розробки мобільних застосунків та використання мультиплатформних технологій. Застосунок було розроблено мовою Kotlin за допомогою таких інструментів, як Android Studio, Firebase, Firestore та Google Maps API. Серед ключових функцій – планування прогулянок, мапа собачих парків, ветеринарний паспорт із QR-кодом, освітні статті та ігрові елементи, зокрема таблиця лідерів. Під час проєктування було використано методи SWOT- та MoSCoW-аналізу, моделювання персон і тестування MVP за допомогою Robo-тесту та користувацьких сценаріїв. Результати тестування підтвердили функціональність і зручність використання застосунку, а також у роботі запропоновано можливості його подальшого розширення.

Ключові слова: Android, Firebase, гейміфікація, Kotlin, мобільний застосунок, догляд за собаками, тестування.

AnnotationThe paper explores current trends in mobile app development and the use of multiplatform technologies. The application was developed in Kotlin using tools such as Android Studio, Firebase, Firestore, and Google Maps API. Key features include walk planning, a dog park map, a QR code veterinary passport, educational articles, and game elements, including a leaderboard. The design used SWOT and MoSCoW analysis, persona modeling, and MVP testing using Robo-test and user scenarios. The testing results confirmed the functionality and usability of the application, and the paper also suggests possibilities for its further expansion.

Keywords: Android, Firebase, gamification, Kotlin, mobile app, dog care, testing.

ЗМІСТ

ВСТУП

РОЗДІЛ 1 АНАЛІЗ СТАНУ ПРОБЛЕМАТИКИ

1.1 Аналіз проблем розробки та тестування мобільних застосунків

1.2 Аналіз використання цифрових технологій у догляді за собаками

1.3 Аналіз мультиплатформних мов програмування

1.4 Аналіз мов програмування для однієї платформи

РОЗДІЛ 2 ТЕОРЕТИЧНИЙ АНАЛІЗ

2.1 Фаза концепції мобільного застосунку

2.2 Фаза технічного планування

2.3 Фаза розробки та реалізації

2.4 Фаза тестування та оптимізації

РОЗДІЛ 3 ПРОЄКТУВАННЯ ПРАКТИЧНОЇ ЧАСТИНИ

3.1 SWOT-аналіз

3.2 MoSCoW-аналіз

3.3 Аналіз персон

3.4 Варіанти використання

РОЗДІЛ 4 РОЗРОБКА ЗАСТОСУНКУ ТА ТЕСТУВАННЯ

4.1 Технічна розробка застосунку

4.2 Функціональна розробка застосунку

4.3 Тестування

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ВСТУП

Догляд за домашніми улюбленцями, особливо за собаками, набуває в сучасному суспільстві дедалі більшого значення. З огляду на зростання кількості домогосподарств, у яких живуть собаки, та супутні цьому потреби на перший план виходить питання цифровізації цієї сфери. Сучасні технології та мобільні застосунки пропонують широкі можливості для того, щоб зробити такий догляд ефективнішим та простішим. Попри це, на ринку досі бракує комплексних інструментів, які надавали б власникам собак цілісну та централізовану систему для управління інформацією, планування активностей, спілкування, і водночас мотивували б їх до регулярного використання застосунку.

Більшість наявних рішень зосереджується лише на окремих часткових функціях – обліку стану здоров'я, відстеженні місцезнаходження або базовій взаємодії з пристроєм. Однак такі застосунки часто не пропонують зручного для користувача середовища, недостатньо поєднують між собою окремі функціональні можливості та не надають можливості обмінюватися досвідом чи спілкуватися між користувачами. Ці недоліки вказують на потребу запропонувати рішення, яке враховувало б практичні вимоги користувачів і водночас додавало б сучасні мотиваційні елементи, наприклад гейміфікацію.

Метою роботи є проєктування та розробка інноваційного мобільного застосунку для системи Android, який слугуватиме для комплексного догляду за собаками. Застосунок включає такі функції, як планування прогулянок, відображення мап собачих парків, ведення ветеринарного паспорта з QR-кодом, читання спеціалізованих статей та комунікація між власниками собак за допомогою чату. Складовою рішення є також ігрові механіки, які заохочують щоденне використання застосунку.

Для досягнення поставлених цілей застосовано методику, що охоплює SWOT- та MoSCoW-аналіз, проєктування користувацьких персон та варіантів використання, а також тестування MVP-версії за допомогою Robo-тесту та

користувацьких сценаріїв. Під час розробки використано технології Kotlin, Android Studio, Firebase, Firestore та Google Maps API.

Робота ґрунтується на актуальних знаннях у сфері розробки мобільних застосунків, цифрових інструментів для догляду за тваринами та трендів у галузі мотивації користувачів. Підсумковий проєкт та реалізація застосунку є конкретним внеском у цифровізацію догляду за собаками і водночас відкривають простір для подальших досліджень та розвитку у цій сфері.

РОЗДІЛ 1 АНАЛІЗ СТАНУ ПРОБЛЕМАТИКИ

1.1 Аналіз проблем розробки та тестування мобільних застосунків

У сучасних умовах стрімкого розвитку інформаційних технологій мобільні застосунки стали важливою складовою цифрового середовища. У зв’язку з цим значна увага науковців приділяється дослідженню процесів розробки, оптимізації та тестування мобільних застосунків, зокрема для операційної системи Android. У наукових працях розглядаються питання повторного використання програмного коду, забезпечення якості програмного продукту, ефективності методів тестування, а також переваги й недоліки сучасних підходів до створення мобільного програмного забезпечення.

Зокрема, метою наукової праці, авторами якої є Israel J. Mojica Ruiz, Meiyappan Nagappan, Bram Adams та Ahmed E. Hassan, було висвітлення проблеми повторного використання коду на ринку Android Market [1].

Автори досліджують, як часто розробники використовують у своїх застосунках готовий код, що може сприяти швидкому зростанню кількості доступних застосунків на ринку. Наукова праця зосереджується на двох аспектах повторного використання: успадкуванні та повторному використанні класів. З огляду на стрімке зростання кількості застосунків, автори припускають, що значна кількість застосунків використовує наявний код. Ця проблема нині є досить поширеною, оскільки багато програмістів діляться своїм кодом, який можуть використовувати й інші. Що це означає для світу IT? Це означає, що на ринку з'являється безліч різноманітних застосунків, знижуються витрати на розробку, а процес створення нового продукту загалом пришвидшується.

Авторами використовується методика під назвою Software Bertillonage, яка дозволяє відстежувати класи застосунку за їхнім байт-кодом, навіть якщо немає доступу до вихідного коду.

Дослідження проводилося в п'яти різних категоріях застосунків і виявило такі цікаві факти:

- приблизно 50% класів у проаналізованих застосунках успадковуються від базових класів системи Android або від класів, специфічних для певної предметної області;

- у середньому 61% класів у кожній категорії повторно використовуються в кількох застосунках;

- більшість часто використовуваних класів належать до бібліотек третіх сторін, наприклад Google AdMob, які допомагають додавати рекламу в застосунки.

Якщо говорити про використання класів API Android, то кожен застосунок, який був предметом дослідження, тією чи іншою мірою використовував цю функцію.

1.1 Відсоток використовуваних класів з інтерфейсу Android API та з інших класів

Категорія	Відсоток класів API Android	Відсоток інших класів
Cards and Casino	21%	29%
Personalization	29%	26%
Photography	29%	19%
Social	19%	31%
Weather	25%	25%

Джерело: власна обробка на основі [1]

Якщо говорити про переваги цієї методики та рішення, яке автори запропонували для аналізу, можна констатувати наступне: використання готових класів зменшує час і витрати на розробку, підвищує продуктивність розробників та дозволяє використовувати вже перевірені та надійні компоненти, наприклад рекламні бібліотеки.

Однак якщо звернутися до дослідження авторів статті «Testing of Mobile Applications in the Wild», стає зрозуміло, що головний акцент робиться на

тестуванні застосунків у реальних умовах [2]. Проблемою тут є недостатнє тестування застосунків перед випуском, що може призводити до помилок користувачів.

Цей аспект уже можна вважати недоліком, проте є ще один досить неприємний недолік, який я усвідомив не одразу, а лише після третього вивчення статті – при використанні цього методу існує ризик того, що застосунок залежатиме від бібліотек третіх сторін, що може спричинити неприємні проблеми під час оновлення або видалення цих бібліотек.

Інше дослідження описує процес тестування мобільних застосунків у реальному світі та те, як зробити його ефективнішим [2].

Основною проблемою, про яку говорять автори, є обмежене та недостатньо систематичне тестування мобільних застосунків у реальних умовах. З огляду на різноманітність пристроїв, операційних систем і умов використання, у мобільних застосунках можуть виникати помилки, які не завжди виявляються під час лабораторного тестування. Недостатнє тестування може призвести до поганого користувацького досвіду і, як наслідок, до негативних відгуків та втрати користувачів.

Загалом автори звернули увагу на досить важливе питання, оскільки проблема тестування є цілком актуальною: жоден продукт не хоче витрачати свій час і ресурси, як людські, так і фінансові, на детальне тестування, що призводить до появи на ринку неякісних застосунків або, у гіршому випадку, до кібератак через уразливі частини застосунку.

Автори пропонують рішення проблеми, яке може сподобатися не всім, але буде досить ефективним. Вони пропонують тестувати застосунок у реальних умовах та водночас провести дослідження використання різних стратегій тестування серед розробників застосунків для Android. До уваги беруться такі моменти:

- кількість тестових сценаріїв, які використовують розробники;
- рівень покриття коду тестами;

- Assertion density – величина, яка вимірює, як часто розробники використовують твердження для виявлення можливих помилок у коді.

На основі цих рішень можна виокремити такі переваги:

- тестування в реальних умовах дозволяє виявити помилки, які не вдалося б виявити в лабораторії;

- використання систематичних тестів та підвищення щільності перевірок може поліпшити надійність застосунків і збільшити їх стійкість до неочікуваних умов.

Однак не варто забувати про недоліки, які в цій ситуації є досить вагомим фактом при ухваленні рішення про потребу в такому тестуванні:

- тестування в реальному світі є дорожчим і може потребувати додаткових ресурсів та інфраструктури;

- висока щільність перевірок може ускладнювати розробку та підтримку коду, особливо для невеликих команд з обмеженими ресурсами.

Якщо повернутися до першого дослідження, то на відміну від цього рішення, в ній автори не використовували детальне тестування і застосовували готовий код, що дозволило їм заощадити фінансові та часові ресурси [1].

Venkata N. Inukollu, Divya D. Keshamoni, Taeghyun Kang, Manikanta Inukollu досліджують проблему низької якості мобільних застосунків та причини їх невдач на ринку, що стає дедалі актуальнішим з огляду на зростання кількості застосунків і конкуренцію [3].

Вони аналізують фактори, які призводять до поганого користувацького досвіду, та описують, як уникнути таких проблем чи непорозумінь.

Багато сучасних застосунків, які з'являються сьогодні, мають низьку якість та не відповідають очікуванням користувачів. Як наслідок, це призводить до негативних оцінок, втрати цінності та загалом видалення застосунку. Основні причини, які описують автори, такі: технічні проблеми, мінімальне тестування, припинення підтримки деяких бібліотек, відсутність чіткої методології розробки.

Такі проблеми можуть мати радше негативний вплив на ринок мобільних застосунків, оскільки швидко розроблений застосунок виходить на ринок, швидко відбирає конкуренцію в інших і з часом зникає через технічні проблеми, тоді як попередні застосунки не встигають набути достатньої популярності. Тому автори в цій статті презентують свої ідеї, як уникнути таких проблем.

Перше, що потрібно зробити, аби уникнути розробки неякісного застосунку, – це дотримуватись життєвих циклів застосунку. Автори рекомендують використовувати традиційні етапи життєвого циклу розробки: збір вимог, проєктування, розробку, тестування та підтримку. Це дозволяє створювати структуровані та якісні продукти.

Не слід забувати і про варіанти використання, які є необхідними для розуміння того, як застосунок реагуватиме та взаємодіятиме з клієнтом і, наприклад, із сервером. Це допоможе краще і простіше зрозуміти, які проблеми можуть виникнути ще до того, як застосунок з'явиться на ринку.

