ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему: **«РОЗРОБКА ANDROID ДОДАТКУ ДЛЯ ШВИДКОГО ВИВЧЕННЯ ІНФОРМАЦІЇ МОВОЮ JS»**

Виконав: студент 4 курсу, групи ПД–41

спеціальності

121-Інженерія програмного забезпечення

(шифр і назва спеціальності)

Пащенко В. Ю.

(прізвище та ініціали)

Керівник Дібрівний О. А.

(прізвище та ініціали)

Рецензент _____

(прізвище та ініціали)

Нормоконтроль _____

(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Телекомунікацій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Напрям підготовки - 121- Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

_____О. В. Негоденко

“____” _____2021 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Пащенку Владиславу Юрійовичу

(прізвище, ім’я, по батькові)

1.Тема роботи: «Тема роботи за наказом» «Розробка Android додатку для швидкого вивчення інформації мовою JS»

Керівник роботи Дібрівний Олесь Андрійович, доктор філософії ,

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “12” березня 2021 року №65.

2. Строк подання студентом роботи 1.06.2021

3. Вхідні дані до роботи:

Android Studio, VS Code

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

4.1 Аналіз предметної області.

4.2 Методи реалізації програмного продукту.

4.3 Опис програмної реалізації.

4.4 Тестування та експлуатація.

5. Перелік демонстраційного матеріалу

1.Титульний слайд

2.Об'єкт, предмет та мета дослідження

3.Порівняльна характеристика аналогів

4.Технічні завдання

5.Програмні засоби реалізації

6.Опис програмної реалізації

7.Апробація результатів дослідження

8.Висновки

9. Дата видачі завдання 19.04.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір та вивчення джерел інформації	19.04.2021	Виконано
2	Аналіз предметної області	20.04.2020	Виконано
3	Розробка програмного продукту	24.04.2020	Виконано
4	Написання основної частини	26.04.2021	Виконано
5	Вступ, висновки, реферат	27.04.2021	Виконано
6	Розробка обов’язкових демонстраційних матеріалів	28.04.2021	Виконано
7	Попередній захист роботи		
8	Подання роботи в деканат	1.06.2021	

Студент _____ Пащенко В. Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дібрівний О. А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 47 с., 27 рис., 23 джерела.

ШВИДКЕ ВИВЧЕННЯ ІНФОРМАЦІЇ, МОБІЛЬНИЙ ДОДАТОК, ANDROID, JAVASCRIPT, REACT NATIVE, FIREBASE.

Об`єкт дослідження – процес вивчення інформації за допомогою мобільних додатків.

Предмет дослідження – застосування сучасних методів та засобів розробки програмного забезпечення для розробки Android додатку для швидкого вивчення інформації.

Мета дослідження – розробка Android додатку для швидкого вивчення інформації.

Методи дослідження – методи теорії інформації, методи теорії тестування, методи що базуються на використанні фреймворку React Native, платформи для розробки мобільних та веб додатків Firebase та мови програмування JavaScript.

Здійснено аналіз існуючих програмних продуктів для вивчення інформації.

На основі отриманих результатів аналізу розроблено універсальний мобільний додаток для швидкого вивчення інформації.

Додаток розроблено на мові програмування JavaScript з використанням фреймворку React Native та платформи для розробки мобільних та веб додатків Firebase.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API (англ. Application Programming Interface, API) – прикладний програмний інтерфейс.

APK (англ. Android Package) – формат архівних файлів-додатків для “Android”.

SDK (англ. software Development Kit) – набір із засобів розробки, утиліт і документації, який дозволяє програмістам створювати прикладні програми.

IDE (англ. Integrated development environment) – інтегроване середовище розробки.

NDK (англ. Native Development Kit) – необхідний набір інструментарію для розробки компонентів програмного забезпечення для платформи Android, який базується на C/C++ та інших мовах програмування.

ART (англ. Android Runtime) – середовище виконання Android-додатків.

AOT (англ. Ahead-of-Time) – метод компіляції перед виконанням.

JDK (англ. Ahead-of-Time) – комплект розробника додатків на мові Java.

BSD (англ. Berkeley Software Distribution) – назва кількох операційних систем сімейства UNIX.

GNU (англ. GNU's Not Unix) – вільна UNIX-подібна операційна система.

DOM (англ. Document Object Model) – специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами.

JSX (англ. JavaScript XML) – це розширення синтаксису JavaScript, яке дозволяє використовувати HTML-подібний синтаксис для опису структури інтерфейсу.

ES6 (англ. ECMAScript 6) – стандарт мови програмування, затверджений міжнародною організацією ECMA.

CSS (англ. Cascading Style Sheets) – це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду.

XML (англ. Extensible Markup Language) – стандарт побудови мов розмітки ієрархічно структурованих даних.

HTML (англ. HyperText Markup Language) – мова розмітки гіпертексту.

NoSQL (англ. non SQL, not only SQL) – база даних, яка забезпечує механізм зберігання та видобування даних відмінний від підходу таблиць-відношень в

реляційних базах даних.

CDN (англ. Content Delivery Network, Content Distribution Network) – географічно розподілена мережева інфраструктура, що дозволяє оптимізувати доправлення та розповсюдження контенту кінцевим користувачам в мережі Інтернет.

HTTPS (англ. Hypertext Transfer Protocol Secure) – розширення протоколу передачі гіпертексту.

SSL (англ. Secure Sockets Layer) – криптографічний протокол, який забезпечує встановлення безпечного з'єднання між клієнтом і сервером.

