МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ПОШУКУ ТЕЛЕФОНУ ЗА ДОПОМОГОЮ REACT NATIVE»**

Виконав: студент 3 курсу, групи 6.1211-пі-с
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія (зі скороченим терміном навчання)
(назва освітньої програми)
О.А. Базилевський
(ініціали та прізвище)
Керівник доцент кафедри програмної інженерії, доцент, к.т.н. Мухін В.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія (зі скороченим терміном навчання)

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Базилевському Олексію Андрійовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка мобільного застосунку для пошуку телефону за допомогою React Native

керівник роботи Мухін Віталій Вікторович, к.т.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2180-с

2. Строк подання студентом роботи 03.06.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Проєктування та реалізація програмного забезпечення.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 25.12.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	09.01.2024	
2.	Збір вихідних даних.	22.01.2024	
3.	Обробка методичних та теоретичних	12.02.2024	
	джерел.		
4.	Розробка першого та другого розділу.	01.04.2024	
5.	Розробка третього розділу.	20.05.2024	
6.	Оформлення та нормоконтроль	27.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	21.06.2024	

Студент _____ О.А. Базилевський
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.В. Мухін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка мобільного застосунку для пошуку телефону за допомогою React Native»: 45 с., 18 рис., 19 джерел.

БАГАТОМОВНИЙ, ІНТЕРФЕЙС КОРИСТУВАЧА, МОБІЛЬНИЙ ЗАСТОСУНОК, МОНЕТИЗАЦІЯ, НАЛАШТУВАННЯ, ПОШУК ТЕЛЕФОНУ, РЕКЛАМА, ФУНКЦІОНАЛЬНІСТЬ.

Об’єкт дослідження – процес розробки мобільного застосунку для пошуку телефону за допомогою React Native.

Мета роботи: розробка мобільного застосунку для пошуку телефону, що включає реалізацію основних функціональних можливостей та інтеграція реклами для монетизації.

Метод дослідження – використання сучасних методів розробки програмного забезпечення, застосування бібліотек для мобільної розробки, тестування користувацького інтерфейсу, аналіз ефективності використання ресурсів пристрою.

У даній кваліфікаційній роботі описано процес розробки мобільного застосунку для пошуку телефону на базі платформи React Native. Застосунок дозволяє користувачеві швидко знайти свій телефон за допомогою звуку (хлопок у долоні), вібрації або ліхтарика.

На початку роботи було проведено аналіз вимог до програмного забезпечення та визначено основні функціональні можливості. Застосунок включає такі екрани: екран завантаження, вибір мови, карусель з інструкціями, головна сторінка з основними функціями, налаштування звуків, отримання доступу до мікрофона та повідомлень, а також екран налаштувань.

SUMMARY

Bachelor’s qualifying paper “Development of a Mobile Application for Finding a Phone Using React Native”: 45 pages, 18 figures, 19 references.

MULTILINGUAL, USER INTERFACE, MOBILE APPLICATION, MONETIZATION, SETTINGS, PHONE FINDER, ADVERTISEMENT, FUNCTIONALITY.

The object of the study is the development process of a mobile application for finding a phone using React Native.

The aim of the study is to develop a mobile application for finding a phone, which includes the implementation of basic functionalities such as phone finding by sound and integrating advertisements for monetization.

The method of research is using modern software development methods such as object-oriented programming, utilizing libraries for mobile development, user interface testing, and analyzing the efficiency of device resource usage.

This bachelor’s thesis describes the development of a mobile application for finding a phone using the React Native platform. The app helps users quickly find their phone with sound (clapping), vibration, or a flashlight.

At the start, a requirements analysis was done, and the main features were defined. The app includes several screens: a loading screen, language selection, an instruction carousel, the main screen with core functions, sound settings, microphone and notification access, and a settings screen.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Вступ 8
1 Огляд Тенденцій, платформ та технологій 10
1.1 Сучасні тенденції в розробці мобільних застосунків 10
1.2 Платформа React Native та її особливості 11
1.3 Порівняння React Native та інших засобів розробки мобільних застосунків 13
1.3.1 React Native 13
1.3.2 Flutter 14
1.3.3 Xamarin 14
1.3.4 NativeScript 15
1.3.5 Висновки 15
1.4 Аналіз застосунків конкурентів 16
1.4.1 Додаток “Find My Phone Whistle” 16
1.4.2 Додаток “Clap to Find” 17
1.4.3 Додаток “Phone Finder by Clap” 18
1.5 Технології для реалізації функцій пошуку телефону 19
1.5.1 Виявлення звуків 19
1.5.2 Використання вібрації та ліхтарика 20
1.5.3 Інші корисні бібліотеки та інструменти 20
1.5.4 Інтеграція з рекламними сервісами 20
2 Планування та реалізація 22
2.1 Архітектура мобільного застосунку 22
2.2 Реалізація основних та другорядних функцій 23
2.2.1 Обробка аудіосигналу та знаходження телефону 23

2.2.2 Інтернаціоналізація мобільного застосунку 26
2.2.3 Керування станом 28
2.2.4 Опис роботи та роль React Navigation у проєкті 31
3 Тестування, монетизація та огляд готового продукту 34
3.1 Тестування та налагодження застосунку 34
3.2 Тестування та налагодження застосунку 35
3.3 Опис закінченого продукту 37
3.4 Огляд конкурентних застосунків 40
Висновки 43
Перелік посилань 44

ВСТУП

Розвиток мобільних технологій та зростання кількості користувачів смартфонів створюють нові можливості для розробки програмного забезпечення, що задовольняє різноманітні потреби користувачів. Однією з таких потреб є можливість швидкого знаходження телефону, що може бути втрачений в межах квартири або офісу. Саме цій проблемі присвячена дана кваліфікаційна робота.

Актуальність теми полягає в необхідності забезпечення користувачів зручним інструментом для пошуку телефону. Сучасні смартфони надають широкий спектр можливостей для розробки таких застосунків, зокрема, завдяки інтегрованим датчикам та можливості обробки сигналів. Використання платформи React Native дозволяє створювати кросплатформені рішення, що є додатковою перевагою, оскільки забезпечує доступність застосунку для користувачів як на платформі iOS, так і на Android.

Мета роботи полягає в розробці мобільного застосунку для пошуку телефону, що включає реалізацію основних функціональних можливостей, таких як пошук телефону по звуку (хлопок у долоні), використання вбудованого ліхтарика та вібрації, а також інтеграція реклами для монетизації.

Завдання роботи:

- провести аналіз вимог до програмного забезпечення;
- вибрати архітектуру та технології для розробки мобільного застосунку;
- реалізувати основні функціональні модулі застосунку;
- забезпечити багатомовність інтерфейсу користувача;
- інтегрувати рекламні банери та reward-рекламу для монетизації застосунку;
- провести тестування застосунку та оптимізувати його продуктивність;
- підготувати рекомендації щодо подальшого розвитку застосунку.

Об’єкт дослідження: процес розробки мобільного застосунку для пошуку телефону за допомогою React Native.

Предмет дослідження: технології та методи, що використовуються для створення мобільного застосунку на платформі React Native, а також інструменти монетизації та забезпечення зручності користування.

Методи дослідження: у роботі використано сучасні методи розробки програмного забезпечення, зокрема об’єктно-орієнтоване програмування, тестування користувацького інтерфейсу, аналіз ефективності використання ресурсів пристрою, а також інтеграцію сторонніх бібліотек для розширення функціональних можливостей.

Структура роботи: кваліфікаційна робота складається з вступу, огляду літератури, розділу про проєктування та розробку мобільного застосунку, розділу про тестування та оптимізацію, висновків та списку використаних джерел. Додатки містять додаткові матеріали, такі як фрагменти коду та скріншоти інтерфейсу застосунку.