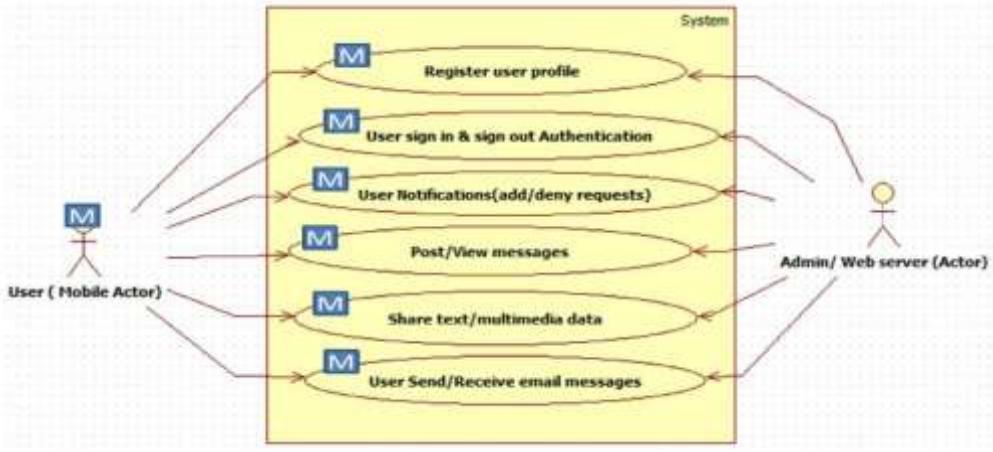


Рис. 1.1 Приклад варіанту використання мобільного застосунку [3]

Ще одним рішенням, яке варто ретельно розглянути та яке було описано у другому дослідженні, є використання перевірених інструментів розробки та тестування [2]. У статті про тестування «в експлуатації» описано, що потрібно якісно реалізувати процес тестування, аби уникнути проблем при виведенні

продукту на ринок та надати кращі послуги, оскільки всі моделі тестування вже будуть описані (принаймні більшість із них).

Однак якщо порівняти обидва дослідження, то у цій прямо не йдеться про настільки детальне тестування, як описано у попередній статті, тому це призведе до втрати ресурсів, що в кінцевому підсумку зумовить можливу невдачу застосунку. Тож тут йдеться про те, що тестування є необхідним, але головне, аби воно не перетворювалося на тривалий процес [2]. Не варто забувати і про якісний маркетинг. Автори рекомендують використовувати соціальні мережі та обирати правильну цільову аудиторію, аби зробити застосунок помітним.

Спочатку слід описати недоліки цих рішень, оскільки вони цілком логічні та прості:

- для проходження всіх етапів життєвого циклу може знадобитися більше часу та ресурсів;

- ефективний маркетинг може потребувати значного бюджету, що ускладнює його застосування для невеликих проєктів;

- висока залежність від зворотного зв'язку може уповільнити процес впровадження нових функцій.

Що стосується переваг, які виглядають доволі привабливо та можуть переконати кожного розробника наслідувати запропоновану авторами стратегію:

- дотримання чіткої структури розробки забезпечує якість, знижує ризик помилок та підвищує стабільність застосунку;

дотримання правил належного тестування допоможе уникнути непередбачуваних проблем та витрат у майбутньому;

- сильний маркетинг забезпечить застосунку значні фінансові надходження та повернення інвестицій, якщо його правильно спроєктовано і він не зашкодить бюджету на запуск.

Аналіз досліджень з окреслної проблематики засвідчив, що для створення конкурентоспроможного мобільного застосунку розробники повинні поєднувати структурований підхід до розробки з ефективним тестуванням і розумним використанням готових рішень. Важливою є збалансована стратегія, яка враховує якість продукту, витрати, час розробки та потреби користувачів. Для забезпечення успіху застосунку на ринку також необхідна інтеграція компетентного маркетингу.

Крім того, варто звертати увагу на гнучкість та масштабованість архітектури застосунку, аби враховувати зміни ринкових умов. Постійний аналіз зворотного зв'язку від користувачів дозволить вчасно виявляти та усувати недоліки продукту.

1.2 Аналіз використання цифрових технологій у догляді за собаками

У сучасному суспільстві цифрові технології дедалі активніше використовуються в різних сферах життя, зокрема й у сфері догляду за домашніми тваринами. Науковці приділяють значну увагу дослідженню можливостей використання мобільних застосунків та інформаційних систем для моніторингу стану здоров’я тварин, організації догляду, взаємодії власників із ветеринарними спеціалістами та підвищення загального рівня відповідального утримання собак. У наукових працях розглядаються сучасні цифрові рішення, їхні функціональні можливості, переваги та недоліки, а також вплив таких технологій на комфорт і безпеку як власників, так і самих тварин. У цьому підрозділі здійснено аналіз наукових джерел, присвячених використанню цифрових технологій у догляді за собаками.

У своїй статті автори Demian C. Johnson, P. Reeja George та Mahesh Chander обговорюють обмежене використання інформаційних технологій у ветеринарній сфері, особливо при взаємодії між власниками собак та ветеринарами [4]. Тому, з огляду на те, що кількість власників собак останніми

роками зростає, автори розробили застосунок VetCan, покликаний полегшити доступ до інформації про здоров'я та догляд за собакою.

Основною проблемою, описаною в цій статті, є недостатній доступ власників собак до ветеринарної інформації. Ця проблема обмежує їхню здатність ефективно піклуватися про своїх тварин. Застосунок VetCan було створено для допомоги у вирішенні проблем зі здоров'ям собак. За допомогою цього застосунку можна отримати всю необхідну інформацію про поширені захворювання, першу допомогу та, не в останню чергу, спланувати візити до лікарів.

Вище вже було описано, як автори вирішили цю проблему – вони створили застосунок, який допомагає власникам собак вирішувати питання здоров'я їхніх тварин.

Однак потрібно описати й інші рішення проблеми відсутності швидкого доступу до ветеринарної інформації:

- розробка релевантного контенту застосунку – вся інформація в застосунку розроблялася у співпраці з ветеринарними лікарями та науковцями таким чином, аби охопити три основні теми: захворювання, менеджмент і додатковий догляд;

- оцінка якості – якість застосунку оцінювалася за параметрами залученості, функціональності, естетики та інформаційного наповнення;

- тестування в реальному житті – тестування проводилося серед власників собак, які потім надали зворотний зв'язок, на основі якого в систему вносилися коригування.

У будь-якому застосунку ми знову повертаємося до етапу якісного тестування, який описано у цьому дослідженні [2]. Завдяки людському тестуванню цей застосунок буде успішним, оскільки розробники під час тестування виявили всі вразливості.

Переваги цього рішення:

- застосунок надає доступ до корисної інформації, яка допомагає власникам собак у щоденному догляді;

- застосунок отримав високі оцінки за функціональність, привабливий дизайн та доступність важливої інформації, що сприяє позитивному користувацькому досвіду.

Однак потрібно зазначити і деякі недоліки, які має ця програма:

- середня залученість свідчить про те, що утримати інтерес користувачів до застосунку протягом тривалого часу може бути складно;

- багато ветеринарних клінік не бажають долучатися до подібних ініціатив, тому є проблема знайти лікарів.

Автори Anisa Rahmi, I Nyoman Piarsa та Putu Wira Buana у своєму дослідженні описують проблему складності та незручності традиційного процесу отримання медичних послуг [5]. Для вирішення цих проблем вони розробили інтерактивний застосунок FinDoctor для Android з інтеграцією географічної інформаційної системи (GIS), Firebase та Google Maps API.

Основною проблемою, яку автори описують у цій статті, є складнощі при реєстрації та отриманні медичної допомоги. Тривалі черги та труднощі з пошуком лікаря з потрібною спеціалізацією, доступного у певному місці – усе це створює максимальний життєвий дискомфорт. Лікарі також мають проблеми з керуванням чергами та зберіганням медичної документації у паперовому вигляді, що займає багато місця та ускладнює доступ до медичних записів пацієнтів.

Автори розглядали рішення цієї проблеми багатьма способами, які могли б покращити досвід людей у користуванні послугами охорони здоров'я [5]. Зі статті було виокремлено два корисних рішення, які можуть бути актуальними для цієї роботи:

1. Firebase для зберігання даних – використання **Firebase** для зберігання чату, медичних записів та надсилання push-сповіщень у реальному часі. Це

забезпечує простий і швидкий доступ до даних без потреби у складній серверній інфраструктурі.

2. Google Maps API для інтеграції GIS – система допомагає пацієнтам знайти найближчу клініку, що дозволяє оптимізувати вибір медичного закладу за місцем розташування та спеціалізацією.

На основі запропонованих рішень слід виокремити дві головні переваги:

- забезпечує зручний доступ до медичних записів та анамнезу в цифровому форматі, що гарантує швидкий доступ для лікарів і пацієнтів;

- геолокаційна функція допомагає пацієнтам легко знайти найближчу клініку, що підвищує комфорт.

Хоч і може здатися, що такі рішення не повинні мати недоліків, вони їх мають:

- застосунок залежить від доступу до Інтернету та GPS, що може бути проблематичним у місцях з обмеженим покриттям;

- потреба у цифровій грамотності пацієнтів, передусім людей старшого віку, які можуть мати проблеми з користуванням застосунком.

Дослідники Daniel Matisoff та Douglas Noonan описують проблему управління громадськими собачими парками в міському середовищі [6]. Вони зосереджуються на проблемах, пов'язаних із підтриманням безпеки і порядку для відвідувачів цих парків, та пропонують рішення, що ґрунтуються на принципах спільного управління ресурсами.

На сьогодні в собачих парках є чимало проблем, про які не всі говорять і які загалом не планують вирішувати. До таких проблем належать, наприклад:

- недостатнє дотримання правил – багато парків з різних причин не контролюються, тому їх відвідувачі можуть поводитися на власний розсуд;

- конфлікти між користувачами та проблеми з агресивністю собак, які можуть знижувати безпеку;

- брак фінансування та підтримки для утримання парку, оскільки такі парки зазвичай залежать від добровільних внесків і волонтерської роботи.

Існує безліч інших проблем, але з цього дослідження було виокремлено ті, які слід вирішувати в першу чергу. Ці проблеми потрібно розв'язувати, оскільки за статистикою такі парки однаково популярні серед усіх людей, незалежно від віку, статі чи інших факторів.

1.2 Категорії людей, які відвідують собачі парки

Категорії	Середнє	Стандартне відхилення	Граничні переписні округи
Жінки	0.53	0.50	0.41
Вік	31.24	9.90	36.00
Шлюб	0.48	0.50	0.34
Мають дітей	0.15	0.34	0.08
Дохід домогосподарства	76.180$	51.883$	68.870$

(Джерело: власна обробка на основі [6])

Автори пропонують логічні та прості рішення, які можуть радикально змінити підхід і думку людей про цей тип парку [6]:

- залучення користувачів собачого парку до управління парком – пропонується, аби відвідувачі активно долучалися до створення правил та самоорганізації при утриманні парку;

- застосування соціальних норм та самоконтролю – замість формального контролю користувачі заохочуються до дотримання неписаних норм, включно з притягненням до відповідальності за порушення правил;

- створення малозатратних механізмів для розв'язання конфліктів – автори рекомендують створити прості механізми для розв'язання конфліктів між користувачами без залучення сторонньої охорони чи дорогих ресурсів [6].

Як і в кожному дослідженні, важливо завжди підкреслювати успіхи та невдачі рішень, запропонованих авторами. Переваги цих рішень будуть такими:

- активна участь користувачів у створенні зручності та порядку у собачих парках;

- механізми самоконтролю є дешевими і за сприятливого соціального середовища можуть бути ефективними;

- добровільні внески звичайних відвідувачів парків можуть бути значною підтримкою для волонтерів та організацій зі стабільним фінансуванням парків.

Однак такі рішення мають і деякі недоліки:

- відсутність формального контролю може призводити до конфліктів та порушень правил, особливо у випадку агресивної поведінки собак;

- залежність від добровільної підтримки не може гарантувати потрібний рівень утримання та збереження парку.

Автор Claire Micklin розглядає проблему безпритульних собак у містах і пропонує рішення у вигляді застосунку DogTracker, який допомагає громадянам та органам влади відстежувати місцезнаходження і поведінку безпритульних собак [7].

Основна проблема, яку автор описує у цьому дослідженні, безпосередньо пов'язана з великою кількістю безпритульних собак у багатьох країнах та містах, особливо в менш розвинених регіонах [7]. Безпритульні собаки можуть становити значну загрозу для людей і сприяти поширенню захворювань, як-от сказ. Сьогодні існує безліч різних способів вирішення цих проблем, але не всі вони хороші, а іноді є досить жорстокими. Тому автор пропонує власне рішення, краще за інші.

У цій статті автор пропонує одне велике рішення – мобільний застосунок – а також містить кілька додаткових рішень:

1. Мобільний застосунок для відстеження собак DogTracker дозволяє користувачам фотографувати безпритульних собак, додавати теги за географічним розташуванням, описувати поведінку собаки та отримувати інформацію про попередні спостереження в певному місці.

2. Концепція краудсорсингу даних – люди можуть повідомляти про великі скупчення собак, що допоможе іншим уникнути небезпечних районів та допоможе зібрати актуальну інформацію для служб контролю за тваринами.

3. Запуск інформаційних кампаній – мета застосунку також полягає у підвищенні обізнаності щодо важливості стерилізації та вакцинації домашніх тварин, а також відповідальності за поведінку собак.

Ці рішення мають такі переваги:

- збір актуальних даних про популяцію безпритульних собак для ефективнішого планування програм контролю чисельності та вакцинації;

- залучення громадян до моніторингу популяції та підвищення обізнаності щодо відповідального ставлення до тварин;

- можливість уникати небезпечних районів з агресивними собаками.

Однак навіть при таких рішеннях існують певні недоліки, які слід враховувати:

- існує ризик того, що застосунок можуть використовувати недобросовісні користувачі для нападу або знищення собак;

- цей тип застосунку потребує постійного доступу до Інтернету для надання надійних даних про місцеположення.