ВСТУП

На сьогоднішній день кількість користувачів смартфонів збільшуються в з великою швидкістю. Смартфони стали невід'ємною частиною нашого життя. Використання цих пристроїв зросло у різних секторах, і в тому числі і в освіті. Останнім часом мобільні додатки для навчання стають все більш популярними тому, шо користуватись ними надзвичайно зручно. Смартфон завжди під рукою, тому користуватись такими додатками можна будь-де та будь-коли. Також мобільні додатки часто використовують різноманітні методи для підвищення швидкості та ефективності навчання, які особливо добре працюють саме на мобільних платформах. Такими методами є використання ігрофікації, використання різних типів медіа контенту таких, як зображення, відео та аудіо для задіяння всіх типів пам'яті та використання інтервального повторення. Всі ці, та багато інших методів дозволяють зробити процес навчання більш цікавим та ефективним. В Наш час, саме мобільні платформи стають все більш поширеними, і дуже швидко розвиваються. Також розвиваються інструменти розробки. Це дозволяє розробникам мобільних додатків використовувати всі можливості платформи та реалізовувати будь-які ідеї. Оскільки навчальні додатки стають все більш популярними і мають великий потенціал розвитку розробка подібного додатку є досить актуальною. Тому, було вирішено створити додаток, що буде використовувати методи підвищення ефективності навчання, використовуватиме переваги мобільної платформи та надаватиме користувачам можливість вивчати будь-яку необхідну інформацію максимально швидко. Об`єкт дослідження – процес вивчення інформації за допомогою мобільних додатків. Предмет дослідження – застосування сучасних методів та засобів розробки програмного забезпечення для розробки Android додатку для швидкого вивчення інформації. Мета дослідження – розробка Android додатку для швидкого вивчення інформації. Методи дослідження – методи теорії інформації, методи теорії тестування, методи що базуються на використанні фреймворку React Native, платформи для розробки мобільних та веб додатків Firebase та мови програмування JavaScript.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Додаток повинен надавати користувачу можливість швидко вивчати інформацію за допомогою методу флеш-карток. Усі користувачі мають пройти етап реєстрації для подальшого зберігання інформації для вивчення, а також інформації про користувача, його статистики та прогресу в навчанні у базі даних. Користувач повинен мати можливість створювати власні набори флеш-карток з інформацією для вивчення та використовувати уже створені іншими користувачами набори для вивчення. Кожен елемент для вивчення повинен бути цифровою версією флеш-карток, а саме – складатися з питання, відповіді на це питання, з додаткової інформації або роз'яснення відповіді (якщо потрібно) та додаткових медіа файлів для полегшення запам'ятовування та підвищення швидкості вивчення. В якості медіа файлів повинні використовуватися зображення у форматах png та jpg, відео у форматі mp4 та аудіо у форматі mp3. Також додаток повинен мати функції автоматичної генерації аудіо за допомогою синтезу тексту в мовлення та автоматичної генерації медіа за допомогою пошуку зображень в інтернеті для підвищення зручності використання додатку. Вивчення інформації повинно складатися з двох етапів: вивчення і повторення. Під час етапу вивчення додаток повинен показувати користувачу елементи обраного користувачем набору та всю інформацію пов'язану з кожним елементом та перевіряти як добре користувач запам'ятав цю інформацію за допомогою автоматично згенерованих тестів. Для підвищення швидкості та ефективності вивчення інформації додаток повинен використовувати методи інтервального повторення та гейміфікації. Додаток повинен автоматично встановлювати інтервал повторення для кожної флеш-картки відповідно до того, як добре користувач запам'ятав її.

1.2 Огляд та аналіз існуючих аналогів

Перш ніж почати розробку програмного продукту необхідно провести аналіз аналогічних продуктів що мають схожий функціонал та визначити їх переваги та недоліки. В межах цієї дипломної роботи було проведено аналіз найбільш популярних додатків що мають схожий функціонал предметної області.

Серед усіх аналогів найбільш популярним є додаток AnkiDroid. Інтерфейс AnkiDroid показано на рисунку 1.1.

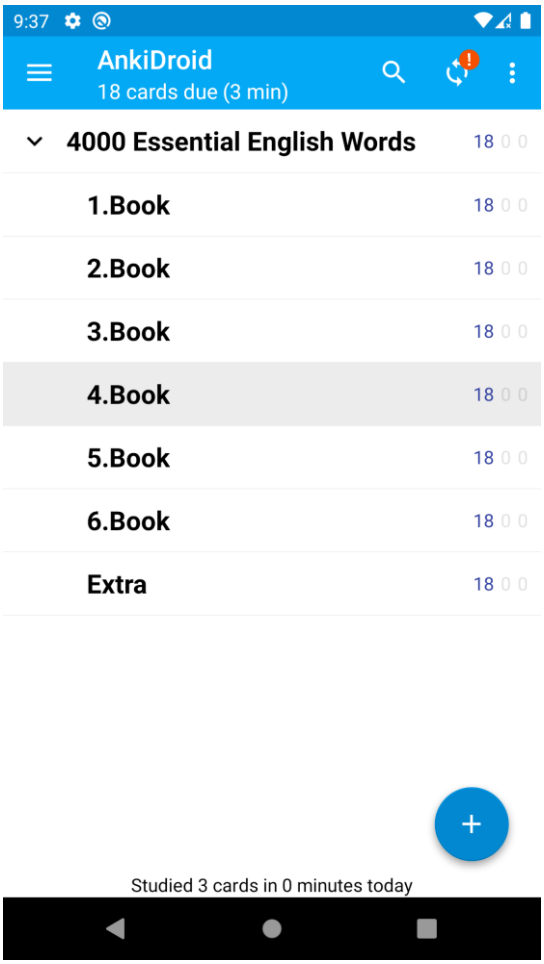


Рисунок 1.1 – Інтерфейс мобільного додатку AnkiDroid

Цей додаток є android версією додатку Anki. Додаток дозволяє користувачу ефективно навчатися за допомогою флеш-карток та методу інтервального повторення. Користувачі можуть завантажувати безкоштовні набори флеш-карток на будь-які теми що доступні багатьма мовами або створювати власні

набори флеш-карток. Загальний вигляд використання AnkiDroid показано на рисунку 1.2.