1 ОГЛЯД ТЕНДЕНЦІЙ, ПЛАТФОРМ ТА ТЕХНОЛОГІЙ

1.1 Сучасні тенденції в розробці мобільних застосунків

Розробка мобільних застосунків є однією з найбільш динамічно розвиваючихся галузей IT-індустрії. Сучасні тенденції в цій сфері визначаються як технічними новаціями, так і змінами в потребах та поведінці користувачів. Є ряд тенденцій, які будуть зазначені нижче.

Кросплатформенна розробка: зростає популярність кросплатформенних інструментів, таких як React Native, Flutter, Xamarin, які дозволяють створювати застосунки одночасно для iOS та Android. Це значно знижує витрати на розробку та підтримку застосунків, забезпечуючи при цьому високу якість та продуктивність [1].

Використання штучного інтелекту (AI) та машинного навчання (ML): Інтеграція AI та ML в мобільні застосунки дозволяє створювати більш персоналізовані та інтелектуальні рішення. Наприклад, застосунки можуть адаптуватися до поведінки користувача, пропонуючи йому релевантний контент, або забезпечувати голосове керування та розпізнавання зображень.

Інтернет речей (IoT): застосунки для керування IoT-пристроями стають все більш поширеними, дозволяючи користувачам контролювати різноманітні пристрої, підключені до інтернету (розумні будинки, автомобілі тощо).

Підвищена увага до безпеки: зростаючий обсяг персональних даних, що зберігаються в мобільних застосунках, вимагає високого рівня захисту. Розробники приділяють більше уваги шифруванню даних, біометричній аутентифікації та іншим методам забезпечення безпеки.

Зростання популярності безконтактних технологій: пандемія COVID-19 сприяла поширенню безконтактних технологій, таких як NFC-платежі, QR-коди та розпізнавання облич. Це також впливає на розробку мобільних застосунків, що підтримують безконтактні операції.

Гейміфікація: включення елементів гри у неігрові застосунки стає все

більш популярним. Гейміфікація сприяє підвищенню залученості користувачів, збільшуючи їх мотивацію використовувати застосунок.

Мікросервіси та хмарні технології: розробка застосунків за допомогою мікросервісної архітектури та використання хмарних сервісів дозволяє підвищити гнучкість, масштабованість та надійність застосунків. Це також спрощує процес розгортання та оновлення програмного забезпечення.

Розширена реальність (AR) та віртуальна реальність (VR): AR та VR технології знаходять своє застосування не тільки у сфері ігор, але й у навчанні, маркетингу, медицині та інших галузях. Інтеграція AR та VR у мобільні застосунки відкриває нові можливості для взаємодії з користувачами.

Прогресивні вебзастосунки (PWA): PWA комбінують переваги вебсайтів та мобільних застосунків, пропонуючи швидкість, офлайн-доступ та можливість встановлення на головний екран без необхідності проходження через магазини застосунків.

Підвищена увага до користувацького досвіду (UX): успіх мобільного застосунку значною мірою залежить від якості користувацького досвіду. Розробники все більше фокусуються на створенні інтуїтивно зрозумілих, привабливих та зручних інтерфейсів.

Сучасні тенденції в розробці мобільних застосунків показують, що для успішної роботи у цій галузі необхідно постійно відслідковувати нові технології, адаптуватися до змін у потребах користувачів та використовувати найкращі практики програмування. Використання платформи React Native у розробці мобільного застосунку для пошуку телефону дозволяє ефективно реалізувати всі основні вимоги та забезпечити високий рівень функціональності та зручності користування.

1.2 Платформа React Native та її особливості

React Native – це популярна платформа для розробки кросплатформених мобільних застосунків, створена компанією Facebook. Вона дозволяє

розробникам використовувати мову програмування JavaScript і бібліотеку React для створення нативних мобільних застосунків для iOS та Android одночасно. Завдяки цьому розробка стає більш ефективною та менш затратною [1]. Наведемо основні особливості платформи React Native.

Кросплатформеність: React Native дозволяє писати один код для двох основних мобільних платформ – iOS та Android. Це значно скорочує час і ресурси, необхідні для розробки та підтримки застосунків.

Нативні компоненти: React Native використовує нативні компоненти для створення користувацького інтерфейсу. Це означає, що застосунки, створені за допомогою React Native, мають нативний вигляд і відчуття, що покращує користувацький досвід.

Hot reloading: ця функція дозволяє розробникам миттєво бачити зміни в коді без необхідності перезавантаження всього застосунку. Це значно пришвидшує процес розробки та дебаггінгу.

Багатий екосистема і бібліотеки: React Native має велику кількість готових бібліотек і модулів, які можна легко інтегрувати в проєкт. Це дозволяє швидко додавати нові функції та розширювати можливості застосунку.

Висока продуктивність: завдяки використанню нативних компонентів і оптимізації процесу рендерингу, застосунки, створені за допомогою React Native, можуть досягати високої продуктивності та плавності роботи.

Спільнота та підтримка: React Native має велику і активну спільноту розробників, що сприяє швидкому вирішенню проблем і обміну знаннями. Крім того, платформа постійно оновлюється та вдосконалюється завдяки підтримці Facebook.

Модульна архітектура: React Native підтримує модульний підхід до розробки, що дозволяє розділяти проєкт на окремі компоненти. Це спрощує підтримку і масштабування застосунку.

Інтеграція з нативним кодом: у разі необхідності, React Native дозволяє писати нативний код для специфічних платформ, що дозволяє використовувати всі можливості пристрою та системи. Це робить платформу дуже гнучкою та універсальною.

Багато відомих компаній використовують React Native для розробки своїх мобільних застосунків. Серед них Facebook, Instagram, Airbnb, Tesla, Skype, і багато інших. Це свідчить про надійність і ефективність платформи.

Основні бібліотеки та інструменти які використовують при створені застосунків на базі React Native:

- redux: використовується для керування станом застосунку;
- react navigation: забезпечує зручну навігацію між екранами застосунку;
- react native firebase: інтеграція з сервісами Firebase, такими як аналітика, аутентифікація, бази даних тощо;
- styled-components: дозволяє використовувати CSS у JavaScript для стилізації компонентів;
- react native permissions: забезпечує зручний доступ до системних дозволів на пристроях.

Отже, з усього вищесказаного, робимо висновок, що React Native є потужним інструментом для розробки сучасних мобільних застосунків. Завдяки його особливостям, розробники можуть швидко і ефективно створювати високоякісні застосунки, що працюють на різних платформах.

Використання React Native у проєкті з розробки застосунку для пошуку телефону дозволило досягти необхідного рівня функціональності та забезпечити зручність користування для широкого кола користувачів.

1.3 Порівняння React Native та інших засобів розробки мобільних застосунків

1.3.1 React Native

React Native, розроблений компанією Facebook, є потужним інструментом для створення кросплатформних мобільних додатків, дозволяючи використовувати один кодовий базис для iOS і Android [1].

Основна перевага React Native полягає в його продуктивності завдяки використанню рідних компонентів. Це забезпечує високу швидкість та ефективність роботи додатків. Крім того, велика спільнота розробників та численні готові рішення сприяють швидкій розробці та усуненню проблем. Однією з ключових функцій є “Hot Reloading”, що дозволяє розробникам вносити зміни в реальному часі та миттєво бачити результати, що значно підвищує продуктивність [1].

Однак, є й певні недоліки. Наприклад, для реалізації деяких специфічних функцій може знадобитися написання рідного коду, що вимагає додаткових знань та ресурсів. Також, складні анімації іноді можуть працювати не так гладко, як в рідних додатках.