У своєму дослідженні автори Aatish Neupane, Derek Hansen, Anud Sharma, Jerry Alan Fails, Bikalpa Neupane та Jeremy Beutler описують ефективність використання ігрових елементів у застосунках для відстеження фізичної активності з метою мотивування користувачів до підвищення фізичної активності, особливо ходьби [8]. Стаття також аналізує інші застосунки з елементами гейміфікації, виявляє їхні переваги та недоліки і пропонує рішення для розробки таких застосунків.

Основна проблема, яку розглядає ця стаття, – це недостатня мотивація користувачів продовжувати активний спосіб життя, а отже, небажання тривало використовувати застосунки для відстеження фізичної активності [8]. Більшість людей швидко втрачає інтерес до таких застосунків через брак

мотивації або розваг. Автори пропонують безліч різних рішень, які мають мотивувати користувачів до якомога довшого використання таких застосунків та принести користь як розробникам, так і звичайним користувачам:

1. Елементи змагання – у застосунках цього типу часто використовуються соціальні впливи, як-от таблиці лідерів, які стимулюють користувачів залишатися активними завдяки змаганню з іншими.

2. Реальні нагороди – деякі застосунки пропонують можливість отримання реальних нагород, які користувач може обміняти, наприклад, на місячну підписку чи на спортивний одяг з партнерського магазину.

3. Цілі – у цьому типі застосунків цілі є дуже поширеними, наприклад щоденні цілі щодо кількості кроків, які людина має зробити, що дозволяє користувачам зберігати інтерес до своїх маленьких досягнень.

4. Storylines – запровадження сюжетної лінії надає користувачам цікаву історію, яка спонукає їх виконувати фізичні завдання, аби просуватися в грі.

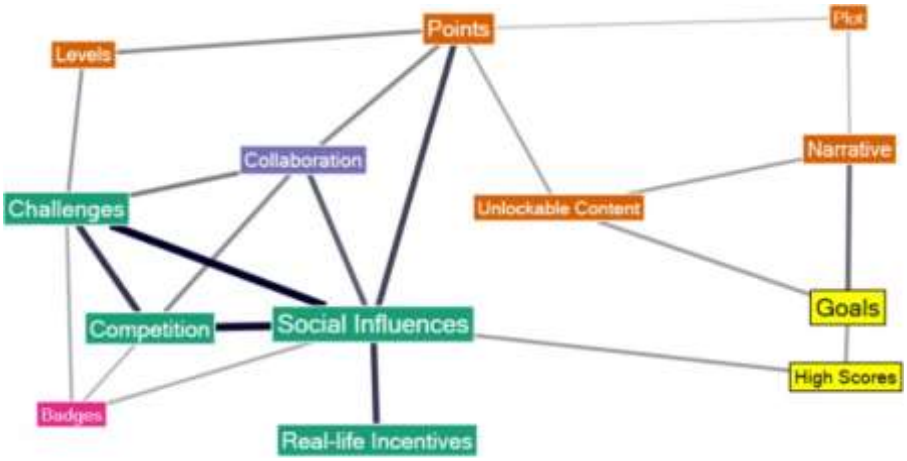


Рис. 1.2 Приклад мережевої моделі з використанням storylines та гейміфікації у фітнес-застосунку [8]

Усі такі рішення потребують збору значної кількості даних. Тому ці застосунки мають доволі схожу структуру збору даних, як, наприклад, застосунок з п'ятого дослідження FinDoctor, який використовує Firebase для зберігання даних та моніторингу активності користувачів у реальному часі,

аналогічно до того, як фітнес-застосунки відстежують фізичну активність користувачів [5].

Вибір такого типу рішення має кілька переваг:

- реальні нагороди є потужною мотивацією для користувачів, оскільки пропонують відчутні переваги за досягнення активності;

- сюжетні елементи надають користувачам новий досвід, завдяки якому фізична активність стає цікавішою;

- таблиці лідерів допомагають привертати увагу та зацікавлення користувачів через змагання з іншими.

Однак існує і кілька важливих недоліків:

- постійне порівняння з іншими може негативно впливати на самооцінку деяких користувачів;

- реальні нагороди можуть бути дорогими для розробників та знижувати прибутковість застосунку;

- висока складність або погано спроєктовані ігрові елементи можуть знижувати інтерес користувачів.

Аналіз сучасних підходів та технологій демонструє значний потенціал цифрових інструментів у розв'язанні проблем догляду за тваринами, надання медичних послуг та поліпшення міського середовища. Використання мобільних застосунків, інтеграція геолокаційних функцій та інноваційні підходи до управління громадськими ресурсами сприяють поліпшенню користувацького досвіду, залученню громад та ефективному вирішенню глобальних проблем, як-от контроль безпритульних тварин та запобігання захворюванням.

Технології забезпечують швидкий доступ до інформації, полегшують комунікацію між користувачами та мотивують до тривалої залученості за допомогою гейміфікації.

1.3 Аналіз мультиплатформних мов програмування

У сучасній розробці мобільного програмного забезпечення значної популярності набувають мультиплатформні технології, які дозволяють створювати застосунки одночасно для операційних систем Android та iOS. Такий підхід дає можливість скоротити витрати часу та ресурсів на розробку, а також спростити підтримку програмного продукту. Науковці активно досліджують особливості використання мультиплатформних мов програмування та фреймворків, аналізують їхню продуктивність, функціональні можливості, переваги й недоліки у порівнянні з нативною розробкою. У цій частині теоретичного аналізу варто звернути увагу на мови програмування, які дозволяють заздалегідь написати застосунок як для платформ Android, так і для iOS. Кожна з цих мов має свої плюси й мінуси, тому потрібно зробити висновки про те, яка мова є більш придатною для написання програми. У процесі аналізу нами було виділено декілька рішень.

1. Перше рішення.

Автори Alexandru Tabusca, Cristina Coculescu та Mironela Pirnau у своїй статті розглядають проблему великого часу та ресурсів, що витрачаються на розробку мобільних застосунків для платформ Android та iOS [10]. Це зумовлено потребою створювати дві окремі команди розробників і координувати їхню роботу для збереження загального дизайну та функціональності застосунків, оскільки для написання застосунку для двох платформ одночасно необхідно писати двома різними мовами програмування.

Основна проблема полягає в тому, що для написання застосунку для платформи Android та для платформи iOS довелося б використовувати дві різні мови програмування, кожна з яких оптимізована для конкретної платформи. Таке рішення потребувало б, аби застосунок одночасно писали дві різні команди програмістів. Цей фактор тягне за собою високі часові та фінансові витрати, а також потребу в більшій кількості людей на проєкті. Тому

автори пропонують одне досить популярне нині рішення цієї проблеми – використання мови програмування Flutter, яка дозволяє створювати застосунки для Android та iOS одночасно. А також кілька інших цікавих рішень [10].

Головним рішенням авторів є використання мови програмування Flutter, але це рішення містить і кілька інших цікавих:

1. Використання мови Dart. Мова програмування Dart забезпечує високу продуктивність, типову безпеку та швидке виконання.

2. Використання віджетів. Flutter надає набір віджетів для створення адаптивного дизайну. Вони працюють незалежно від нативних елементів операційної системи та забезпечують уніфікований інтерфейс.

3. Інтеграція зі службою Firebase. Просте використання баз даних у реальному часі за допомогою Firebase (переваги використання Firebase вже описано у статті про FinDoctor) [5].

Ці рішення мають такі переваги:

- ефективність розробки та правильне використання ресурсів;

- гнучкість дизайну – використання віджетів дозволяє створити однаковий інтерфейс на різних платформах;

- можливість інтеграції інструментів Google, як-от Firebase, Skia;

- графічна бібліотека Skia забезпечує достатньо швидке відтворення.

Однак попри всі переваги існує також кілька недоліків:

- великий розмір застосунків, створених мовою програмування Flutter;

- брак нативних компонентів – деяким розробникам може знадобитися доступ до нативних функцій операційної системи, які Flutter напряму не підтримує.

2. Друге рішеня.

Першим рішенням цієї проблеми є використання інструменту React Native, який призначений для написання мультиплатформних застосунків.

Існує доволі багато різноманітної літератури та статей, які описують, як добре використовувати React Native. Найкращий спосіб його використання, проте, описують Vipul Kaushik, Kamali Gupta та Deepali Gupta [11]. Автори пропонують використовувати React Native як універсальний інструмент, який дозволяє писати один код JavaScript для обох платформ. У статті також представлено мобільний застосунок Medikyte для замовлення ліків, який демонструє можливості цього фреймворка.

Це рішення дозволяє програмістам припинити використовувати окремі мови програмування для написання застосунків для Android та iOS. А якщо говорити про бібліотеки чи фреймворки, ті, що вже існували, не забезпечували однакового користувацького досвіду на всіх платформах.

Тому автори на прикладі застосунку, розробленого для онлайн-замовлення ліків, демонструють усі функції, переваги та недоліки React Native [11]. Гідними уваги є такі переваги:

1. Економія часу та ресурсів. Цей фактор є ключовим, оскільки впливає на якість програми та її майбутнє.

2. Гібридна розробка. React Native забезпечує майже нативну продуктивність.

3. Зручність для розробників. React Native має великий набір готових модулів для інтеграції, наприклад Firebase для backend та Google Places для геолокації.

Якщо порівнювати переваги з мовою Flutter, автори виявили, що головною перевагою цієї мови програмування є використання функцій, які надає безпосередньо Google [10]. І тут також можна побачити щось подібне. Тож знову підтверджується висока ефективність використання React Native.

Однак не варто забувати про недоліки:

1. Залежність від зовнішніх бібліотек. Велика кількість залежностей ускладнює керування проєктом і може впливати на продуктивність.

2. Продуктивність. У випадку складних застосунків продуктивність може бути нижчою, ніж у нативних рішень.

Недоліків мало, але вони є, і якщо знову взяти Flutter, він менш залежний від бібліотек третіх сторін, тому за цією логікою має використовуватися зручніше [10].

Однак на завершення автори на прикладі застосунку **Medikyte** показали, що цей застосунок успішно вирішує поставлені завдання, надаючи зручний для користувача інтерфейс та функції для замовлення ліків [11].

React Native є ефективним фреймворком для створення гібридних застосунків з мінімізацією зусиль, потрібних для мультиплатформної розробки, хоча в деяких випадках нативні рішення можуть бути більш доцільними.

3. Третє рішення.

Вирішенням проблеми дорогої розробки мобільних застосунків як для Android, так і для iOS є використання платформи Xamarin, яка також призначена для написання для двох різних мобільних платформ одночасно.

Це рішення є менш популярним, ніж згадані вище рішення, але автор Gokul Ramadoss у своїй статті дуже добре описав, чому варто використовувати Xamarin на основі мови C#, його переваги та навіть досить добре порівняв його з попередніми рішеннями [12].

Автор пропонує Xamarin як один з варіантів для розробки мультиплатформних застосунків і порівнює його з React Native та Flutter [10] [11]. Xamarin використовує мови C# та .Net і дозволяє створювати застосунки для iOS, Android та Windows за допомогою єдиної бази коду.

Використання цієї конкретної техніки програмування має кілька гарних переваг:

- до 90% коду можна повторно використовувати;
- нативна продуктивність завдяки бінарній компіляції;
- сильна підтримка з боку компанії Microsoft.

Однак використання цієї платформи має і деякі недоліки: великі розміри застосунків; високі початкові витрати для великих підприємств (через ліцензії на Visual Studio тощо).

У цій статті автор також дуже цікаво описує відмінності між згаданими вище техніками – йдеться про Flutter та React Native [10] [11]. Головних відмінностей доволі багато, але декілька основних варто згадати:

1.3. Порівняння Flutter, React Native та Xamarin

Критерій	Flutter	React Native	Xamarin
Мова програмування	Dart (оптимізована для мобільних застосунків).	JavaScript (гнучка, популярна при розробці вебсайтів).	C# (статично типізована, інтегрована з .NET).
Можливості UI	Потужні віджети з можливістю налаштування, філософія «все – віджет».	Потребує написання нативних компонентів для складного інтерфейсу.	Xamarin.Forms дозволяє створити загальний інтерфейс, але кастомізація гірша, ніж у Flutter.
Продуктивність	Сильна візуалізація завдяки графічному рушію Skia.	Може мати затримку через JavaScript-міст.	Нативна продуктивність завдяки компіляції в бінарний код, доступ до нативних API.
Витрати на ліцензії	Безкоштовно	Безкоштовно	Потребує Visual Studio (для малих компаній безкоштовно, для великих платно).

(Джерело: власна обробка на основі [12])

Автор дійшов висновку, що кожна з запропонованих платформ має свої переваги та недоліки, і використовувати їх слід для різних цілей, оскільки між

ними існують значні відмінності. Якщо говорити саме про Xamarin, він підходить для проєктів, де важлива інтеграція з .NET, високий рівень коду та ефективність.

4. Четверте рішення.