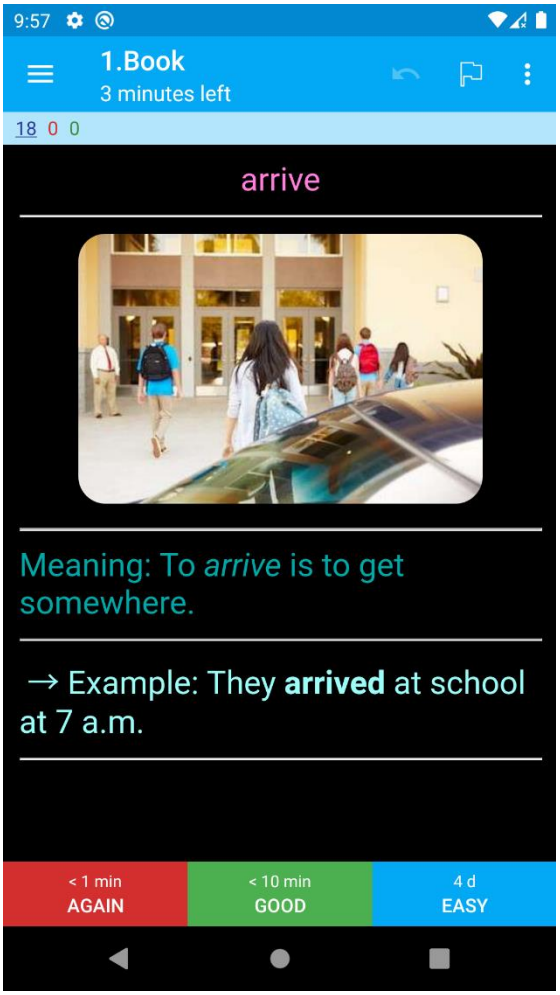


Рисунок 1.2 – Використання мобільного додатку AnkiDroid

Основні переваги додатку:

1. Підтримка різних видів матеріалів для флеш-карток, таких як текст, зображення та аудіо файли.
2. Використання методу інтервального повторення.
3. Можливість використання синтезу тексту в мовлення.
4. Велика кількість безкоштовних наборів флеш-карток.
5. Надання користувачу детальної статистики та можливості відслідковувати свій прогрес в навчанні.

Основні недоліки додатку:

1. Відсутня підтримка відео файлів в якості матеріалів для флеш-карток.
2. Відсутня можливість автоматичного пошуку відповідних медіа матеріалів для флеш-карток.
3. Завантаження наборів флеш-карток відбувається через веб версію додатку і займає дуже багато часу.

Наступном аналогом є додаток Memrise. Інтерфейс Memrise показано на рисунку 1.3.

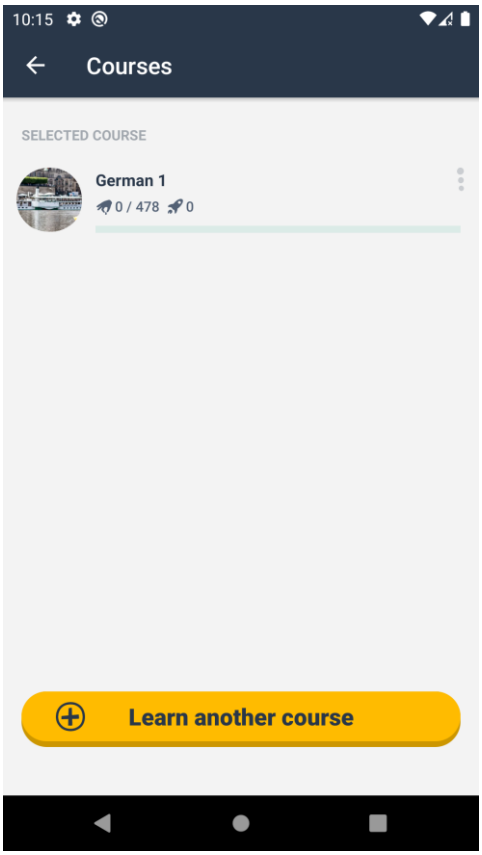


Рисунок 1.3 – Інтерфейс мобільного додатку Memrise

Memrise це додаток що переважно використовується для вивчення іноземних мов, але може використовуватися для вивчення будь-чого. Memrise також використовує флеш-картки та метод інтервального повторення але принцип використання додатку відрізняється від принципу використання аналогічних додатків. Замість звичайних наборів з флеш-картками додаток використовує курси, що в свою чергу поділяються на рівні, які складаються з флеш-карток.

Процес вивчення розділяється на два етапи: етап навчання та етап повторення. Під час етапу вивчення користувачу надається вся інформація що є на флеш-картці для вивчення, а під час етапу повторення за допомогою автоматично згенерованих тестів перевіряється прогресс користувача в навчанні. За результатами тестування також адаптується інтервал повторення. Додаток використовує методи ігрофікації для заохочення користувачів навчатися. Наприклад, після кожної сесії навчання або повторення користувач отримує бали і може змагатися з іншими користувачами, намагаючись набрати найбільшу кількість балів.

Основні переваги додатку:

1. Підтримка різних видів матеріалів для флеш-карток, таких як текст, зображення та аудіо файли.
2. Використання методу інтервального повторення та ігрофікації.
3. Синхронізація між різними пристроями.
4. Велика кількість курсів для вивчення.

Основні недоліки додатку:

1. При використанні android версії додатку відсутня можливість створювати власні курси та користуватись курсами інших користувачів.
2. При створенні власних курсів відсутня підтримка відео файлів в якості матеріалів для флеш-карток.
3. Відсутня можливість автоматичного пошуку відповідних медіа матеріалів для флеш-карток та синтезу тексту в мовлення.
4. Велика частина матеріалів курсів та функціоналу додатку доступа лише з платною підпискою. Загальний вигляд використання Memrise показано на рисунку 1.4.