1.3.2 Flutter

Flutter, розроблений Google, також є популярним інструментом для кросплатформної розробки. Він вирізняється високою продуктивністю завдяки власному графічному движку та можливістю створення кастомних інтерфейсів з гнучкими можливостями дизайну.

Flutter дозволяє розробникам використовувати єдиний кодовий базис для iOS і Android, що спрощує процес розробки і знижує витрати часу.

Недоліками Flutter є великий розмір додатків у порівнянні з іншими технологіями, а також менша спільнота розробників, що означає меншу кількість готових рішень та бібліотек. Це може створювати певні труднощі при розробці великих і складних проєктів [3].

1.3.3 Xamarin

Xamarin, створений Microsoft, інтегрується з .NET, що робить його привабливим для розробників, які працюють з цією екосистемою. Xamarin

надає повний доступ до рідних API, дозволяючи використовувати рідний код для створення додатків з високою продуктивністю і доступом до всіх функцій платформи.

Недоліки Xamarin включають складність налаштувань середовища розробки та великий розмір додатків. Це може створити додаткові труднощі для розробників, особливо для тих, хто не знайомий з екосистемою .NET [4].

1.3.4 NativeScript

NativeScript також є потужним інструментом для кросплатформної розробки, надаючи розробникам повний доступ до рідних API. Це дозволяє створювати високопродуктивні додатки з використанням одного кодового базису для iOS і Android.

Основними недоліками NativeScript є менша спільнота розробників та складність у підтримці. Для деяких завдань може знадобитися знання рідного коду, що може ускладнити розробку та підтримку проєктів [5].

1.3.5 Висновки

Кожен інструмент для розробки мобільних додатків має свої унікальні переваги та недоліки. React Native вирізняється високою продуктивністю та швидкістю розробки завдяки великій спільноті та готовим рішенням, але може вимагати використання рідного коду для реалізації специфічних функцій. Flutter пропонує гнучкі можливості дизайну та високу продуктивність, але має великий розмір додатків та меншу спільноту. Xamarin інтегрується з .NET і надає повний доступ до рідних API, але складний у налаштуванні. NativeScript забезпечує доступ до рідних API, але має меншу спільноту та вимагає знання рідного коду для деяких задач.

Вибір інструменту залежить від специфічних потреб проєкту, наявних ресурсів та навичок команди.

1.4 Аналіз застосунків конкурентів

1.4.1 Додаток “Find My Phone Whistle”

Додаток “Find My Phone Whistle” (див. рис. 1.1) дозволяє користувачам знаходити свій телефон за допомогою свисту. Основна перевага цього додатку полягає в його простоті та легкості використання. Після встановлення, користувачі можуть просто свистнути, і телефон видасть звуковий сигнал, допомагаючи знайти його.

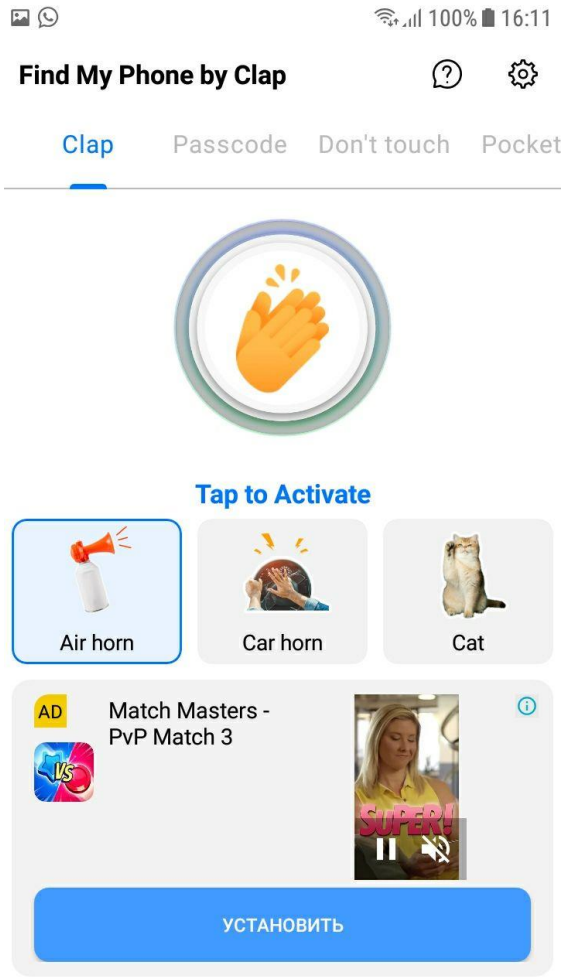

Рисунок 1.1 – Застосунок “Find My Phone Whistle”

Додаток реагує на свист користувача, налаштовуючи чутливість мікрофона для найкращого виявлення звуку.

Переваги:

– простота налаштування і використання;
– не потребує підключення до інтернету;
– висока чутливість до свисту.

Недоліки:

– можлива помилкова активація через інші подібні звуки;
– обмежена ефективність у шумних середовищах.

1.4.2 Додаток “Clap to Find”

“Clap to Find” (див. рис. 1.2) є ще одним популярним додатком для знаходження телефону за допомогою хлопка.

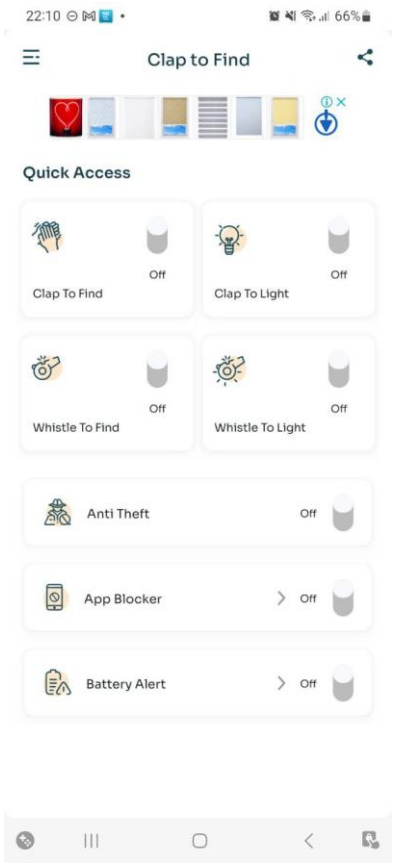

Рисунок 1.2 – Застосунок “Clap to Find”

Цей додаток відрізняється підтримкою різних типів сигналів та налаштувань інтервалу реакції, що дозволяє користувачам налаштовувати додаток під свої потреби.

Додаток реагує на хлопки, підтримуючи налаштування різних сигналів та інтервалу між ними для точнішого виявлення.

Переваги:

– висока ефективність виявлення телефону;
– підтримка різних типів сигналів;
– налаштування інтервалу реакції.

Недоліки:

– високе споживання батареї через постійне прослуховування;
– залежність від якості мікрофона, що може впливати на ефективність роботи.

1.4.3 Додаток “Phone Finder by Clap”

“Phone Finder by Clap” (див. рис. 1.3) використовує хлопок для активації сигналу на телефоні. Цей додаток пропонує більш гнучкі налаштування, дозволяючи користувачам налаштувати чутливість до звуків та обирати різні типи сигналів.

Додаток реагує на хлопки, дозволяючи користувачам налаштувати чутливість мікрофона та вибрати бажаний звуковий сигнал.

Переваги:

– зручний інтерфейс користувача;
– можливість налаштування чутливості;
– вибір типу звукового сигналу.

Недоліки:

– може не працювати належним чином у шумних умовах;
– споживає більше енергії через постійне прослуховування звуків.