Ще одним важливим рішенням цієї проблеми є використання інструменту, такого як Ionic. Якщо порівнювати його з популярнішими бібліотеками чи мовами програмування, він не такий популярний, але має свої сильні сторони.

Авторка Priyanka Chaudhary говорить про ефективність використання цього фреймворка та його особливості [13].

Авторка пропонує використовувати Ionic Framework як гібридний підхід до розробки мобільних застосунків. Цей фреймворк дозволяє розробляти застосунки за допомогою стандартних веб-технологій (HTML, CSS, JavaScript) і запускати їх на різних мобільних платформах завдяки Apache Cordova.

До головних функцій застосунку Ionic належать мультиплатформні можливості та плагіни Cordova, які забезпечують доступ до нативних функцій пристрою.

Використання цієї платформи має такі переваги:

- економія ресурсів – одна база коду для всіх платформ;

- просте використання знайомих веб-технологій, як-от HTML та CSS;

- підтримка плагінів – використання нативних функцій через інтерфейс API;

- доступність готових шаблонів, що забезпечує швидкий старт та швидке навчання.

При виборі Ionic слід також звертати увагу на його недоліки:

- залежність від WebView. Через цей недолік продуктивність може бути нижчою, ніж у нативних застосунках;

- у випадку складних інтерфейсів важче досягти високої графічної продуктивності через обмежені можливості налаштування;

- цей застосунок може бути доволі великим через включення додаткових залежностей.

Попри всі недоліки використання, авторка дійшла висновку, що Ionic Framework є ефективним інструментом для створення гібридних застосунків, що особливо важливо для малих та середніх проєктів, які потрібно швидко запустити на кількох платформах.

Він дозволяє створювати застосунки за допомогою єдиної бази коду, що значно скорочує час розробки, і, не в останню чергу, Ionic завдяки своїй open-source природі та підтримці веб-технологій є доступним за ціною та простим у використанні порівняно, наприклад, з React Native [11].

На основі поданих аналізів досліджень можна зробити кілька цікавих висновків. Ці висновки покажуть, як можна ефективно використовувати мультиплатформні мови програмування для написання програм:

Перший висновок, який можна зробити, – це який інструмент краще використовувати для написання застосунку як для Android, так і для iOS. У цьому пункті потрібно врахувати аналіз мови програмування Flutter та аналіз React Native [10] [11]. Ці платформи в порівнянні з іншими продемонстрували всі переваги, необхідні для зручної розробки застосунку, та мають у своєму розпорядженні все – різний доступ до функцій, дизайн або бази даних.

Ще одним висновком, який потрібно розглянути, є питання найпростішого способу написання мультиплатформного застосунку. Тут відповідь стовідсоткова – використовуйте Ionic. Як показав аналіз цієї платформи, вона досить проста для написання коду, оскільки ґрунтується на використанні інтернет-технологій та використанні нативних функцій, які містить API [13]. Однак все ж варто зазначити, що Ionic не є найкращим варіантом для написання застосунку, оскільки попри простоту використання він має обмежені функції порівняно з конкурентами.

Не в останню чергу варто також зробити кілька висновків з порівняння Flutter та Xamarin. Автор досить добре та наочно описав переваги та відмінності між цими різними інструментами для написання мультиплатформних застосунків. Головна відмінність полягає в тому, що

Xamarin.Forms дозволяє створити загальний інтерфейс, але кастомізація гірша, ніж у Flutter [12].

На основі цих висновків можна стверджувати, що ідеальної мови програмування, платформи чи бібліотеки для написання мультиплатформних застосунків не існує – кожна з них має свої переваги та недоліки, які зрештою повертають програмістів до використання окремих мов програмування для написання окремого коду для Android та iOS, але якщо виходити з висновків цих аналізів, варто згадати про використання Flutter, оскільки він є найближчим за функціональністю до умов, коли програмісти використовували окремі платформи [10].

1.4 Аналіз мов програмування для однієї платформи

У процесі розробки мобільних застосунків важливе місце займають нативні мови програмування, які створені спеціально для певної операційної системи. Такі технології забезпечують високу продуктивність застосунків, стабільність роботи та повний доступ до функціональних можливостей платформи. Науковці та практики у сфері програмної інженерії активно досліджують особливості використання нативних мов програмування для Android та iOS, аналізують їхню ефективність, переваги й недоліки у порівнянні з мультиплатформними рішеннями. У цьому підрозділі розглянуто сучасні мови програмування для окремих мобільних платформ, зокрема Kotlin для Android та Swift для iOS, а також особливості їх застосування під час створення мобільних застосунків. Зокрема, опишемо Android Studio (Kotlin) та мову програмування для iOS – Swift.

Kotlin – це мова програмування, що працює на віртуальній машині Java (JVM) і поєднує об'єктно-орієнтовані та функціональні можливості. Вона повністю сумісна з мовою Java, що дозволяє використовувати Kotlin в існуючих Java-проєктах або писати нові застосунки з нуля [21].

Компанія Google оголосила Kotlin офіційною мовою для розробки застосунків для Android у 2017 році, а з 2019 року рекомендує її як основну мову для застосунків для Android.

Kotlin використовується у таких сферах:

- розробка для Android – основна мова для мобільних застосунків для Android;

- розробка серверної частини – використовується при розробці бекенду (наприклад, з Ktor, Spring Boot);

- розробка між платформами – завдяки Kotlin Multiplatform її можна використовувати для застосунків iOS та веб-застосунків;

- розробка десктопних застосунків – підтримуються JavaFX та TornadoFX;

- скриптування – у деяких сценаріях використовується як заміна Bash або Python.

На основі цього функціоналу автори Bruno Gois Mateus та Matias Martinez виокремлюють кілька важливих переваг, якими можна керуватися при виборі мови програмування для написання програм [21]:

- чиста та лаконічна синтаксис – менше коду, ніж у Java;

- повна сумісність з Java – можна використовувати код та бібліотеки Java;

- безпека від NullPointerException (NPE) – вбудована система null-safety;

- продуктивність – можливість використовувати inline-функції, що зменшують витрати на пам'ять;

- підтримка застосунків Google та Android Studio – інтеграція на рівні середовища розробки.

Вони також звернули увагу на кілька важливих недоліків, які можуть вплинути на рішення про розробку мобільного застосунку:

- вища складність компіляції – компіляція триває довше, ніж у Java;

- обмежена документація – менше матеріалів, ніж для Java;

- потребує адаптації від розробників – якщо команда працювала тільки з Java, доведеться вчитися;

- складнощі при налагодженні – деякі конструкції важко аналізувати через вбудовану оптимізацію.

Проаналізувавши всю інформацію, автори дійшли висновку, що Kotlin є потужною мовою, яка значно покращує розробку застосунків для Android, роблячи код чистішим, безпечнішим і продуктивнішим [21]. Завдяки офіційній підтримці компанії Google вона поступово витісняє Java з екосистеми Android.

Swift – це мова програмування, розроблена компанією Apple у 2014 році як альтернатива Objective-C для розробки застосунків для iOS та macOS. Swift має значно кращу продуктивність, безпеку коду та читабельність порівняно з Objective-C [22].

Swift є компільованою мовою, що генерує машинний код (на відміну від Java або Python, які використовують байт-код). Він також підтримує динамічний аналіз коду, що дозволяє досліджувати його поведінку на рівні виконання [22].

Swift використовується у таких сферах:

- розробка для iOS – основна мова для створення застосунків для iPhone та iPad;

- розробка для macOS – використовується для створення застосунків для MacBook та iMac;

- watchOS та tvOS – використовується для розробки застосунків для Apple Watch та Apple TV;

- розробка серверної частини – може використовуватися на бекенді завдяки Swift on Server (наприклад, Vapor);

- розробка машинного навчання – Swift у компанії Apple використовується для створення моделей штучного інтелекту.

Більшість мобільних застосунків в App Store написана мовою Swift, і її популярність з 2014 року швидко зростає.

Що стосується переваг та недоліків цієї мови програмування, автори Malte Kraus та Vincent Haupert насамперед виокремили такі переваги [22]:

- висока продуктивність – швидший за Objective-C завдяки ефективній компіляції в машинний код;

- безпека коду – Swift має сувору null safety (не дозволяє помилок з нульовими значеннями);

- стислий синтаксис – для виконання тих самих завдань, що й Objective-C, потребує менше коду;

- сумісність з Objective-C – зі Swift можна використовувати старі бібліотеки та код;

- підтримка сучасних функцій – lambda-функції, coroutines, closures та динамічна типізація;

- активна підтримка компанії Apple – компанія Apple повністю підтримує Swift і активно його розвиває.

І, звичайно, потрібно звернути увагу на кілька основних недоліків, які зазвичай виникають у процесі розробки програми:

- залежність від компанії Apple – основне використання лише в екосистемі Apple, що обмежує його можливості для крос-платформної розробки;

- проблеми з сумісністю версій – оновлення Swift іноді спричиняють проблеми у старіших проєктах;

- обмежена підтримка серверної розробки – хоча Swift можна використовувати на сервері, він менш популярний, ніж Python чи JavaScript;

- менше ресурсів для навчання – у порівнянні з популярними мовами, як-от Java або Python, існує менше курсів та навчальних матеріалів.

На завершення можна сказати, що Swift – це сучасна, швидка та безпечна мова, ідеальна для розробки застосунків для iOS/macOS [22]. Вона замінила мову Objective-C завдяки простішому синтаксису та вищій продуктивності. Однак Swift значною мірою залежить від екосистеми Apple, що може бути недоліком для тих, хто хоче розробляти мультиплатформні рішення.

Якщо зробити стислий висновок щодо цих мов програмування, можна виокремити одну головну рису – кожна з цих мов була створена для своїх

цілей. Наприклад, Kotlin створений для написання застосунків виключно для платформи Android, тому в цій сфері має свої особисті переваги та недоліки [21]. Якщо говорити про Swift, ця мова програмування використовується виключно для написання застосунків для iOS, тому було б недоцільно порівнювати її з мовами, створеними для платформи Android, оскільки вона має свою специфіку у цій сфері [21].

Однак загалом можна скласти невеличку таблицю, яка покаже основні відмінності між цими двома мовами програмування:

1.4. Порівняння мов Kotlin та Swift

Параметр	Kotlin	Swift
Основне використання	Розробка для Android, серверів, мультиплатформна	iOS/macOS/watchOS/tvOS
Підтримка кількох платформ	Так (Kotlin Multiplatform)	Обмежена (Swift для сервера)
Синтаксис	Стислий, схожий на Java та Python	Мінімалістичний, схожий на Python
Продуктивність	Висока, але залежить від JVM	Висока, скомпільована в машинний код
Управління пам'яттю	Автоматичний збирач сміття	ARC (автоматичний підрахунок посилань)

(Джерело: власна обробка на основі [21] [22])

У цій порівняльній таблиці виокремлено основні компоненти, за допомогою яких можна було порівняти дві мови програмування, створені для різних платформ.

РОЗДІЛ 2 ТЕОРЕТИЧНІ ЗАСАДИ РОЗРОБКИ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1 Фаза концепції мобільного застосунку

У цьому розділі розглянуто теоретичні засади та технологічні підходи, які були сформовані на основі аналізу наукових джерел і використані під час розробки мобільного застосунку. Проведений аналіз сучасних досліджень у сфері мобільної розробки, тестування програмного забезпечення, використання цифрових технологій та мов програмування дозволив визначити найбільш ефективні інструменти, методи й концепції для створення програмного продукту. Особливу увагу приділено етапам проєктування мобільного застосунку, вибору технологій розробки, організації взаємодії користувача із системою та забезпеченню якісної роботи застосунку. Для кращого розуміння процесу створення програмного забезпечення теоретичний матеріал структуровано відповідно до основних фаз розробки мобільних застосунків.

На цьому етапі необхідно виходити з двох доволі популярних концепцій – життєвого циклу мобільного застосунку та варіантів використання.

Ці два поняття використовуються для розуміння того, що буде робити застосунок і як ним правильно користуватиметься користувач чи розробник. Як описує автор Tejas Vithani у своїй статті, процес життєвого циклу розробки програмного забезпечення є своєрідною структурою, що використовується при розробці будь-якого програмного продукту [14]. Якщо говорити про варіанти використання, це просто сценарії, які покажуть, як користувач, наприклад, взаємодіє із сервером тощо. Це доволі добре описано в аналізі третьої статті [3].

До цього пункту також доцільно додати таке поняття, як краудсорсинг даних. Ця фаза концепції мобільного застосунку, як вже описано автором в аналізі сьомої статті, могла б бути доволі корисною при зборі необхідної

інформації від користувачів, їхніх думок та зауважень, і на основі цього розробити варіанти використання застосунку [7].