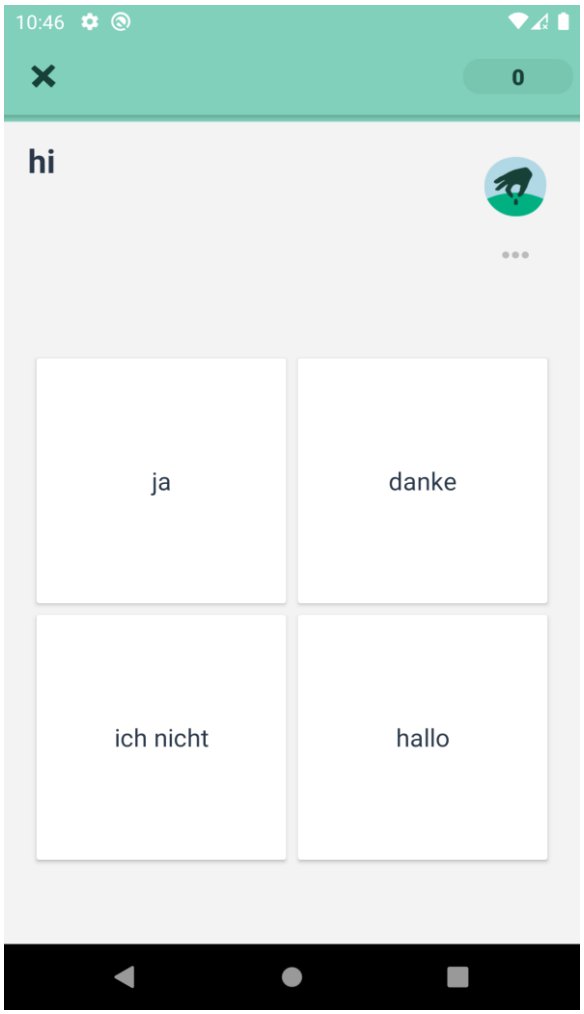


Рисунок 1.4 – Використання мобільного додатку Memrise

Наступном аналогом який варто розглянути є додаток Lexilize Flashcards. Додаток дозволяє користувачу ефективно навчатися та збільшувати свій словниковий запас. Lexilize Flashcards також використовує флеш-картки та метод інтервального повторення але може використовуватись тільки для вивчення іноземних мов. Так само як і в додатку Memrise процес вивчення інформації поділяється на етапи навчання та повторення однак після завершення сесії навчання чи повторення додаток не надає жодних результатів користувачу. Lexilize Flashcards надає мінімальну статистику навчання, але більша частина цієї статистики доступна лише з платною підпискою. Додаток підтримує тільки зображення в якості матеріалів для флеш-карток, але в ньому реалізована можливість автоматичного пошуку зображень в інтернеті. Також в додатку реалізована можливість автоматичного перекладу слів, але для використання цієї

можливості необхідно отримати API ключ перекладача Yandex який не працює в Україні. У Lexilize Flashcards є стартовий набір флеш-карток для кожної з підтримуваних мов, але кількість карток в цих наборах дуже мала, а можливість шукати і використовувати набори створені іншими користувачами відсутня. Інтерфейс Lexilize Flashcards показано на рисунку 1.5.

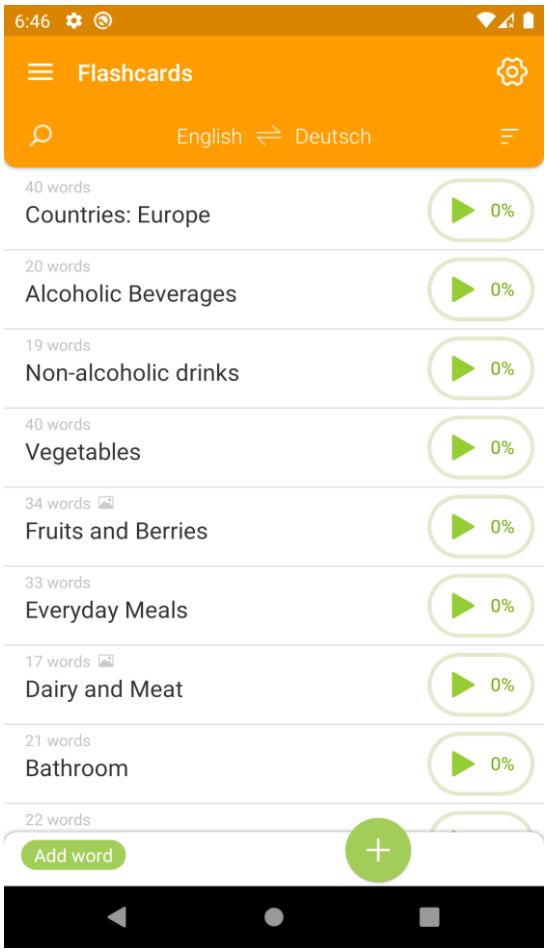


Рисунок 1.5 – Інтерфейс мобільного додатку Lexilize Flashcards

Основні переваги додатку:

1. Використання методу інтервального повторення.
2. Додаток працює без підключення до Інтернету.
3. Можливість використання синтезу тексту в мовлення.
4. Можливість автоматичного пошуку відповідних медіа матеріалів для флеш-карток.

Основні недоліки додатку:

1. Відсутня підтримка відео та аудіо файлів в якості матеріалів для флеш-карток.
2. Статистика доступна тільки з платною підпискою.
3. Дуже мала кількість флеш-карток в стартових наборах та відсутність можливості пошуку та використання наборів створених іншими користувачами.
4. Відсутня можливість синхронізації наборів флеш-карток між різними пристроями. Загальний вигляд використання Lexilize Flashcards показано на рисунку 1.6.