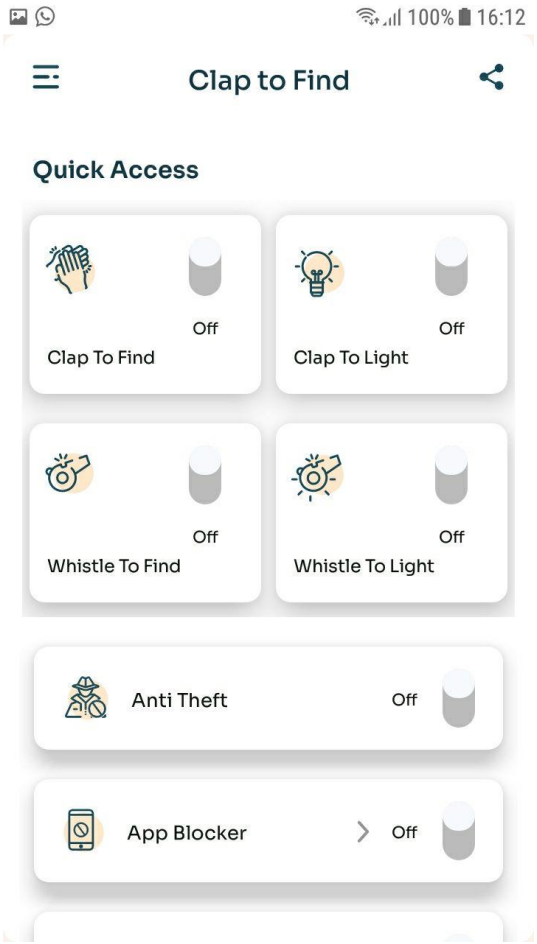

Рисунок 1.3 – Застосунок “Phone Finder by Clap”

1.5 Технології для реалізації функцій пошуку телефону

1.5.1 Виявлення звуків

Обробка аудіосигналів може виконуватися нативними методами та за допомогою React Native бібліотек для обробки звуку [19].

В нашому випадку ми будемо дотримуватись нативної реалізації, через те, що нативними методами легше буде настроїти функції обробки аудіосигналів в фоновому режимі.

Виявлення звуків у додатку реалізовано за допомогою нативного коду на Java для Android. Для цього використовуються AudioRecord та MediaRecorder для отримання та обробки аудіосигналів [2].

1.5.2 Використання вібрації та ліхтарика

Нативні API:

– android: використовується клас Vibrator для керування вібрацією та реалізації різних патернів [15]. А для керування ліхтариком застосовується клас CameraManager [14];
– iOS: застосовується клас UIImpactFeedbackGenerator для вібрації та клас AVCaptureDevice – для контролю ліхтарика.

React Native бібліотеки:

– react-native-haptic-feedback: забезпечує зручний інтерфейс для реалізації вібрацій на обох платформах;
– react-native-torch: ця бібліотека забезпечує простий інтерфейс для роботи з ліхтариком.

1.5.3 Інші корисні бібліотеки та інструменти

Також у нагоді можуть статися react-native-permissions – який допомагає керувати системними дозволами для використання мікрофона, ліхтарика та інших функцій [12], redux та redux-persis – для управління станом додатку та збереження даних між сеансами [9], @react-native-firebase/app – інтеграція з Firebase для аналітики, push-повідомлень та баз даних [7].

1.5.4 Інтеграція з рекламними сервісами

Для монетизації застосунку інтегровані рекламні сервіси. Такі як react-native-google-mobile-ads – використовується для інтеграції рекламних банерів та reward-реклами від Google AdMob, що дозволяє монетизувати додаток [10].

Використання сучасних технологій для реалізації функцій пошуку

телефону забезпечує високу точність розпізнавання звуків, ефективне керування вібрацією та ліхтариком, а також інтеграцію з рекламними сервісами для монетизації. Реалізація таких функцій у мобільному застосунку дозволяє користувачам швидко та зручно знаходити свої пристрої за допомогою хлопків.

2 ПЛАНУВАННЯ ТА РЕАЛІЗАЦІЯ

Цей розділ присвячений процесу планування та розробки мобільного застосунку для пошуку телефону за допомогою хлопків. Ми детально розглянемо архітектуру застосунку та реалізацію основних функцій.

2.1 Архітектура мобільного застосунку

Архітектура мобільного застосунку визначає його загальну структуру, компоненти та їх взаємодію. Застосунок для пошуку телефону за допомогою хлопків має наступну архітектуру.

Екран завантаження: цей екран відображає open app рекламу під час старту застосунку. Він призначений для забезпечення користувачів інформацією та заробітку за рахунок реклами. Після завершення показу реклами користувач автоматично переходить до наступного екрану.

Екран вибору мови: відображається при першому запуску застосунку. Користувач має можливість обрати бажану мову інтерфейсу. Використовується бібліотека react-native-i18n для забезпечення багатомовності.

Карусель з інструкціями: складається з трьох слайдів, що пояснюють користувачу, як працює застосунок. Перший слайд розповідає про функцію пошуку телефону за допомогою хлопків, другий – про використання ліхтарика, третій – про можливість знаходити телефон. Для реалізації каруселі використовується бібліотека react-native-snap-carousel.

Головна сторінка: основний інтерфейс для керування функціями пошуку телефону. Включає кнопки для увімкнення пошуку по хлопку, налаштування звуків, активації вібрації та ліхтарика.

Бокове меню: використовується для навігації по додатку. Включає

пункти для доступу до налаштувань, вибору звуків, перегляду історії використання, відгуків та інформації про застосунок. Реалізовано з використанням бібліотеки @react-navigation/drawer [8].

Екран для детального налаштування чутливості мікрофона, гучності звуку, типів вібрації (наприклад, «Сильна вібрація», «Сердцебиття») та режимів ліхтарика (наприклад, «Режим диско», «SOS»). Використовуються компоненти з бібліотеки react-native-paper.

Екрани, на яких користувач може обирати звуки для сигналізації. Це можуть бути як попередньо завантажені звукові ефекти, так і власні звуки користувача. Для завантаження користувацьких звуків використовується react-native-document-picker.

Архітектура застосунку базується на принципах модульності та повторного використання коду. Це забезпечує легкість у додаванні нових функцій і підтримці застосунку в майбутньому. Кожен екран і модуль розроблено як окремий компонент, що забезпечує гнучкість і простоту в управлінні проєктом.

2.2 Реалізація основних та другорядних функцій

У цьому розділі ми розглянемо реалізацію основних функцій мобільного застосунку для пошуку телефону за допомогою хлопків. Почнемо з того, як працює найголовніший функціонал, який виконує головну функцію застосунку – пошук телефона.

2.2.1 Обробка аудіосигналу та знаходження телефону

Для того, щоб застосунок «зрозумів», що його шукають, після увімкнення пошуку на головній сторінці, він починає прослуховувати та

аналізувати амплітуду звуків навколо. Коли амплітуда досягає необхідного рівня, який дорівнює середньостатистичному хлопку, застосунок вмикає обраний користувачем звуковий сигнал, вібрацію та блимання ліхтариком, якщо ці функції активовані. Усі ці функції повинні працювати як при увімкненому застосунку, так і у фоновому режимі.

Реалізувати це засобами React Native неможливо, оскільки він погано підтримує функції фонової роботи. Тому ці функції реалізовані за допомогою нативних модулів на Java [6]. Було створено три модулі: ClapDetectorModule, FlashlightsModes, і VibrationModes.

Модуль ClapDetectorModule передає налаштування, такі як поточний аудіофайл, режим роботи ліхтарика та вібрації, і аналізує вхідне аудіо з мікрофона. Головне виконується в методі recordAudio(), який призначен для запису аудіо та виявлення звуку хлопка. Ось як він працює покроково.