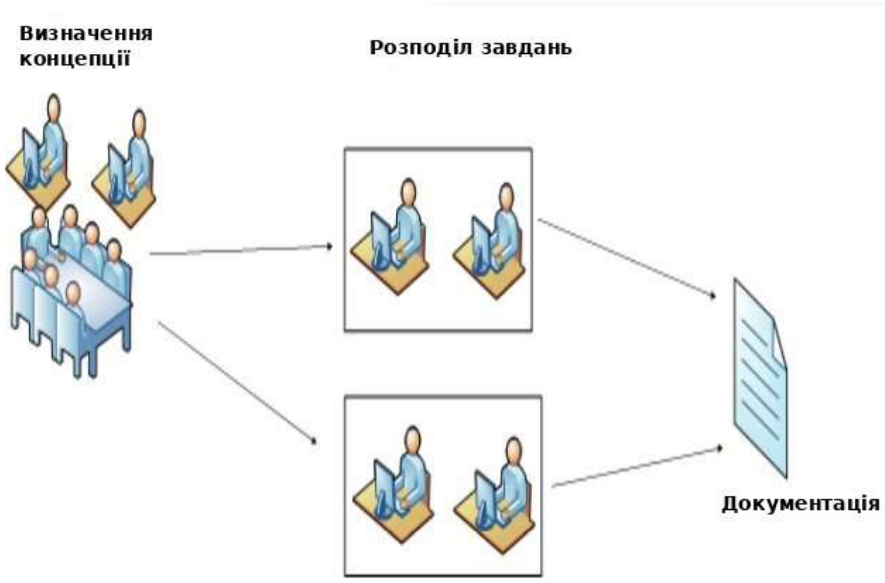


Рис. 2.1. Приклад розробки ідеї [14]

На основі цієї теорії можна стверджувати, що вона є надзвичайно важливою для розробки будь-якого мобільного застосунку чи будь-якого продукту загалом, оскільки вона дозволяє розробникам зрозуміти основну концепцію продукту та процеси, які відбуватимуться під час його створення.

2.2 Фаза технічного планування

Ще однією фазою після концепції є фаза технічного планування. Тут дуже актуальним є вибір технологічного стеку, який використовуватиметься для різних цілей – написання коду, дизайн тощо.

На цьому етапі потрібно пройтися по поняттях, пов'язаних з повторним використанням коду – поняттях байт-кода та успадкування. Автори першої статті досить добре описав, як можна якісно та грамотно використовувати ці два поняття так, аби вони не заважали функціональності застосунку [1]. Однак якщо доводиться обирати, краще використовувати принцип успадкування, який є принципом об'єктно-орієнтованого програмування, що дозволяє

створювати нові класи на основі існуючих та успадковувати їх властивості. Це може стати великою допомогою при написанні розгалуженого коду та зекономить розробникам багато часу.

Наступна фаза планування може містити інструменти, які зазвичай використовуються в сучасному світі розробки мобільних застосунків – інструменти платформи Google.

Як описують автори в аналізі п'ятої статті, Firebase – це хмарна платформа, створена компанією Google, яка пропонує широкий спектр інструментів для розробки мобільних та веб-застосунків [5]. У цьому випадку нас цікавлять можливості зберігання та синхронізації даних, аби можна було зберігати дані в реальному часі та синхронізувати їх між усіма пристроями користувачів.

Ще одним досить важливим аспектом є планування того, які API будуть використовуватися розробниками при написанні застосунку. Їх доволі багато, і вони також створені для різних цілей та напрямків [5]. У цьому аналізі нас цікавлять API, пов'язані з відстеженням активностей чи плануванням маршруту.

Автор Hao Zhong у своїй статті на цю тему дуже добре описав, що таке API та які його переваги [15]. Інтерфейс API (Application Programming Interface) визначається як набір видимих елементів коду, які надають фреймворки чи бібліотеки. Ці елементи забезпечують інтерфейс для взаємодії між різними частинами програмного забезпечення. За допомогою API можна повторно використовувати код, а також забезпечити певну масштабованість проєкту, що було б неможливо без використання цього інструменту.

Ця фаза також могла б охоплювати аналіз мультиплатформних технологій, але це буде описано пізніше.

2.3 Фаза розробки та реалізації

На цій фазі основний простір буде надано теорії, яка б допомогла написати програму та реалізувати її для подальшого тестування. Можна сказати, що це один з найважливіших процесів, який є ресурсомістким.

Якщо виходити з попереднього огляду статей та досліджувати інструменти, придатні для розробки мультиплатформного застосунку, важливими можна вважати знання, наведені у двадцять першій статті, що описує використання мови програмування Kotlin, а також її сучасних інструментів, які можуть значно сприяти ефективному проєктуванню та розробці мобільного застосунку [21].

Kotlin сьогодні вважається однією з основних мов для розробки на платформі Android. Її перевагою є зрозумілий синтаксис, інтероперабельність з Java та підтримка сучасних інструментів розробки, як-от Jetpack Compose. Jetpack Compose являє собою декларативний підхід до створення користувацького інтерфейсу, що дозволяє швидше та гнучкіше створювати UI-компоненти. Хоча сам Kotlin не є безпосередньо мультиплатформною мовою, у межах технології Kotlin Multiplatform можна ділитися логікою між різними платформами (наприклад, Android та iOS), що створює можливість розробки мультиплатформних рішень [21].

На додаток можна згадати і реалізацію компоненту WebView та інструменту Apache Cordova, які добре описано у статті авторки Priyanka Chaudhary [13]. Чому ці інструменти були створені?

WebView являє собою компонент для відображення вебвмісту в межах нативного застосунку, що часто використовується, наприклад, для відображення сторінок з умовами використання чи форм зворотного зв'язку [13]. Apache Cordova, навпаки, є комплексним фреймворком, який дозволяє створювати мультиплатформні застосунки на основі веб-технологій та

WebView [13]. За допомогою плагінів він забезпечує доступ до нативних функцій пристрою без потреби писати нативний код.

Розгляду заслуговує також реалізація ігрових функцій у мобільних застосунках. Використання таких функцій допомагає застосунку отримати більшу популярність та запобігає тому, аби люди видаляли застосунок після кількох днів використання. За дослідженням авторів восьмої статті, варто використовувати елементи гейміфікації та storyline [8]. Ці два поняття існують разом, оскільки саме їх поєднання створює найбільшу зацікавленість користувачів. Їхньою метою є реалізація елементів ігрового дизайну для покращення користувацького досвіду загалом, а також для максимального збільшення масштабу застосунку.

2.4 Фаза тестування та оптимізації

Останньою фазою перед тим, як продукт отримає кінцевий користувач, є фаза тестування та оптимізації. Для цієї фази існує безліч різних способів тестування застосунків з різних боків, але необхідно звернутися до концепції Assertion Density [16].

Assertion density визначається як кількість тверджень у тестовому коді на тисячу рядків коду (KLOC). Твердження – це перевірки, які підтверджують правильність програмного коду під час виконання. Чим більше тверджень містить тест, тим вища ймовірність того, що він зможе виявити помилки [16].

Що дає таке тестування продукту загалом? По-перше, воно зменшує кількість помилок у продакшн-коді, оскільки збільшення показника тверджень пов'язане зі зменшенням кількості помилок. Автори також вказують, що досвід команди розробників безпосередньо впливає на щільність асерцій [16]. Більш досвідчені розробники зазвичай додають більше асерцій, завдяки чому їхні тести є ефективнішими. Однак найважливішою метою є забезпечення стабільності коду завдяки великій кількості тверджень.

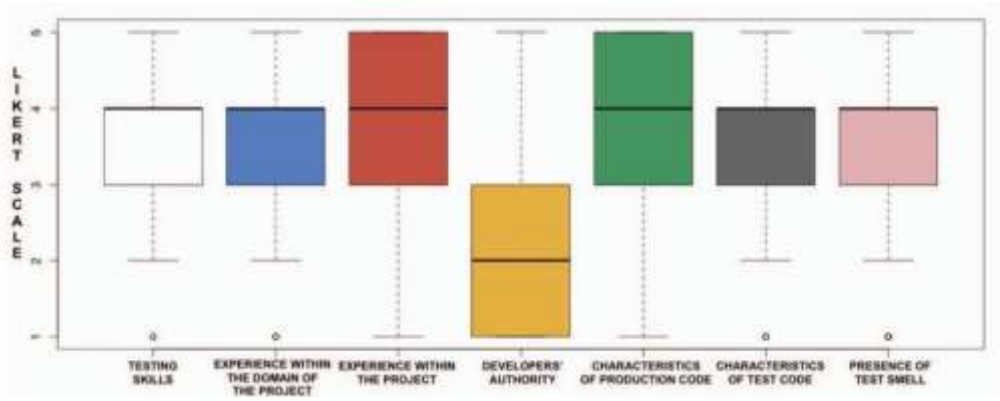


Рис. 2.2. Результати щодо релевантних аспектів для написання асерцій [16]

У цьому теоретичному аналізі представлено найважливіші інструменти та приклади, які могли б допомогти не лише ефективно створити повністю функціональний застосунок, але й забезпечити його зручність для користувача за допомогою різних не зовсім стандартних інструментів і, що не менш важливо, аби цей застосунок був максимально надійним та добре протестованим.

РОЗДІЛ 3 ПРОЄКТУВАННЯ ПРАКТИЧНОЇ ЧАСТИНИ

3.1 SWOT-аналіз

У цьому розділі розглянуто практичні аспекти проєктування та розробки мобільного застосунку. На основі проведеного теоретичного аналізу визначено основні методи, інструменти та підходи, необхідні для реалізації програмного продукту. Особливу увагу приділено плануванню функціональних можливостей застосунку, аналізу потреб користувачів, вибору технологій розробки, а також дослідженню факторів, які можуть впливати на ефективність та якість роботи системи. Використання структурованого підходу до проєктування дозволяє підвищити надійність, зручність використання та продуктивність мобільного застосунку.

SWOT-аналіз є фундаментальним інструментом стратегічного планування, який широко використовується у бізнесі та інших сферах. Його основною метою є оцінка внутрішнього та зовнішнього середовища організації для визначення її стратегічної позиції. Аналіз допомагає виявити сильні й слабкі сторони, можливості та загрози, які можуть вплинути на досягнення цілей організації [17].

Використання SWOT-аналізу в сучасному світі є ключовим аспектом при плануванні розробки будь-якого застосунку, але варто зазначити, що цей аналіз застосовується не лише в технічній сфері, але й у таких галузях [17]: використовується в галузі освітніх послуг для оцінки становища університетів, розробки навчальних планів та впровадження інновацій, як-от дистанційне навчання; у сільському господарстві використовується для розробки стратегій розвитку ферм, управління ресурсами та планування нових напрямів бізнесу; широко застосовується також у сфері охорони здоров'я для покращення якості послуг, оптимізації роботи медичних установ та запровадження нових медичних процедур.

У наступній таблиці зведено всі основні аспекти SWOT-аналізу цього типу програми.

3.1. SWOT-аналіз для мобільного застосунку для власників собак

Сильні сторони (Strengths)	Інтеграція з мапами для планування та пошуку маршруту собаки. Унікальна функція чату з власниками собак у вашому регіоні. Можливість монетизації шляхом підключення преміумфункцій. Потенціал збору корисних даних (звички собак, місця). Можливість співпрацювати з ветеринарами при створенні корисних статей чи при консультаціях. Можливості партнерства з аптеками для швидкого пошуку ліків.
Слабкі сторони (Weaknesses)	Недостатня популярність продукту в малих містах, де мало користувачів собак чи собачих парків. Інтеграція всіх функцій може бути технічним викликом і потребує ретельного захисту даних. Залежність продукту від API третіх сторін. Обмежений перелік ключових користувачів.
Можливості (Opportunities)	Запуск спеціальних промоакцій з аптеками (знижки для користувачів). Можливість користувачів оцінювати собачі парки з метою їх покращення. Додавання функції нагадування про щеплення та обробку від паразитів. Можливість мати всю необхідну інформацію про собаку в одному місці (паспорт, сертифікати).
Загрози (Threats)	Складнощі при створенні партнерств з аптеками чи клініками. Ринок застосунків для домашніх тварин дуже конкурентний. Низька середня частота використання застосунку. Згідно з дослідженням [9], багато собак живе в будинках з власним подвір'ям, тому люди можуть не використовувати такий застосунок. Незацікавленість лікарів писати статті для платформи.

Використання SWOT-аналізу має кілька основних переваг: простота та доступність; універсальність для використання у багатьох сферах бізнесу; може використовуватися як самостійний інструмент або у поєднанні з іншими техніками.

Слід також виокремити певні недоліки, які виникають у цьому аналізі: надає лише загальну картину, тому може бути недостатнім для комплексних стратегічних рішень; менш ефективний для середовищ, що швидко змінюються, оскільки зосереджується на статичному аналізі.

SWOT-аналіз є ефективним інструментом стратегічного планування, тому логічним є висновок, що його необхідно використати при написанні застосунку для власників собак.

3.2 MoSCoW-аналіз

MoSCoW-аналіз – це метод пріоритизації вимог у проєктах, особливо популярний при розробці програмного забезпечення [18]. Ця назва містить чотири основні критерії, які використовуються для аналізу проєкту чи продукту. Ось як вони розшифровуються:

- Обов'язкові вимоги (Must have) – критичні вимоги, без яких проєкт не може бути успішно реалізований.
- Should-have – важливі, але не критичні вимоги, які слід впровадити, якщо це можливо.
- Бажані функції (Could have) – додаткові вимоги, які покращують продукт, але не є обов'язковими.
- Необов'язкові вимоги (Would not have), які виключені з поточного обсягу проєкту, але можуть бути розглянуті в майбутньому.