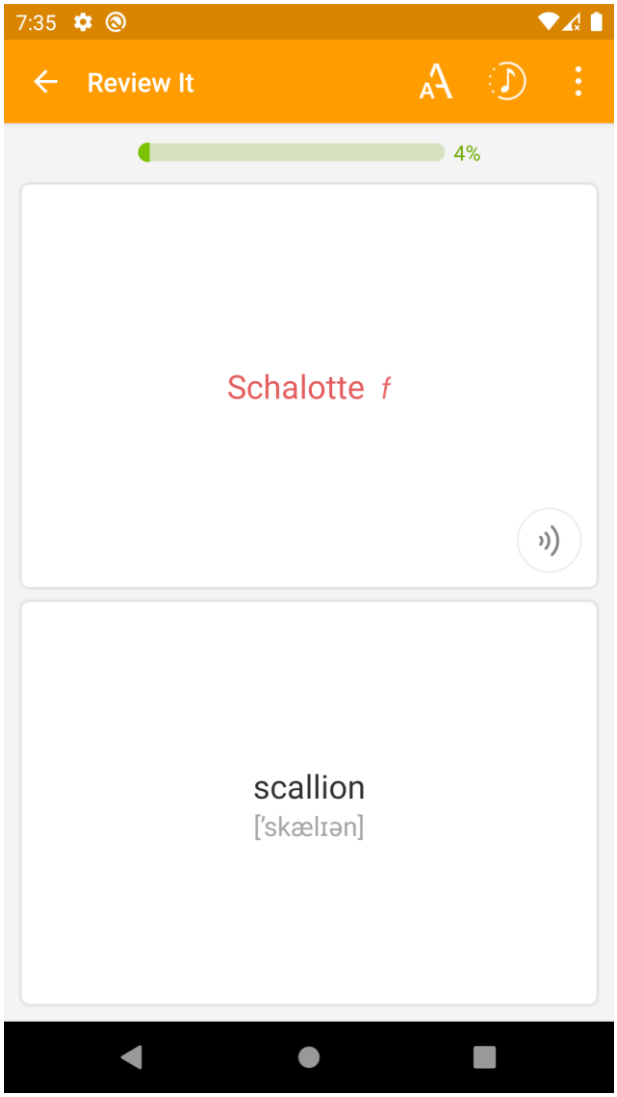


Рисунок 1.6 – Використання мобільного додатку Lexilize Flashcards

1.3 Висновки до розділу

В даному розділі було подано опис предметної області, розглянуто вимоги до додатку що розроблюється, та весь функціонал додатку, шо необхідно реалізувати. Також було проведено детальний аналіз усіх найбільш популярних існуючих аналогів додатку. Було розглянуто мобільні додатки для швидкого та ефективного вивчення інформації для операційної системи Android, що мають схожий функціонал предметної області і доступні для безуоштовного завантаження в магазині Android додатків Google Play Store. Для проведення аналізу було обрано три додатки з найбільшою кількістю постійних користувачів за статистикою магазина Google Play Store, а саме: AnkiDroid Flashcards, Memrise та Lexilize Flashcard. Було подано повний опис та детально проаналізовано весь функціонал та усі можливості наведених додатків, показано зовнішній вигляд користувацького інтерфейсу цих додатків і показані зображення з прикладами їх використання. Також для кожного з цих додатків було виділено і описано усі основні переваги та недоліки. В результаті було отримано детальну і повну характеристику кожного з проаналізованих додатків. Цю характеристику в подальшому було використано для формування вимог до основного функціоналу і зовнішнього вигляду користувацького інтерфейсу власного універсального додатку для швидкого і ефективного вивчення будь-якої інформації.

2 МЕТОДИ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТУ

2.1 Платформа Android

Android – це операційна система для сенсорних мобільних пристроїв, розроблена на основі модифікованої версії ядра Linux. Android розробляється консорціумом розробників Open Handset Alliance, і комерційно фінансується компанією Google. Вперше Android був представлений у листопаді 2007 року, а перший комерційний пристрій Android був запущений у вересні 2008 року.

Операційна система Android безкоштовна, а її код відкритий і в основному ліцензується за ліцензією Apache. Однак назва та логотип "Android" є товарними знаками компанії Google, які встановлюють стандарти щодо обмеження використання "несертифікованих" пристроїв за межами їх екосистеми для використання бренду Android. Комерційний логотип Android, який використовується Google із серпня 2019 року показано на рисунку 2.1.

Рисунок 2.1 – Комерційний логотип Android

Додатки для операційної системи Android використовують формат APK та зазвичай розповсюджуються через магазини додатків, такі як Google Play Store, Samsung Galaxy Store та Huawei AppGallery.

Android є найбільш популярною операційною системою у світі для смартфонів з 2011 року, а для планшетів - з 2013 року. Станом на травень 2017 року операційною системою користуються понад два мільярди активних користувачів і це робить Android системою з найбільшою базою користувачів серед усіх операційних систем. Станом на січень 2021 року в магазини додатків Google Play Store можна завантажити понад 3 мільйони додатків. Поточна стабільна версія – Android 11, випущена 8 вересня 2020 року.

Усі Android додатки розробляються за допомогою Android SDK. Для розробки зазвичай використовується мова програмування Kotlin, якій компанія Google віддає перевагу замість мови програмування Java. Однак Java також підтримується, так само як і мови програмування C та C++.

Android SDK включає в себе набір інструментів для розробки додатків таких як відлагоджувач, різноманітні бібліотеки, Android емулятор, Android Studio IDE, документація, приклади коду та навчальні посібники. Серед інших інструментів для розробки Android додатків також доступні Android NDK для додатків написаних мовами програмування C та C++ і різноманітні крос-платформні фреймворки для розробки мобільних додатків.