Спочатку створюється об’єкт MediaRecorder. Встановлюється джерело аудіо – мікрофон пристрою. Формат виводу аудіо визначається як THREE_GPP, а аудіо кодується за допомогою AMR_NB. Записаний аудіофайл буде збережений за вказаним шляхом [13] (див. рис. 2.1).

```
recorder = new MediaRecorder();
recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
recorder.setOutputFile("/data/data/" +
getReactApplicationContext().getPackageName() + "/music.3gp");
```

Рисунок 2.1 – Ініціалізація рекордера

Встановлюються початкова амплітуда, кінцева амплітуда та додаткова амплітуда, яка залежить від чутливості (sensetive). Порогове значення амплітуди обчислюється на основі додаткової амплітуди.

```
try {
   recorder.prepare();
   recorder.start();
   startAmplitude = recorder.getMaxAmplitude();
} catch (IOException e) {
   e.printStackTrace();
}
```

Рисунок 2.2 – Налаштування параметрів для виявлення хлопка

Рекордер готується до роботи та починає запис. Початкова амплітуда записується (див. рис. 2.3).

```
try {
   recorder.prepare();
   recorder.start();
   startAmplitude = recorder.getMaxAmplitude();
} catch (IOException e) {
   e.printStackTrace();
}
```

Рисунок 2.3 – Підготовка та запуск рекордера

У циклі (див. рис. 2.4) постійно перевіряється амплітуда звуку. Якщо вона перевищує порогове значення, вважається, що хлопок виявлено. Після цього:

- збільшується лічильник виявлених хлопків;
- якщо доступ до ліхтарика дозволений, він вмикається;
- якщо доступ до вібрації дозволений, пристрій починає вібрувати;
- викликається метод promise.invoke("Clap detected").

```
do {
   waitSome();
   finishAmplitude = recorder.getMaxAmplitude();
   if (finishAmplitude >= amplitudeThreshold) {
      clapDetectedNumber++;
      if(IsTorchAccess){
         flash.handler.post(flash.runnableFlash);
      }
      if(IsVibroAccess){
         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            mIntent.putExtra("VibroMode", vibroMode);
            mIntent.putExtra("isClose", "false");
            myReactContext.startForegroundService(mIntent);
         }
      }
      promise.invoke("Clap detected ");
   }
} while(counterMax > clapDetectedNumber && !isDecline);
```

Рисунок 2.4 – Цикл для виявлення хлопка

Після виходу з циклу викликається метод done(), який завершує роботу рекордера.

2.2.2 Інтернаціоналізація мобільного застосунку

Інтернаціоналізація мобільного застосунку є критичним аспектом розробки, що дозволяє зробити продукт доступним для широкої аудиторії, яка розмовляє різними мовами. В нашому випадку, застосунок створений з використанням React Native, забезпечує переклад інтерфейсу та контенту на

десять різних мов, що охоплюють основні мовні групи світу [11]. Для цього ми використовуємо бібліотеку react-native-i18n та асинхронне зберігання налаштувань користувача з допомогою @react-native-async-storage/async-storage [16] (див. рис. 2.5).

```
import I18n from 'react-native-i18n';
import AsyncStorage from '@react-native-async-storage/async-storage';
import en from './locales/en';
import arabic from './locales/arabic';
…
import ru from './locales/ua’;

const languageDetector = async () => {
  const preferredLang = await AsyncStorage.getItem('@lang');
  if (preferredLang) {
    I18n.locale = preferredLang;
  }
};

I18n.fallbacks = true;
languageDetector();
I18n.translations = {en, ua, ar: arabic, es: esp, fr: french, de: german, hi: hindi,
id: indonesian, ko: korean, pt: portuguese,};
export default I18n;
```

Рисунок 2.5 – Файл Ініціалізації

Опишемо процес інтернаціоналізації.

Визначення мов: у застосунку підтримуються наступні мови: англійська, арабська, іспанська, французька, німецька, хінді, індонезійська, корейська, португальська та українська. Це покриває більшість популярних мов, які

використовуються в різних регіонах світу.

Завантаження перекладів: усі переклади зберігаються у файлах всередині каталогу locales. Кожен файл містить ключ-значення пари, де ключі відповідають різним текстовим елементам в застосунку, а значення – перекладені тексти.

Визначення пріоритетної мови: для того, щоб зберегти налаштування користувача щодо обраної мови, використовується асинхронне зберігання. При запуску застосунку викликається функція languageDetector, яка зчитує збережену мову з локального сховища та встановлює її як поточну мову інтерфейсу.

Резервне відображення: якщо переклад для обраної мови відсутній, застосунок автоматично переключається на англійську мову як резервну. Це гарантує, що користувач завжди матиме доступ до зрозумілого інтерфейсу, навіть якщо обраний переклад ще не завершено.

Імплементація: всі переклади та логіка вибору мови імплементуються через бібліотеку react-native-i18n, що забезпечує зручну та гнучку реалізацію інтернаціоналізації.

Завдяки цьому підходу, наш мобільний застосунок готовий до використання у різних мовних середовищах, забезпечуючи користувачам зрозумілий та зручний інтерфейс незалежно від їхньої рідної мови.

2.2.3 Керування станом

У нашому React Native застосунку використовується Redux для управління станом. Це дозволяє централізовано зберігати та керувати станом застосунку, забезпечуючи передбачуваність та легкість у масштабуванні та підтримці коду [9]. У цьому проєкті використовується redux-persist для збереження стану між сесіями користувача, зберігаючи його у AsyncStorage [18].

Основний редусер застосунку складається з таких полів:

– currentSong: поточна обрана пісня;
– count_of_rewards: кількість обов’язкових реклам для винагород;
– blockAddOn24h: об’єкт, що містить інформацію про блокування реклами на 24 години;
– settings: об’єкт налаштувань, що включає чутливість, гучність, вібрацію, спалах, та інші параметри;
– customSongsList: список кастомних пісень користувача;
– blocked: об’єкт, що містить інформацію про заблоковані елементи, такі як стандартні звуки, вібрації тощо, які можна розблокувани за перегляд реклами.

Спочатку ми визначаємо action у файлі actions.ts. Action setCurrentMusic має тип SET_CURRENT_MUSIC та payload, який містить нову пісню (див. рис. 2.6).

```
import actionsTypes from "./actionsTypes";

export const setCurrentMusic = (value: string) => ({
   type: actionsTypes.SET_CURRENT_MUSIC,
   payload: value
});
```

Рисунок 2.6 – Створення action

У React компоненті ми викликаємо цей action для зміни поточної пісні. Компонент MusicPlayer підключається до Redux стору. Він отримує поточну пісню з state через mapStateToProps і функцію setCurrentMusic через mapDispatchToProps. Коли користувач натискає кнопку “Change Song”, викликається функція handleChangeSong, яка викликає setCurrentMusic з новою назвою пісні. (див. рис. 2.7).

```
import React from 'react';
import { connect } from 'react-redux';
import { setCurrentMusic } from '../actions/actions';

const MusicPlayer = ({ currentSong, setCurrentMusic }) => {
  const handleChangeSong = (newSong) => {
    setCurrentMusic(newSong);
  };

  return (
    <div>
      <h1>Current Song: {currentSong}</h1>
      <button onClick={() => handleChangeSong('newSongName')}>Change
Song</button>
    </div>
  );
};

const mapStateToProps = (state) => ({
  currentSong: state.mainFunctionalReduser.currentSong,
});

const mapDispatchToProps = {
  setCurrentMusic,
};

export default connect(mapStateToProps, mapDispatchToProps)(MusicPlayer);
```