Цей аналіз часто використовується на самому початку розробки продукту, аби зрозуміти, які функції (вимоги) потрібно надати користувачеві зараз, які пізніше, а які слід відкласти [18]. Однак якщо детальніше розглянути цю проблематику, потрібно звернути увагу на основні функції, для яких застосовується MoSCoW-аналіз: встановлення пріоритетів вимог при розробці програмного забезпечення з метою зосередитися на найважливіших завданнях; використовується для виключення менш важливих завдань з поточної фази роботи; використовується також для оптимізації ресурсів, що означає, що команда приділяє більше уваги критичним функціям, які забезпечують головний успіх проєкту.

Переваги цього аналізу легко зрозуміти з попереднього опису, що вказує, чому ця методологія використовується, але доцільно описати недоліки, які можуть виникнути при роботі з цим типом аналізу:

- відсутність формальних критеріїв означає, що немає чітких правил, які б обґрунтовували, що є «обов'язковим» чи «бажаним»;

- складнощі при координації категорій, наприклад надмірна кількість «Must have» може призвести до перевантаження проєкту.

Однак попри всі недоліки цього аналізу він залишається популярним у проєктах і дуже корисним на початку аналізу вимог та цілей продукту. Тому також варто провести MoSCoW-аналіз для мобільного застосунку для власників собак.

3.2. MoSCoW-аналіз для мобільного застосунку для власників собак

Обов'язкові функції (Must have)	Функція пошуку маршрутів прогулянок із собакою з інтеграцією мапи. Мапа з позначеними парками для вигулу собак та правилами відвідування. Чат для спілкування з власниками собак у вашому регіоні. Профіль користувача з усією інформацією про собаку. Забезпечення елементів гейміфікації застосунку для збору балів за щоденні активності. Таблиця лідерів за кількістю прогулянок. Онлайн ветеринарний паспорт у програмі. Реєстрація через Google. Цікаві статті про догляд за собаками.
Важливі функції (Should have)	Календар для планування прогулянок та зустрічей з іншими власниками собак. Сповіщення про погоду при плануванні прогулянок. Інтеграція з ветеринарними аптеками та клініками поблизу на мапі.
Бажані функції (Could have)	Спільна інтеграція з застосунком Здоров'я в iOS та Android. Поради щодо догляду за собакою. Інтеграція інтернет-магазинів з товарами для собак. Функція консультації з ветеринарами через застосунок (через чат або відеоконсультацію).
Непотрібні функції (Would not have)	Офлайн-режим застосунку. Автоматизовані пішохідні маршрути на основі штучного інтелекту. Преміумфункції за дорожчою підпискою. Оцінювання власників собак (відгуки про прогулянки з іншими користувачами).

3.3 Аналіз персон

Аналізи персон є гіпотетичними типами реальних користувачів, які описують їхні цілі, навички та інтереси. Вони використовуються для створення продуктів, орієнтованих на людей [19].

Цей аналіз вважається одним із найважливіших аналізів при створенні будь-якого застосунку та призначений для [19]:

- кращого розуміння користувачів членами команди;
- цей аналіз допомагає адаптувати рішення щодо проєкту так, аби краще відповідати логіці та потребам застосунку;
- допомагає команді створити простіший для використання продукт.

Це лише головні функції, для яких створено цей аналіз, але їх набагато більше. Варто звернути увагу на переваги та недоліки аналізу персон. Думки тут розділилися, і виокремлюються чотири головні переваги і чотири головні недоліки. Переваги:

- аналізи персон створюють спільне розуміння користувачів серед розробників, дизайнерів та менеджерів;
- цей аналіз надає хороші приклади для прийняття рішень щодо функціональності та дизайну застосунку;
- використання цього типу аналізу допомагає скоротити час, який міг би бути витрачений на дослідження та опитування;
- створює спільну думку, аби команда зосередилася на цих потребах користувачів.

Слід також згадати про недоліки, які теж відіграють роль у тому, чому деякі команди можуть відмовлятися використовувати цей тип аналізу: персони часто здаються занадто абстрактними, оскільки не завжди зрозуміло, як дані перетворюються на персони; не є достатньо детальним для комплексних дизайнерських завдань; вибір неправильних та неважливих частин може мати негативний вплив на обмеження дизайну; якщо говорити про великі продукти,

аналіз персон може бути складним для створення і займатиме більше часу, ніж було спочатку заплановано.

Персони є корисним інструментом для комунікації в командах, але не завжди ефективні для безпосереднього проєктування.

Однак варто зазначити, що для малих продуктів це must-have аналіз, який має бути в наявності.



Рис. 3.1. Аналіз першої персони



Рис. 3.2. Аналіз другої персони



Рис. 3.3. Аналіз третьої персони



Рис. 3.4. Аналіз четвертої персони

3.4 Варіанти використання

Діаграма варіантів використання – це графічне зображення функціонування системи, включно з тим, як користувач спілкуватиметься з системою через певні сценарії [20]. Діаграма складається з таких елементів:

- актори – це ролі, що взаємодіють із системою, наприклад користувач або адміністратор системи;

- варіанти використання – це дії або завдання, які виконує система для виконання вимог актора;

- зв'язки між акторами та варіантами використання, які показують їхню взаємодію;

- існує також елемент, який не завжди виокремлюється, але є фундаментальним, як описують автори у своїй статті, – це система – область, що обмежує сценарії, які складають її структуру.

Ця діаграма найчастіше використовується для визначення функціональних вимог – допомагає зрозуміти, що повинна робити система. Варто також зазначити, що варіанти використання допомагають визначити пріоритети функціональності продукту. У деяких випадках тестувальники використовують цю діаграму як основу для тестування [20].

При створенні продукту будь-якого масштабу та складності слід звернути увагу на наступні переваги варіантів використання:

- простота читання та аналізу навіть для нетехнічних учасників проєкту;

- зосередження на тому, як система використовується в реальному житті;

- перевагою є те, що цю діаграму можна використовувати на початкових етапах розробки для детального опису вимог;

- просте масштабування та налаштування функцій.

Слід також звернути увагу на недоліки, які проявляються при розробці варіантів використання:

- діаграма не відображає внутрішньої логіки системи чи технічних деталей;

- у випадку систем з багатьма функціями діаграма може бути занадто великою і важкою для розуміння;

- ця діаграма, на жаль, не відображає послідовність кроків.

Звичайно, як і будь-який інший інструмент аналізу продукту, він має свої переваги та недоліки, але варто зрозуміти, що варіанти використання є одним

з найважливіших аналізів, які потрібні команді, аби зрозуміти, що повинна робити система чи користувач і як вони мають загалом взаємодіяти.

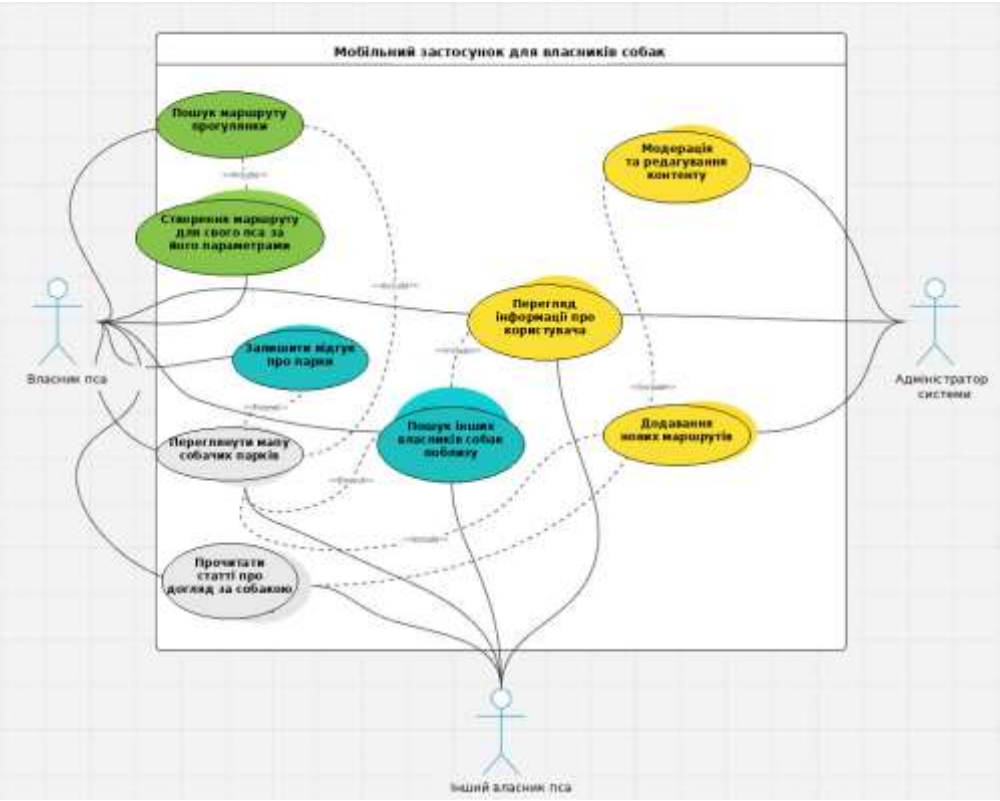


Рис. 3.5. Варіанти використання мобільного застосунку для власників собак

На завершення цієї діаграми варіантів використання доцільно стисло описати, що означають окремі елементи, які тут зображено:

- діаграма зображує акторів, які відіграють роль в інформаційній системі. Першим актором є власник собаки, який використовує застосунок для власних цілей. Другим актором також є власник собаки, але тут показана функція спілкування між двома власниками для спільного вигулу своїх домашніх тварин. Третім актором є адміністратор системи, який може, наприклад, редагувати статті та загалом підтримувати застосунок;

- звичайно, варіанти використання, зображені в цій діаграмі, є різними. Стисло кажучи, варіанти використання, виділені зеленим, – це ті, що пов'язані з налаштуванням вигулу собаки. Сірим позначені варіанти використання, які більше пов'язані з інформаційною функцією. Жовті переважно є функціями,

призначеними для адміністратора системи. На завершення варто згадати і темно-зелений колір варіантів використання – вони пов'язані зі спілкуванням користувачів у застосунку;

- використовуються також відносини include та extend, які дуже важливі для відображення того, як варіанти використання пов'язані між собою.

РОЗДІЛ 4 РОЗРОБКА ЗАСТОСУНКУ ТА ТЕСТУВАННЯ

4.1 Технічна розробка застосунку

У цьому розділі описано процес розробки мобільного застосунку, використані технології та методи реалізації програмного продукту. Особливу увагу приділено вибору інструментів розробки, архітектурі застосунку, реалізації основних функціональних можливостей та забезпеченню зручного користувацького інтерфейсу. Крім того, розглянуто процес тестування MVP-версії мобільного застосунку, наведено основні тестові сценарії, результати перевірки працездатності системи та відгуки користувачів, отримані під час тестування. Проведене тестування дозволило оцінити ефективність реалізованих рішень, виявити можливі недоліки та визначити напрями подальшого вдосконалення застосунку.

Застосунок було розроблено на основі мови програмування Kotlin. Ця мова програмування є основною для створення застосунків для Android. Детальніше її було описано в аналізі роботи, зокрема у статті номер [21].

Середовище програмування було обрано як універсальне для найкращої розробки застосунків – Android Studio. Це середовище дозволяє якісно написати всі необхідні функції та ідеї застосунку, а також протестувати його на емуляторі чи на реальному пристрої. Ці два моменти можна вважати двома найважливішими при створенні застосунку для Android, оскільки без них було б неможливо підключити до проєкту інші функції та технології.

Однак також варто згадати наступні ключові технології, які використовувалися при написанні застосунку. Кожна з цих технологій була необхідна для реалізації різних функцій: Git використовувався для управління версіями, аби всі процеси та файли проєкту не були локальними та аби в разі виникнення помилки можна було повернутися до попередньої версії застосунку; як описано вище, для написання застосунку було використано основну мову програмування Kotlin та середовище програмування Android

Studio. Перед написанням застосунку було два варіанти написання коду – а саме за допомогою мови Flutter, оскільки ця мова програмування є мультиплатформною, але вибір випав на систему Android [10]. Цей вибір був зумовлений двома ситуаціями: по-перше, для тестування мультиплатформності потрібно було мати комп'ютер Mac, а для Android підходила система Windows, по-друге, Kotlin вважається однією з найдинамічніше розвиваючих, як описано у статті, а також найкращим вибором для написання MVP [21].

Для тестування застосунків у процесі розробки використовувався метод тестування емулятора, який мав такі параметри: Pixel 5 API 30, Android 11.0("R") | x86. Цей емулятор було налаштовано безпосередньо в застосунку Android Studio. Варто також згадати про два інші тести. Ці тести буде детальніше описано в окремій частині цього документа, що стосуватиметься тестування мобільного застосунку.

Firebase було під'єднано для забезпечення якісного входу через пароль та електронну пошту, а також для можливості реєстрації/входу через обліковий запис Google.