Ядро Android створено на основі ядра Linux. Зазвичай використовується версія ядра Linux 4.14, 4.19 або 5.4, але версія може відрізнятися на різних пристроях. Однак архітектура ядра Android суттєво відрізняється від ядра Linux так як компанія Google впровадила ряд змін, які включають дерева пристроїв, модуль “ashmem“ для анонімної спільної пам'яті, розподільник загального призначення “ION”, обробка нестачі пам'яті та функція управління живленням під назвою “wakelocks“.

Поверї ядра Linux є проміжне програмне забезпечення, бібліотеки та API, написані на мові програмування C, та прикладне програмне забезпечення, що працює на програмному фреймворку, що включає Java-сумісні бібліотеки. Розробка ядра Linux триває незалежно від інших проектів вихідного коду Android.

Починаючи з версії 5.0 Android використовує ART, що є середовищем виконання, яке використовує компіляцію AOT, щоб повністю скомпілювати байт-код програми в машинний код після встановлення програми. У версіях Android до версії 5.0 використовувалася віртуальна машина з компіляцією JIT “Dalvik” для запуску dex-коду Dalvik, який зазвичай перекладався з байт-коду Java. В якості імплементації Java Android використовує імплементацію з відкритим кодом “OpenJDK”.

Стандартна бібліотека мови програмування C під назвою “Bionic” що використовується в операційній системи Android була розроблена компанією

Google на основі BSD версії коду стандартної бібліотеки мови програмування C. Основними перевагами використання Bionic є його менший обсяг роботи та оптимізація для низькочастотних процесорів.

Android за замовчуванням не підтримує повний набір стандартних бібліотек GNU, але починаючи з версії Android NDK r5 Android підтримує додатки, що повністю написані мовами програмування C або C ++.

Загальна архітектура системи Android показано на рисунку 2.2.

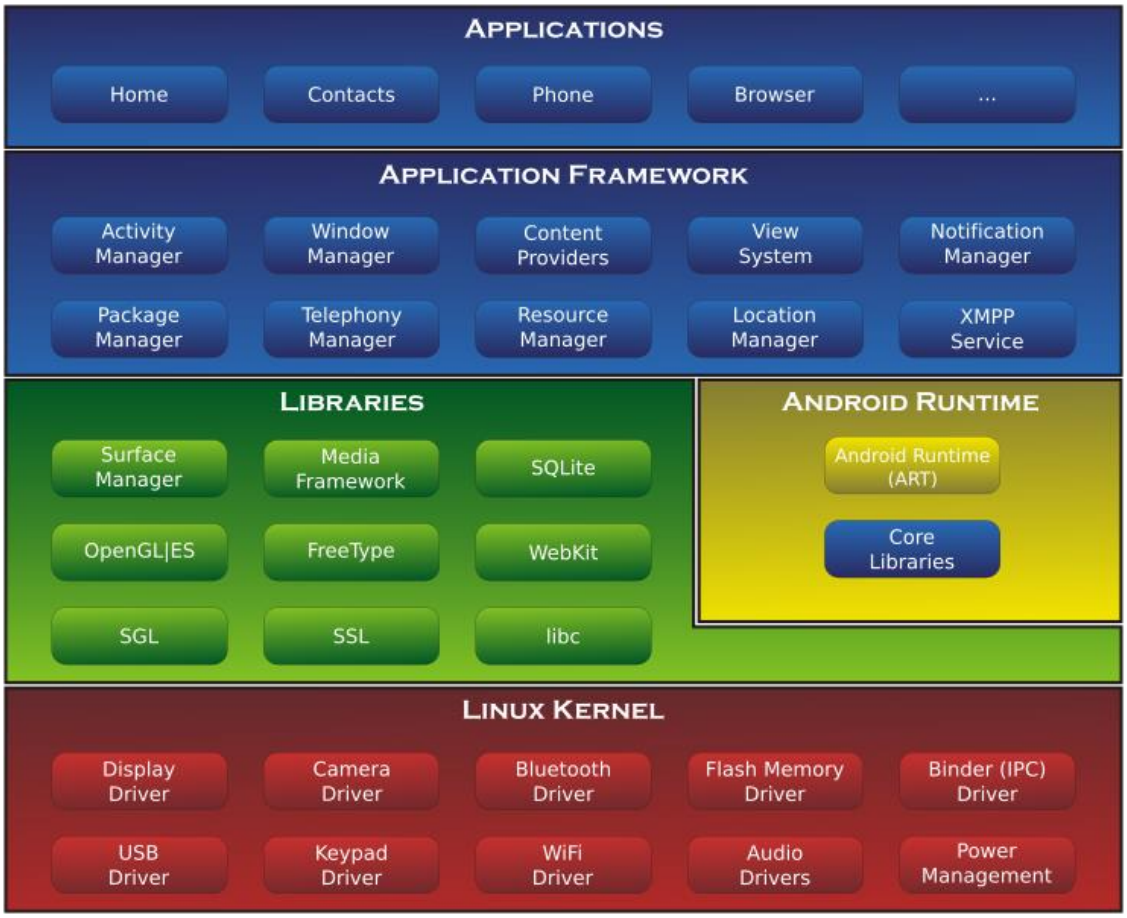


Рисунок 2.2 – Архітектура системи Android

2.2 Мова програмування JavaScript

JavaScript – це легка, високорівнева, інтерпретована, об'єктно-орієнтована мова з першокласними функціями і найвідоміша як мова сценаріїв веб-сторінок, але вона використовується і в багатьох середовищах, що не належать до браузера.