Рисунок 2.7 – Використання action у компоненті

Коли action SET_CURRENT_MUSIC диспатчиться, редусер mainFunctionalReduser отримує його і оновлює поле currentSong у state відповідно до значення з payload. (див. рис. 2.8). Таким чином, ми бачимо

повний цикл життя action у Redux: від його створення до використання у компоненті та обробки у редусері для оновлення стану застосунку.

```
import types from '../actions/actionsTypes';

const initialState = {
  currentSong: 'standard',
  // інші поля
};

const mainFunctionalReduser = (state = initialState, actions) => {
  switch (actions.type) {
    case types.SET_CURRENT_MUSIC:
      return { ...state, currentSong: actions.payload };
    // інші case
    default:
      return state;
  }
};

export default mainFunctionalReduser;
```

Рисунок 2.8 – Обробка action у редусері

2.2.4 Опис роботи та роль React Navigation у проєкті

React Navigation – це бібліотека для управління навігацією у React Native застосунках. Вона забезпечує простий та потужний інструментарій для створення багатосторінкових застосунків з різноманітними типами навігації, такими як стекова, табова та бокова навігація [8]:

– react navigation дозволяє легко створювати навігаційну структуру

застосунку за допомогою компонентів навігації. Основні компоненти включають;

– stack navigator: використовується для управління стеком екранів, що дозволяє переходити вперед і назад між екранами;
– drawer navigator: реалізує бокову панель навігації, яка дозволяє користувачам переходити між основними розділами застосунку;
– tab navigator: створює таби для навігації між різними екранами, що корисно для організації контенту в застосунку.

У нашому застосунку “Find my phone by Clap” React Navigation використовується для управління навігацією між різними екранами (див. рис. 2.9).

```
const Router = ({initialRoute}) => {
 return (
  <Provider store={store}>
   <PersistGate loading={null} persistor={persistor}>
    <NavigationContainer theme={MyTheme}>
     <Drawer.Navigator
      initialRouteName={"MainMenu"}
      drawerContent={props => <CustomDrawerContent {...props} />}>
      <Drawer.Screen name="MainMenu" component={MainMenu} />
      <Drawer.Screen name="ChangeLang" component={ChangeLang} />
      <Drawer.Screen name="HowItWork" component={HowItWork} />
      <Drawer.Screen name="VIP" component={VIP} />
      <Drawer.Screen name="MicroPermissionScreen" component={PermissionScreen} />
      <Drawer.Screen
       name="soundOfFrolic"
       component={SoundFor}
       options={{title: 'soundOfFrolic'}}
      />
     </Drawer.Navigator>
    </NavigationContainer>
   </PersistGate>
  </Provider>
 );
};
```

Рисунок 2.9 – Приклад реалізації React Navigation

Основні функції включають:

– головна сторінка: доступ до основних функцій застосунку, таких як увімкнення пошуку по хлопку та доступ до налаштувань;
– екрани налаштувань: налаштування чутливості мікрофона, гучності звуку, типів вібрації та режимів ліхтарика;
– бокове меню: дозволяє користувачам переходити між різними розділами застосунку, такими як «Головна», «Вибір звуків», «Історія використання» та «Відгуки».

Навігація в застосунку реалізована за допомогою Drawer.Navigator, що забезпечує бокове меню для переходу між основними розділами застосунку. Наведений нижче приклад демонструє, як це реалізовано у нашому проєкті.

У цьому прикладі ми використовуємо Drawer.Navigator для створення бокового меню з різними екранами, такими як “MainMenu”, “ChangeLang”, “HowItWork”, “Settings” та інші. Кожен екран має свої власні налаштування та компоненти, що дозволяє гнучко налаштовувати навігацію в застосунку.

Бібліотека React Navigation забезпечує плавний та інтуїтивно зрозумілий користувацький досвід, дозволяючи легко орієнтуватися в застосунку. Вона також підтримує глибоку інтеграцію з іншими бібліотеками та нативними модулями, що дозволяє створювати високоякісні та функціональні мобільні застосунки.

3 ТЕСТУВАННЯ, МОНЕТИЗАЦІЯ ТА ОГЛЯД ГОТОВОГО ПРОДУКТУ

3.1 Тестування та налагодження застосунку

Тестування є важливим етапом розробки програмного забезпечення, що забезпечує якість і надійність застосунку. У цьому розділі ми розглянемо методологію тестування, інструменти, що використовувалися, результати тестування та виправлення помилок.

Для забезпечення якості застосунку використовувалися різні методи тестування [17].

Функціональне тестування – перевірка основних функцій застосунку, таких як виявлення хлопків, керування вібрацією і ліхтариком, відтворення звуків. Це дозволяє переконатися, що всі функції працюють згідно з вимогами.

Тестування користувацького інтерфейсу (UI) – оцінка зручності використання застосунку, відповідності дизайну та правильності відображення елементів інтерфейсу на різних пристроях і розмірах екранів.

Інтеграційне тестування – перевірка взаємодії між різними компонентами застосунку, включаючи нативні модулі та бібліотеки React Native.

Тестування продуктивності – оцінка швидкості роботи застосунку, час відгуку на дії користувача, ефективність використання ресурсів пристрою.

Регресійне тестування – перевірка, що нові зміни та виправлення не вплинули на вже існуючі функції [18].

Для тестування застосунку використовувалися такі інструменти:

– jest: основний фреймворк для тестування JavaScript, який використовується для написання юніт-тестів для компонентів React Native;
– react native testing library: бібліотека для тестування компонентів

React Native, що забезпечує легкість у написанні та підтримці тестів;

– detox: інструмент для end-to-end тестування мобільних застосунків, який дозволяє автоматизувати інтеграційні та UI-тести;
– android studio та Xcode: використовувалися для запуску емуляторів, профілювання продуктивності та налагодження нативного коду.

Тестування застосунку виявило, що були проблеми з точністю розпізнавання хлопків при високому рівні фонових шумів, яке було виправлено шляхом налаштування порогів амплітуди та використання фільтрації шумів, некоректне вимкнення аудіозапису після виявлення хлопка, помилку в режимах роботи ліхтарика при зміні режимів, затримку в активації вібрації та деякі недоліки відображення елементів.

Також було виявлено високе споживання ресурсів при тривалому використанні застосунку у фоновому режимі. Оптимізовано процеси для зменшення навантаження на процесор і зниження споживання батареї.

3.2 Монетизація

Монетизація мобільного застосунку реалізована через інтеграцію рекламних сервісів. Це дозволяє забезпечити безкоштовне використання застосунку для користувачів, одночасно генеруючи дохід для розробників. У застосунку використовуються кілька типів реклами.

Рекламні банери – це статичні оголошення, які відображаються в нижній частині екрана. Вони присутні на таких екранах:

– головне меню;
– екран вибору звуків;
– екран жартівливих звуків.

Існують два види банерів: звичайні банери та нативні банери. Останні інтегруються у дизайн застосунку і гармонійно вписуються в інтерфейс, що робить їх менш нав’язливими для користувачів.

Reward-реклама – це відеооголошення, які користувач може переглянути для отримання певних бонусів у застосунку. Вони відображаються у таких випадках:

– для доступу до додаткових звуків у розділах “Sound for search” та “Prank sounds”;
– при активації вібрації та ліхтарика на головному екрані та у налаштуваннях;
– користувач може переглянути два відеооголошення підряд, щоб повністю вимкнути рекламу на певний час (екран “No ads for reward”).

Reward-реклама стимулює користувачів до взаємодії з рекламою, надаючи їм відчутні переваги, що позитивно впливає на їхній користувацький досвід.