Для реалізації ідеї гейміфікації та створення бази даних користувачів було під'єднано базу даних Firestore, яка збирає та згодом повертає дані про користувачів, а саме: нік користувача, електронну пошту користувача та кількість ігрових балів.

Однією з найважливіших технологій, використаних при розробці застосунку, є використання різних інтерфейсів API та їх відкритої документації. Наприклад, для створення розділу про цікаві статті використано API newsapi.org, після чого статті було відсортовано за ключовими словами. Для створення ідеї «Instagram для собак» також використано API dog.ceo, а також важливе API Google Maps, яке використано для створення мапи парків для вигулу собак всередині застосунку.

4.1 Перелік основних використаних технологій

Процес	Технологія
Контроль версій та коду	Github через Gitbush console
Програмування	Kotlin
Редактор коду	Android Studio
Емулятор та версія системи Android	Pixel 5 API 30, Android 11.0("R") | x86
База даних	Firestore
Вхід та реєстрація	Firebase
API clients	newsapi.org, dog.ceo; Google Maps SDK для Android

4.2 Функціональна розробка застосунку

Основна функціональність застосунку була реалізована на основі MoSCoW-аналізу, який був проведений під час теоретичного аналізу цього проєкту [18].

Слід також зазначити, що основні функції було взято на основі теоретичного вивчення статей. Цей MVP мобільного застосунку охоплює всі головні аспекти, аби користувач мав точне уявлення про застосунок, про те, що він являє собою та які переваги приносить.

Нижче подано стислий опис основних функцій, завдяки яким ця програма є унікальною у своїй галузі:

Функція чату між користувачами, які вигулюють своїх собак. Ця функція спроєктована таким чином, аби дозволити користувачам зав'язувати нові дружні стосунки і разом гуляти в парку з іншим користувачем під час вигулу свого собаки. Для пошуку нових людей у застосунку також є функція пошуку людей у вашому регіоні та початку чату чи прогулянки з ними.

Функція статей про догляд за собаками та про собак загалом. Ця функція була розроблена так, аби користувачеві не потрібно було самостійно шукати конкретні статті та поради щодо догляду за собакою, а аби все було в одному місці і він міг будь-коли відкрити застосунок та прочитати цікаві моменти, наприклад під час прогулянки. Її було реалізовано за допомогою API і, що важливо, статті відкриваються за допомогою Webview, що дуже зручно,

оскільки користувач завжди залишається в застосунку та не переходить до сторонніх ресурсів.

Функція мапи для зручних прогулянок із собакою. Головною ідеєю цієї функції було надати користувачеві можливість відкрити мапу безпосередньо в застосунку та знайти потрібну інформацію, у цьому випадку є фільтр, який покаже користувачеві лише парки, де він може вигулювати свого собаку.

Початкові елементи гейміфікації. На основі статті [8] виникла ідея, яка дозволяє користувачеві більше зацікавитися частішими прогулянками із собакою та збирати за це додаткові бали. Ця гейміфікація для MVP являє собою таблицю лідерів, у якій користувачі змагаються за те, хто отримає більше балів та виглядатиме успішніше, а також за мотивацію проводити більше часу в застосунку та використовувати його щодня. Ще одним елементом гейміфікації є те, що користувач має нік, а не звичайне ім'я, та має аватар.

Розширена версія профілю користувача. Окрім базової інформації про користувача, доступний також ветеринарний паспорт собаки, який містить основну інформацію про тварину. Цей ветеринарний паспорт також можна верифікувати за допомогою QR-коду, який динамічно змінюється кожні 3 секунди, тому його неможливо підробити.

Варто також зазначити, що застосунок має домашню сторінку, яка містить базову інформацію про застосунок, а також інші функції, наприклад «Dogs Instagram», де користувач може переглядати фотографії собак, які завантажуються через інтерфейс API. Задумом було створити не лише застосунок, який був би корисним при вигулі собаки, але й застосунок, який використовувався б щодня та допомагав би покращити процес вигулу та догляду за собаками.

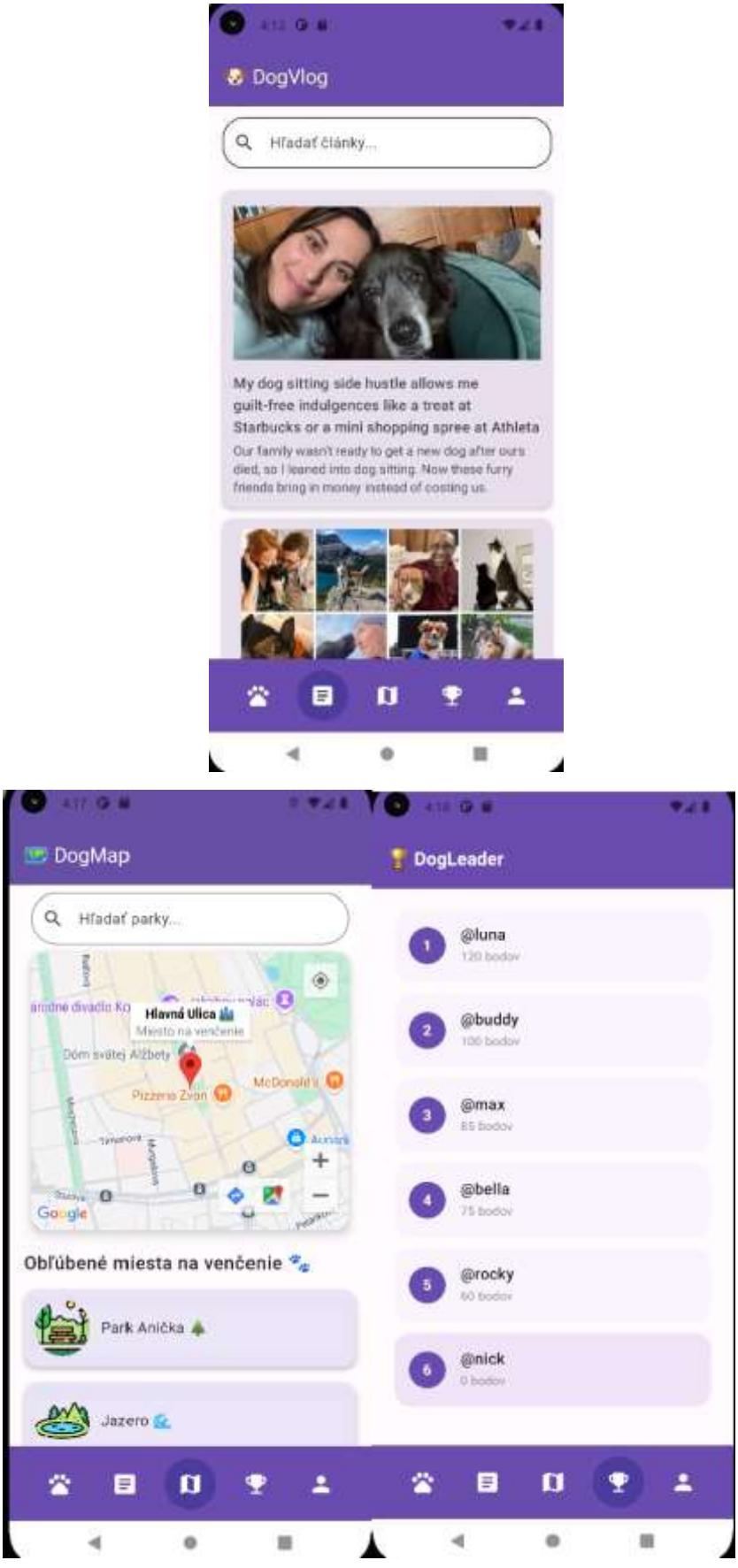


Рис. 4.1 Приклади основних сторінок застосунку

4.3 Тестування

Процес тестування проєкту застосунку для вигулу та догляду за собаками проходив у кілька етапів. Кожен з цих етапів був ключовим і повинен був принести певні позитивні результати, аби в кінці можна було стверджувати, що MVP застосунку успішно пройшло тестування і готове до базового початкового використання або, можливо, до подальшого вдосконалення.

Системне тестування

Першим кроком у процесі тестування було тестування підсумкового файлу .apk за допомогою служби Firebase Test Lab. Ця служба дозволяє автоматизоване тестування Android-застосунків на різних віртуальних та фізичних пристроях у хмарному середовищі.

У межах цього тестування використовувався Robo test, який без потреби писати тестовий код симулює поведінку користувача та автоматично взаємодіє із застосунком.

Метою було перевірити базову функціональність, стабільність застосунку та його правильний запуск на різних версіях операційної системи Android. Після запуску тесту було згенеровано вихідні дані: відеозаписи перебігу тесту; знімки екрана; логи про виникнення помилок та винятків.

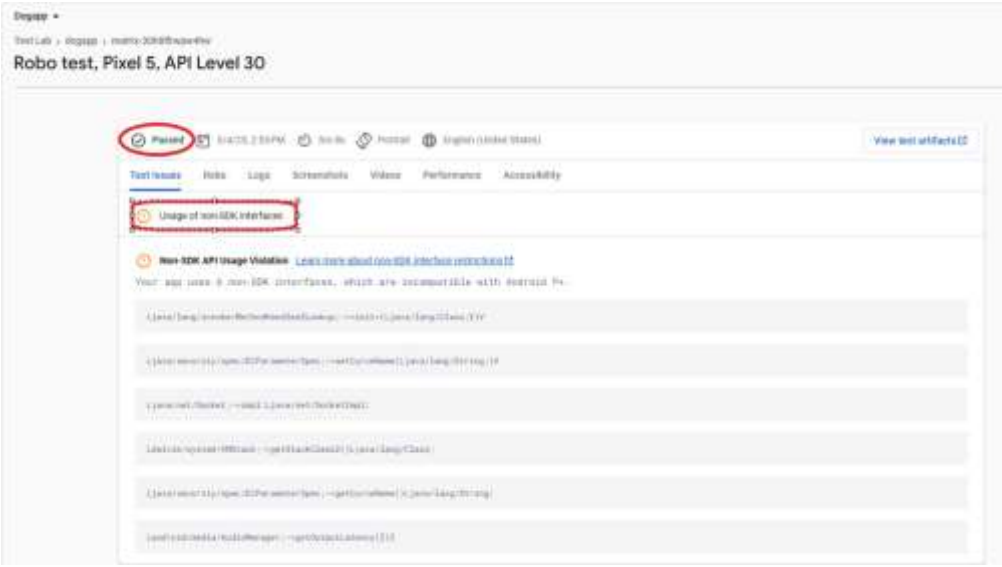


Рис. 4.2. Висновок тестування за допомогою Robo test

Зі зображення вище можна зробити висновок, що застосунок успішно пройшов тестування. Однак у логах з'явилося 6 попереджень, які пов'язані переважно з версією системи Android. Тому в майбутніх версіях та при подальшій розробці застосунку потрібно буде звернути на ці попередження підвищену увагу.



Рис. 4.3. Запис споживання системних ресурсів під час тесту

Це тестування було надзвичайно важливим, оскільки саме воно могло показати реальну картину того, як застосунок поводиться при навантаженні та тестових вимогах. Значення такого тестування, а також важливість якісного тестування загалом було проаналізовано також у статті про тестування у чужому середовищі [2].

Користувацьке тестування

Користувацьке тестування було проведено серед шести учасників, кожному з яких було призначено власний тестовий сценарій. Метою було отримати зворотний зв'язок щодо загальної зручності застосунку та перевірити окремі функціональні можливості з точки зору реального користувача.

Кожен користувач виконав серію завдань за індивідуальним тестовим сценарієм. Ці сценарії були розроблені так, аби охопити різні частини функціоналу MVP-версії застосунку. Ось перелік 6 тестових сценаріїв:

1. Зареєструватися → Обрати нік → Увійти в застосунок → Знайти статтю на «Help» → Написати повідомлення користувачу → Переглянути паспорт щеплень собаки → Вийти.

2. Зареєструватися → Обрати нік → Увійти → Переглянути фотографії собак → Знайти людей поблизу → Переглянути свій рейтинг → Вийти.

3. Увійти через Google → Переглянути мапу → Знайти парк Jazero → Прочитати інформацію про застосунок → Вийти.

4. Увійти через Google → Знайти статтю на «Т» → Прочитати статтю → Перевірити паспорт щеплень → Вийти.

5. Зареєструватися → Обрати нік → Увійти → Розпочати прогулянку → Обрати парк на мапі → Відкрити головний профіль → Вийти.

6. Зареєструватися → Обрати нік → Увійти → Знайти розмову з «kubo2» → Надіслати повідомлення → Переглянути два парки → Перевірити паспорт щеплень → Вийти.

Слід також зазначити, що цей тест проводився серед людей, які мають або колись мали собаку, оскільки саме такі користувачі можуть надати щиру думку про реальне використання цього застосунку.

Після завершення тестування кожен учасник заповнив Google-опитування, яке складалося з двох частин:

1. Перших 5 запитань склав розробник застосунку, аби краще зрозуміти основні проблеми чи успіхи застосунку.