Це мова сценаріїв, що базується на прототипах, з використанням багатьох парадигм, яка є динамічною та підтримує об’єктно-орієнтований, імперативний та функціональний стилі програмування. JavaScript – це проста у вивченні, а також потужна мова сценаріїв, широко застосовувана для контролю поведінки веб-сторінок. Основний синтаксис навмисно схожий як на Java, так і на C ++, щоб зменшити кількість нових понять, необхідних для вивчення мови. Мовні конструкції, такі як оператори if, цикли for і while, а також блоки switch та try ... catch функціонують так само, як у цих мовах. JavaScript може функціонувати як процедурна та як об'єктно-орієнтована мова. Об'єкти створюються програмно в JavaScript, приєднуючи методи та властивості до порожніх об'єктів під час виконання, на відміну від синтаксично визначених класів, поширених у компільованих мовах, таких як C++ та Java. Після побудови об’єкта його можна використовувати як прототип для створення подібних об’єктів. JavaScript має API для роботи з текстом, датами, регулярними виразами, стандартними структурами даних та об'єктною моделлю документа. Стандартом JavaScript є ECMAScript. Станом на 2012 рік, всі сучасні браузери повністю підтримують ECMAScript 5.1. Стандарт ECMAScript не включає операції введення/виведення, мережеві операції, сховища чи графічні засоби. На практиці веб-браузер або інша система виконання забезпечує API для вводу-виводу. Спочатку рушії JavaScript використовувались лише у веб-браузерах, але зараз вони є основними компонентами інших систем виконання, таких як Node.js та Deno. Ці системи використовуються для побудови серверів, а також інтегруються у такі фреймворки, як Electron та Cordova, для створення різноманітних програм. Electron, Cordova та інші програмні засоби використовуються для створення додатків з функціоналом, реалізованим за допомогою мови JavaScript. JavaScript нещодавно почав з'являтися в деяких вбудованих системах, як правило, за допомогою Node.js. React Native дозволяє створювати власні мобільні додатки для Android та iOS, які використовують версію фреймворку React, подібну до веб-сайтів.

2.3 React

React, також відомий як React.js та ReactJS – це JavaScript бібліотека для створення інтерфейсу користувача з відкритим кодом. React підтримується компанією Facebook та спільнотою окремих розробників та компаній. React можна використовувати для розробки веб-додатків або мобільних додатків. Однак React використовується тільки для управлінням станом та відображенням цього стану за допомогою DOM, тому для розробки додатків зазвичай React використовується у поєднанні з іншими JavaScript бібліотеками.

Код написаний за допомогою бібліотеки React складається з компонентів. Компоненти можуть бути відображені як певний елемент DOM. Під час відображення компонента йому можна передавати будь-які значення для подальшого використання, що називаються “пропсами”. Компоненти можна вкладати один в одного, а також передавати дочірнім компонентам будь-яку інформацію у тому числі і методи батьківських компонентів, які можуть викликатися всередині дочірніх компонентів. Таким чином будуються комплексні ієрархії компонентів з яких складається додаток. Існує два основних типи компонентів у React – це функціональні компоненти та компоненти на основі класів. Функціональні компоненти створюються за допомогою функцій, які повертають JSX. Компоненти на основі класів створюються за допомогою класів ES6. Однією з основних особливостей React є використання віртуальної об'єктної моделі документа. React створює кеш-структуру даних у пам’яті, обчислює отриману різницю, а потім оновлює відображуваний DOM. Це дозволяє програмісту писати код так, ніби вся сторінка відображається при кожній зміні, тоді як бібліотеки React відображають лише підкомпоненти, які насправді змінюються. Такий спосіб відображення забезпечує значне підвищення продуктивності. Це економить зусилля з перерахування стилю CSS, макета сторінки та відображення всієї сторінки.

В бібліотеці React є декілька основних методів життєвого циклу компонента. Метод shouldComponentUpdate дозволяє розробнику запобігти непотрібному

повторному відображенню компонента, повертаючи значення false, якщо візуалізація не потрібна. Метод componentDidMount викликається, як тільки компонент "змонтований" (компонент був створений в інтерфейсі користувача, часто шляхом асоціювання його з DOM-вузлом). Це зазвичай використовується для запуску завантаження даних з віддаленого джерела через API. Метод componentWillUnmount викликається безпосередньо перед тим, як компонент буде "демонтований". Це зазвичай використовується для очищення компоненту шляхом видалення всіх ресурсів які використовувалися в компоненті, які не будуть просто видалені автоматично при демонтуванні компонента. Метод render викликається кожного разу, коли стан компонента оновлюється, що повинно відображатися в інтерфейсі користувача. Це найважливіший методот життєвого циклу і єдиний метод, що є обов'язковим у будь-якому компоненті. Саме цей метод визначає як саме повинен відображатися компонент.

Розмітка React компонентів визначається за допомогою JSX. JSX, або JavaScript XML, є розширенням синтаксису мови JavaScript. Подібно до зовнішнього вигляду HTML, JSX надає спосіб структурувати відображення компонентів за допомогою синтаксису, знайомого багатьом розробникам.

React не намагається надати повну бібліотеку для створення додатків. React розроблений спеціально для побудови користувальницьких інтерфейсів, тому не включає багато інструментів, які деякі розробники можуть вважати необхідними для створення додатка. Це дозволяє вибрати, яку бібліотеку розробник воліє використовувати для реалізації того чи іншого функціоналу.

Для підтримки концепції React про односпрямований потік даних була розроблена архітектура Flux як альтернатива архітектурі model-view-controller. Особливістю Flux є дії, які надсилаються через центральний диспетчер до сховища, після чого всі зміни в сховищі відправляються назад відповідним компонентам. В React для такого розповсюдження використовуються властивості компонентів. Такий спосіб реалізації Flux є не дуже зручним, тому зазвичай для цього використовується бібліотека Redux. Flux можна вважати варіантом паттерну “observer”. Компонент React під архітектурою Flux не модифікує

передані йому пропи, але передає функції зворотного виклику, які створюють дії, які відправляються диспетчером для модифікації сховища. Дія – це об'єкт, що використовується для опису того, що відбулося. Сховища – це моделі, які можуть змінюватись у відповідь на дії, отримані від диспетчера. Таку закономірність інколи називають "властивості стікають вниз, дії течуть вгору". Модель архітектури Flux показано на рисунку 2.3. З моменту його створення було створено багато реалізацій Flux, але найвідомішим є Redux, який має єдине сховище, яке часто називають єдиним джерелом істини. Redux – це бібліотека JavaScript з відкритим кодом для управління станом програми. Найчастіше використовується з такими бібліотеками, як React або Angular, для побудови користувальницьких інтерфейсів.