Інтерстиціальні відео – це повноекранні оголошення, що відображаються під час переходу між різними екранами застосунку:

– після завершення інструктажу на перших трьох екранах “How it works?”;
– при натисканні кнопки «Назад» на будь-якому екрані, включаючи “Sound for Search”, “Prank Sounds”, “Settings”, “How it works (manual)”, “Language” (при переході з налаштувань).

Це допомагає максимізувати дохід від реклами, не завдаючи значного дискомфорту користувачам.

А також Open App реклама – рекламне оголошення, яке відображається при запуску застосунку. Це забезпечує максимальне охоплення аудиторії, оскільки кожен користувач бачить це оголошення при кожному відкритті застосунку. Цей тип реклами ефективний для генерації високих доходів.

Використання різних форматів реклами дозволяє ефективно монетизувати застосунок, забезпечуючи стабільний дохід для розробників і зберігаючи позитивний користувацький досвід.

3.3 Опис закінченого продукту

Застосунок “Find my phone by clap” вирізняється своїм зручним і сучасним візуальним дизайном, який поєднує темну кольорову схему з фіолетовими акцентами.

Основний екран застосунку містить іконку руки, що плескає, та кнопку «Старт» (див. рис. 3.1), що дозволяє користувачам легко розпочати користування.

Інтерфейс застосунку забезпечує чітке розділення між різними налаштуваннями та опціями, що спрощує навігацію та налаштування.

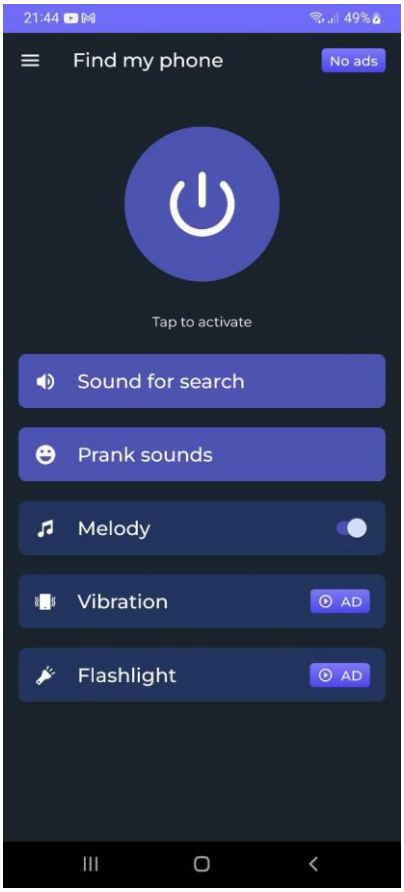

Рисунок 3.1 – Основний екран застосунку

Важливою складовою є екран доступу до мікрофону та повідомлень (див. рис. 3.2), який відображає повідомлення про необхідність дозволу на використання мікрофону для правильної роботи застосунку.

Застосунок пропонує широкий спектр звукових налаштувань, що дозволяє користувачам вибирати різні рингтони, налаштовувати рівень гучності та активувати вібрацію для сповіщень.

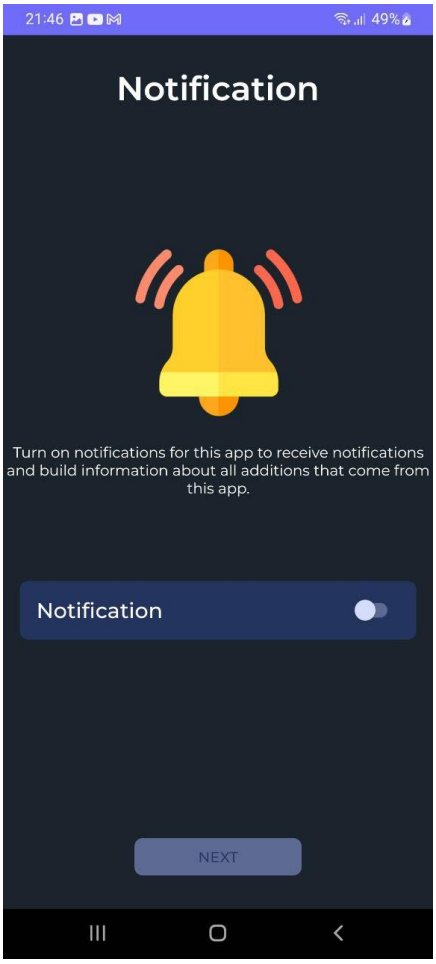

Рисунок 3.2 – Екран доступу до повідомлень

Додаткові функціональні можливості включають використання фонарика, який можна увімкнути за допомогою плеску, та видачу звукового сигналу, коли телефон знайдено.

Користувачі також можуть оцінити застосунок за допомогою спеціального вікна для оцінок і відгуків, що сприяє покращенню його функціональності та зручності.

Крім того, користувачі можуть обирати або додавати свої улюблені пісні для сповіщень, що дозволяє персоналізувати застосунок відповідно до власних вподобань (див. рис. 3.3).

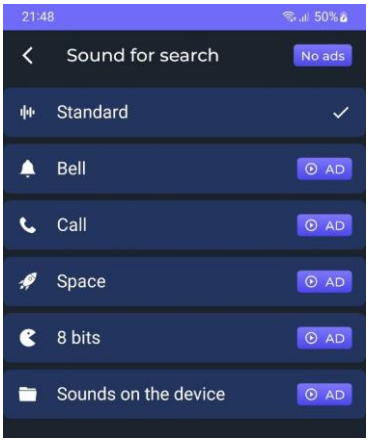

Рисунок 3.3 – Екран зміни пісні для сповіщень

Для користувачів, які бажають уникнути реклами (див. рис. 3.4), застосунок пропонує можливість вимкнення реклами за додаткову плату. Також наявні інші платні функції, що надають розширені можливості використання.

Загалом, застосунок “Find my phone by clap” має інтуїтивно зрозумілий інтерфейс, який забезпечує легкий доступ до всіх необхідних функцій, роблячи його ефективним інструментом для пошуку телефону.

Рисунок 3.4 – Приклад реклами у застосунку

3.4 Огляд конкурентних застосунків

Розробка мобільного застосунку для пошуку телефону за допомогою хлопків має багато конкурентів на ринку, кожен з яких має свої особливості та функціонал. Одним із основних конкурентів є застосунок “Clap to Find My Phone”. Цей застосунок пропонує просту та інтуїтивно зрозумілу функціональність з виявлення хлопків для активації сигналу. Він дозволяє налаштувати чутливість мікрофона, підтримує вібрацію та ліхтарик, але має обмежену кількість звукових сигналів та відсутність розширених налаштувань.

Іншим конкурентом є “Whistle Phone Finder” (див. рис. 3.5), який пропонує виявлення свисту для активації сигналу. Цей застосунок підтримує різні звукові сигнали та налаштування чутливості, але менш точний у розпізнаванні свисту у шумному середовищі і не підтримує функції вібрації та ліхтарика.

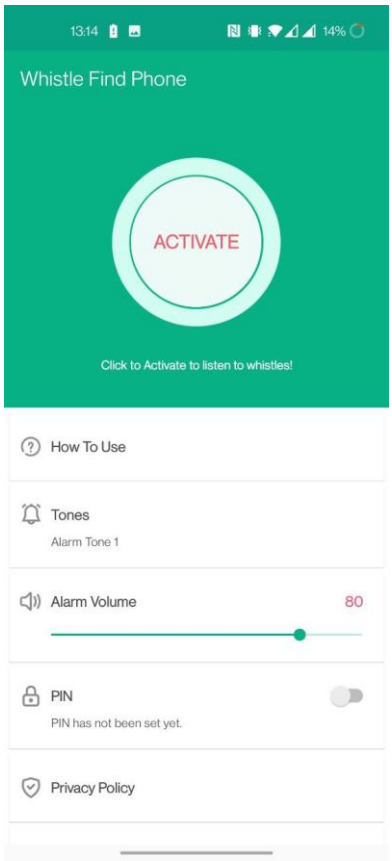

Рисунок 3.5 – Застосунок “Whistle Phone Finder”

Застосунок “Find My Phone by Clap & Whistle” (див. рис. 3.6) поєднує функції виявлення як хлопків, так і свисту. Він дозволяє налаштовувати чутливість мікрофона, підтримує вібрацію та ліхтарик, але може бути складнішим у налаштуванні та мати більше помилкових спрацьовувань у шумному середовищі.