2. Друга частина базувалася на методиці SUS і складалася з 10 стандартизованих запитань, призначених для оцінки зручності використання програмного продукту.

На основі оцінки опитувань було розраховано SUS score = 84,58%, що є дуже хорошим результатом для MVP-версії застосунку.

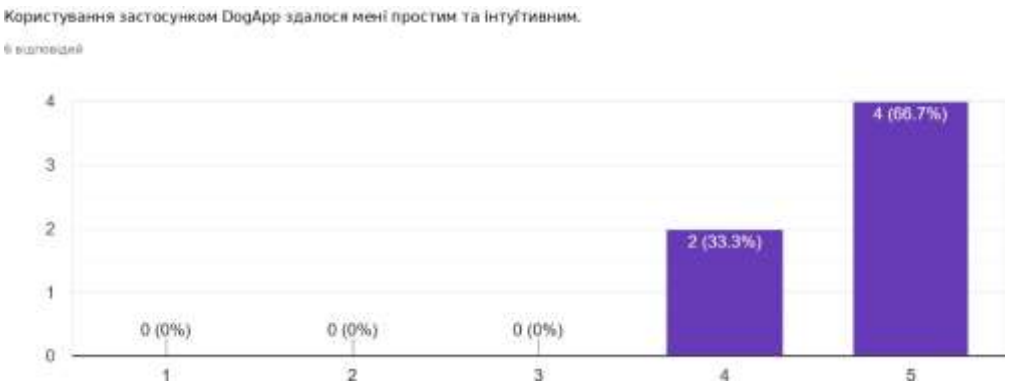


Рис. 4.4. Приклад візуалізації важливих результатів опитування

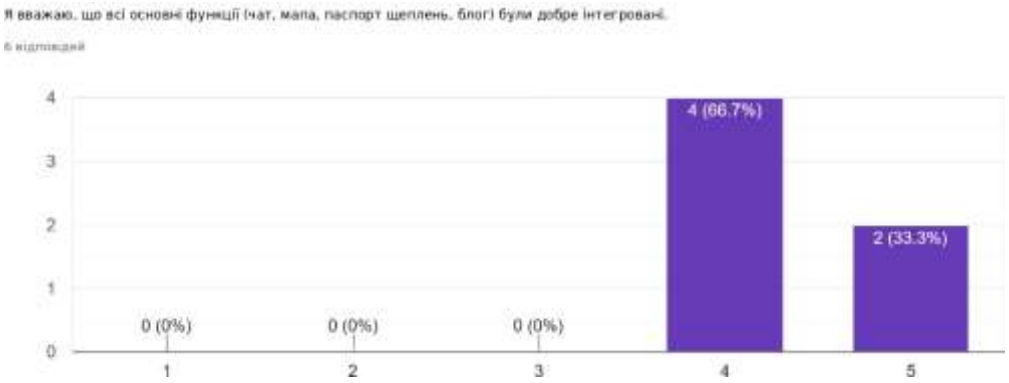


Рис. 4.5. Візуалізація результату щодо інтеграції основних функцій

Наприкінці тестування деякі користувачі також залишили зауваження щодо вдосконалення чи зауваження до MVP застосунку. Ці зауваження будуть збережені та використані для покращення подальших версій мобільного застосунку. Приклади зауважень наведено нижче.

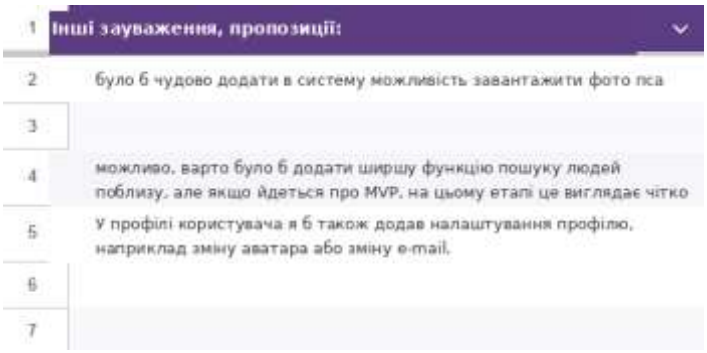


Рис. 4.6. Зауваження опитаних користувачів

На основі результатів тестування, зокрема технічного та користувацького тестування MVP, можна стверджувати, що застосунок справив враження, яке спочатку було поставлене як головна мета. Це означає, що застосунок виконує всі основні функції, які роблять його унікальним для користувачів, і з технічного боку він добре реалізований та зібраний.

ВИСНОВКИ

Предметом представленої кваліфікаційної роботи бакалавра було проєктування та розробка інноваційного мобільного застосунку для системи Android, який би комплексно підтримував догляд за собаками і водночас пропонував функціональні можливості, що полегшують повсякденне життя їхніх власників. Результатом роботи є функціональна MVP-версія застосунку, яка поєднує практичні функції (як-от планування прогулянок, ведення ветеринарного паспорта, мапа собачих парків та статті про догляд) з гейміфікаційними та комунальними елементами, спрямованими на тривалу залученість користувачів.

На основі аналізу сучасних рішень, вибору відповідних технологій та застосування методик SWOT- та MoSCoW-аналізу, аналізу персон та варіантів використання вдалося створити концепцію, що відповідає актуальним тенденціям у сфері мобільної розробки. При реалізації використовувалися технології Kotlin, Firebase, Firestore та Google Maps API, що дозволило реалізувати надійну та масштабовану архітектуру. Результати користувацького та системного тестування підтвердили високий рівень зручності використання застосунку та позитивні реакції цільової групи.

Значним елементом роботи було також ефективне використання штучного інтелекту під час окремих фаз розробки. Штучний інтелект використовувався для кількох допоміжних цілей, включно з автоматизованим перекладом текстів, корекцією орфографічних та стилістичних помилок, узагальненням наукових статей з метою швидшого розуміння їхнього змісту, а також при генерації дизайнерських компонентів користувацького інтерфейсу. Таким чином вдалося значно скоротити час, потрібний на повторювані дії, та зосередитися на концептуальних та розробницьких аспектах рішення.

Внесок роботи полягає насамперед у поєднанні кількох сфер – технічної розробки, UX-дизайну, штучного інтелекту та користувацько-орієнтованого

проєктування – в єдине ціле, яке має потенціал реального впровадження. Застосунок водночас відкриває можливості для подальшого розвитку, як-от розширення новими функціями (наприклад, відеоконсультаціями з ветеринаром, персоналізованими рекомендаціями на основі поведінки собаки, інтеграцією з носимими пристроями) або мультиплатформним розширенням для систем iOS.

Серед відкритих питань, що виходять за межі цієї роботи, можна, наприклад, назвати тривале зберігання даних у хмарі та побудову довіри між користувачами в рамках комунальних функцій. Так само як виклик на майбутнє постає можливість монетизації застосунку без негативного впливу на користувацький досвід.

Загалом можна стверджувати, що цілі роботи були виконані, а створене рішення являє собою відповідну основу для подальших досліджень та практичної реалізації у сфері цифрового догляду за тваринами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Israel J. Mojica Ruiz, Meiyappan Nagappan, Bram Adams, Ahmed E. Hassan. Understanding Reuse in the Android Market. Queen's University, Kingston, Canada. Доступно онлайн: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6240477.

2. Fabiano Pecorelli, Gemma Catolino, Filomena Ferucci, Andrea De Lucia, Fabio Palomba. Testing of Mobile Applications in the Wild. University of Salerno, Italy. Доступно онлайн: https://dl.acm.org/doi/abs/10.1145/3387904.3389256.

3. Venkata N. Inukollu, Divya D. Keshamoni, Taeghyun Kang, Manikanta Inukollu. Factors Influencing Quality Of Mobile Apps: Role Of Mobile App Development Life Cycle. Texas Tech University, USA. Доступно онлайн: https://arxiv.org/abs/1410.4537.

4. Demian C. Johnson, P. Reeja George, Mahesh Chander. Demographic and Perception Studies of a Mobile Application among Dog Breeders and Owners. Доступно онлайн: https://www.extensioneducation.org/index.php/jee/article/view/966.

5. Anisa Rahmi, I Nyoman Piarsa, Putu Buana. FinDoctor – Interactive Android Clinic Geographical Information System Using Firebase and Google Maps API. International Journal of New Technology and Research (IJNTR). Доступно онлайн: https://www.ijntr.org/download_data/IJNTR03070047.pdf.

6. Daniel Matisoff, Douglas Noonan. Managing contested greenspace: neighborhood commons and the rise of dog parks. International Journal of the Commons. Доступно онлайн: https://www.jstor.org/stable/26523090?seq=1.

7. Claire Micklin. DogTracker: A Mobile App Engaging Citizens and Officials in Addressing the Stray Dog Crisis. Chicago, IL, USA. Доступно онлайн: https://dl.acm.org/doi/abs/10.1145/2693787.2693798.

8. Aatish Neupane, Derek Hansen, Anud Sharma, Jerry Alan Fails, Bikalpa Neupane, Jeremy Beutler. A Review of Gamified Fitness Tracker Apps and Future

Directions. Virtual Event, Canada. Доступно онлайн: https://dl.acm.org/doi/abs/10.1145/3410404.3414258.

9. R. E. Matos, T. Jakuba, I. Mino, M. Fejsakova, A. Demeova, J. Kottferova. Characteristics and risk factors of dog aggression in the Slovak Republic. University of Veterinary Medicine and Pharmacy in Kosice, Slovak Republic. Доступно онлайн: https://www.agriculturejournals.cz/pdfs/vet/2015/08/05.pdf.

10. Alexandru Tabusca, Cristina Coculescu, Mironela Pirnau. FLUTTER TECHNOLOGY AND MOBILE SOFTWARE APPLICATIONS. Journal of Information Systems & Operations Management, Vol. 16.2. Доступно онлайн: https://web.rau.ro/websites/jisom/Vol.16%20No.2%20-%202022/JISOM%2016.2_250-262.pdf.

11. Vipul Kaushik, Kamali Gupta, Deepali Gupta. React Native Application Development. International Journal of Advanced Studies of Scientific Research, Vol. 4, No. 1. Доступно онлайн: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3330011.

12. Gokul Ramadoss. Choosing Xamarin Platform for App Development. North American Journal of Engineering and Research. Доступно онлайн: https://najer.org/najer/article/view/56.

13. Priyanka Chaudhary. IONIC FRAMEWORK. International Research Journal of Engineering and Technology (IRJET), Volume: 05 Issue: 05. Доступно онлайн: https://www.irjet.net/archives/V5/i5/IRJET-V5I5609.pdf.

14. Tejas Vithani, Anand Kumar. Modeling the Mobile Application Development Lifecycle. Proceedings of the International MultiConference of Engineers and Computer Scientists 2014 Vol I. Доступно онлайн: https://www.iaeng.org/publication/IMECS2014/IMECS2014_pp596-600.pdf.

15. Hao Zhong, Hong Mei. An Empirical Study on API Usages. IEEE Transactions on Software Engineering, Vol. 45, No. 4. Доступно онлайн: https://ieeexplore.ieee.org/abstract/document/8186224.

16. Gemma Catolino, Fabio Palomba, Andy Zaidman, Filomena Ferrucci. How the Experience of Development Teams Relates to Assertion Density of Test Classes. IEEE International Conference on Software Maintenance and Evolution. Доступно онлайн: https://ieeexplore.ieee.org/abstract/document/8919174.

17. Mostafa Ali Benzaghta, Abdulaziz Elwalda, Mousa Mohamed Mousa, Ismail Erkan, Mushfiqur Rahman. SWOT analysis applications: An integrative literature review. Journal of Global Business Insights, Vol. 6. Доступно онлайн: https://digitalcommons.usf.edu/globe/vol6/iss1/5.

18. Suchetha Vijayakumar, Krishna Prasad K., Raviraja Holla M. Assessing the Effectiveness of MoSCoW Prioritization in Software Development: A Holistic Analysis across Methodologies. EAI Endorsed Transactions on Internet of Things, Vol. 10 (2024). Доступно онлайн: https://publications.eai.eu/index.php/IoT/article/view/6515.

19. Tara Matthews, Tejinder K. Judge, Steve Whittaker. How do designers and user experience professionals actually perceive and use personas? CHI '12: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Доступно онлайн: https://dl.acm.org/doi/abs/10.1145/2207676.2208573.

20. Reza Fauzan, Daniel Siahaan, Siti Rochimah, Evi Triandini. Use Case Diagram Similarity Measurement: A New Approach. 12th International Conference on Information & Communication Technology and System (ICTS). Доступно онлайн: https://ieeexplore.ieee.org/abstract/document/8850978.

21. Bruno Gois Mateus, Matias Martinez. On the adoption, usage and evolution of Kotlin features in Android development. Bari, Italy. Доступно онлайн: https://dl.acm.org/doi/abs/10.1145/3382494.3410676.

22. Malte Kraus, Vincent Haupert. The Swift Language from a Reverse Engineering Perspective. Vienna, Austria. Доступно онлайн: https://dl.acm.org/doi/abs/10.1145/3289595.3289599.