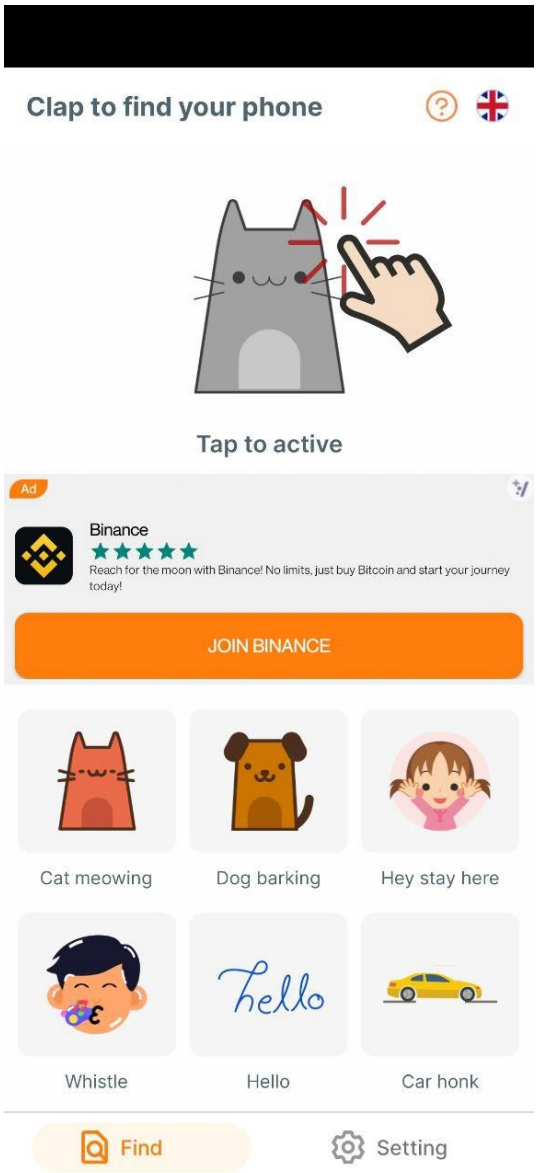

Рисунок 3.6 – Застосунок “Find My Phone by Clap & Whistle”

Наш застосунок “Find my phone by Clap” має переваги у порівнянні з конкурентами завдяки інтуїтивно зрозумілому інтерфейсу, широкому набору функцій, включаючи різноманітні звукові сигнали, вібрацію та ліхтарик, а також розширеним налаштуванням чутливості мікрофона.

Важливою перевагою є можливість стабільної роботи у фоновому режимі завдяки використанню нативних модулів на Java.

Хоча наш застосунок може бути складнішим у налаштуванні та вимагати більше ресурсів пристрою, його багатофункціональність і зручність використання роблять його привабливим для користувачів. Інтеграція різних типів реклами забезпечує стабільний дохід для розробників без негативного впливу на користувацький досвід. Таким чином, “Find my phone by Clap” є конкурентоспроможним продуктом на ринку мобільних застосунків для пошуку телефону.

ВИСНОВКИ

Розробка мобільного застосунку для пошуку телефону за допомогою хлопків виявилася цікавою та корисною задачею, яка вимагала використання сучасних технологій та підходів. Проєкт включав різні аспекти, такі як архітектура застосунку, реалізація основних функцій, тестування та налагодження, а також монетизація.

В результаті виконаної роботи можна зробити наступні висновки.

Архітектура застосунку: було створено чітку та модульну архітектуру, яка включає основні екрани, такі як екрани завантаження, вибору мови, інструкцій, головну сторінку, бокове меню, налаштування та екрани вибору звуків. Це забезпечило зручність використання та легкість підтримки застосунку.

Реалізація основних функцій: для реалізації функцій виявлення хлопків, керування вібрацією та ліхтариком використовувалися нативні модулі на Java. Це дозволило забезпечити стабільну роботу застосунку як у фоновому режимі, так і при активному використанні.

Тестування та налагодження: було проведено детальне тестування застосунку, що включало функціональне, інтеграційне, регресійне та продуктивне тестування. Виявлені помилки були виправлені, що забезпечило високу якість та надійність застосунку.

Монетизація: застосунок було успішно монетизовано через інтеграцію різних типів реклами, включаючи рекламні банери, reward-рекламу, відео-рекламу між екранами та open app рекламу. Це дозволило забезпечити стабільний дохід для розробників, не погіршуючи користувацький досвід.

Проєкт показав, що використання сучасних технологій та підходів у розробці мобільних застосунків дозволяє створювати ефективні та зручні продукти, які відповідають потребам користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. React Native Documentation. URL: https://reactnative.dev/docs/getting-started (дата звернення: 09.01.2024).
2. Android Developers Documentation. URL: https://developer.android.com/docs (дата звернення: 12.01.2024).
3. Flutter documentation. URL: https://docs.flutter.dev/ (дата звернення: 23.01.2024).
4. What is Xamarin? URL: https://learn.microsoft.com/en-us/previous-versions/xamarin/get-started/what-is-xamarin (дата звернення: 07.02.2024).
5. NativeScript Documentation. URL: https://docs.nativescript.org/ (дата звернення: 12.02.2024).
6. Java SE Documentation. *Oracle*. URL: https://docs.oracle.com/en/java/ (дата звернення: 23.02.2024).
7. Firebase Documentation. *Google Firebase*. URL: https://firebase.google.com/docs (дата звернення: 01.03.2024).
8. React Navigation Documentation. URL: https://reactnavigation.org/docs/getting-started (дата звернення: 05.03.2024).
9. Redux Documentation. URL: https://redux.js.org/introduction/getting-started (дата звернення: 04.03.2024).
10. AdMob Documentation. Google AdMob. URL: https://developers.google.com/admob/android/quick-start (дата звернення: 17.04.2024).
11. React-native-i18n Documentation. URL: https://github.com/AlexanderZaytsev/react-native-i18n/blob/master/README.md (дата звернення: 28.04.2024).
12. React Native Permissions Documentation. URL: https://github.com/zoontek/react-native-permissions (дата звернення: 23.04.2024).

13. Java MediaRecorder API Guide. URL: https://developer.android.com/reference/android/media/MediaRecorder (дата звернення: 03.05.2024).
14. Android Camera2 API Guide. URL: https://developer.android.com/reference/android/hardware/camera2/package-summary (дата звернення: 07.05.2024).
15. Vibration API Guide. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/Vibration_API (дата звернення: 07.05.2024).
16. Async Storage for React Native Documentation. URL: https://react-native-async-storage.github.io/async-storage/docs/install/ (дата звернення: 07.05.2024).
17. What is software testing? URL: https://www.ibm.com/topics/software-testing (дата звернення: 08.05.2024).
18. The Importance Of Software Testing: All You Need To Know. URL: https://www.verbat.com/blog/the-importance-of-software-testing-all-you-need-to-know/ (дата звернення: 09.05.2024).
19. React Native Sound Documentation. URL: https://github.com/zmxv/react-native-sound (дата звернення: 05.02.2024).