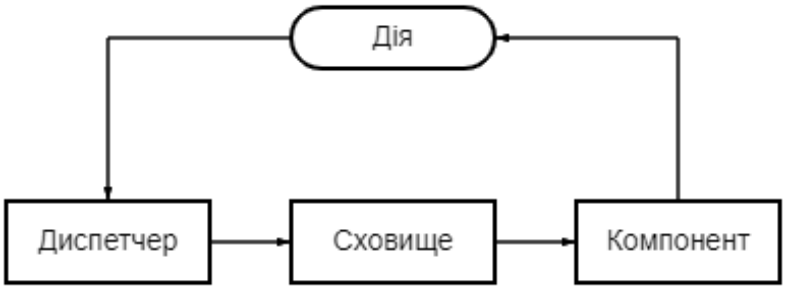


Рисунок 2.3 – Модель архітектури Flux

2.4 React Native

React Native – це фреймворк для розробки мобільних додатків з відкритим кодом, створений компанією Facebook. Він дозволяє розробникам використовувати для створення додатку фреймворк React та усі можливості платформи для якої розробляється додаток.

Принцип роботи React Native практично ідентичний принципу роботи фреймворку React, але на відміну від React, React Native не маніпулює DOM через віртуальний DOM, а працює у фоновому процесі, який інтерпретує JavaScript код та взаємодіє з платформою використовуючи серіалізовані дані які надсилаються через асинхронний міст.

Компоненти React обгортають нативний код платформи та взаємодіють із API платформи через декларативну парадигму інтерфейсу користувача React та JavaScript. Хоча стиль React Native має подібний синтаксис до CSS, він не використовує HTML або CSS. Натомість повідомлення з потоку JavaScript використовуються для маніпулювання нативними поданнями. React Native також дозволяє розробникам писати нативний код такими мовами, як Java або Kotlin для Android та Objective-C або Swift для iOS, що робить його ще більш гнучким.

2.5 Firebase

Firebase – це платформа для створення мобільних та веб-додатків, що розробляється компанією Google. Спочатку це була незалежна компанія, заснована в 2011 році, але у 2014 році компанія Google придбала цю платформу.

Платформа Firebase надає продукти та рішення які можна використовувати для розробки додатків. Серед усіх продуктів та рішень потрібно виділити такі рішення як Firebase Auth, Realtime Database, Cloud Firestore, Firebase Storage, Firebase Hosting, Firebase Cloud Messaging та Firebase Analytics.

Firebase Auth – дозволяє аутентифікувати користувачів. Firebase Auth підтримує різні провайдери такі як Facebook, GitHub, Twitter і Google. Крім того, Firebase Auth включає в себе систему управління користувачами.

Realtime Database – це база даних в режимі реального часу. Ця база даних надає розробникам можливість синхронізувати дані між клієнтами та зберігати їх у хмарі Firebase.

Cloud Firestore є наступним поколінням Firebase Realtime Database. Cloud Firestore – це повністю керована NoSQL база даних документів для розробки мобільних та веб-додатків. Вона призначена для легкого зберігання та синхронізації даних додатків у глобальному масштабі. Ця база даних дозволяє синхронізувати дані між різними пристроями в реальному часі, створювати колекції та документи для структурування даних та швидкого пошуку даних та отримувати доступ до даних навіть при відсутності інтернет з'єднання.

Firebase Storage – це сховище файлів для додатків. Сховище надає розробникам можливість зберігати в ньому зображення, аудіо, відео та інші типи файлів.

Firebase Hosting – це статичний та динамічний веб-хостинг, що підтримує статичні файли, такі як CSS, HTML, JavaScript та інші, а також реалізує підтримку Node.js через систему Cloud Functions. Firebase Hosting передає всю інформацію через CDN використовуючи протокол HTTPS та шифрування SSL.

Firebase Cloud Messaging – це це крос-платформне рішення для повідомлень для Android, iOS та веб-додатків.

Firebase Analytics – рішення що дозволяє виконувати аналітичний аналіз та переглядати статистику додатків.

Платформа Firebase підтримує велику кількість платформ та мов програмування. За допомогою Firebase можна створювати Android, iOS та веб-додатки використовуючи такі мови програмування, як C++, JavaScript, Java, Kotlin, Objective-C, та Swift.

2.6 Висновки до розділу

В даному розділі було розглянуто методи реалізації програмного продукту. Було розглянуто операційну систему Android, її історію, архітектуру, особливості роботи та особливості розробки Android додатків. Було описано набір технологій, програмних засобів та сервісів, що були використані в процесі створення додатку, а саме: мову програмування JavaScript, що була використана як основна мова для розробки додатку, бібліотека React, що була використана для створення користувацького інтерфейсу додатку, фреймворк React Native, що дозволяє використовувати бібліотеку React для створення нативних мобільних додатків, платформу для розробки мобільних та веб додатків Firebase, що була використнана в додатку для аутентифікації користувачів та за зберігання медіа файлів користувачів, а також були розглянуті їх принципи роботи та використання.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Опис функціональності додатку

Додаток містить в собі одного актора, що взаємодіє з системою. Цим актором є користувач. На рисунку 3.1 представлена діаграма варіантів використання, яка описує функції та дії актора в додатку.

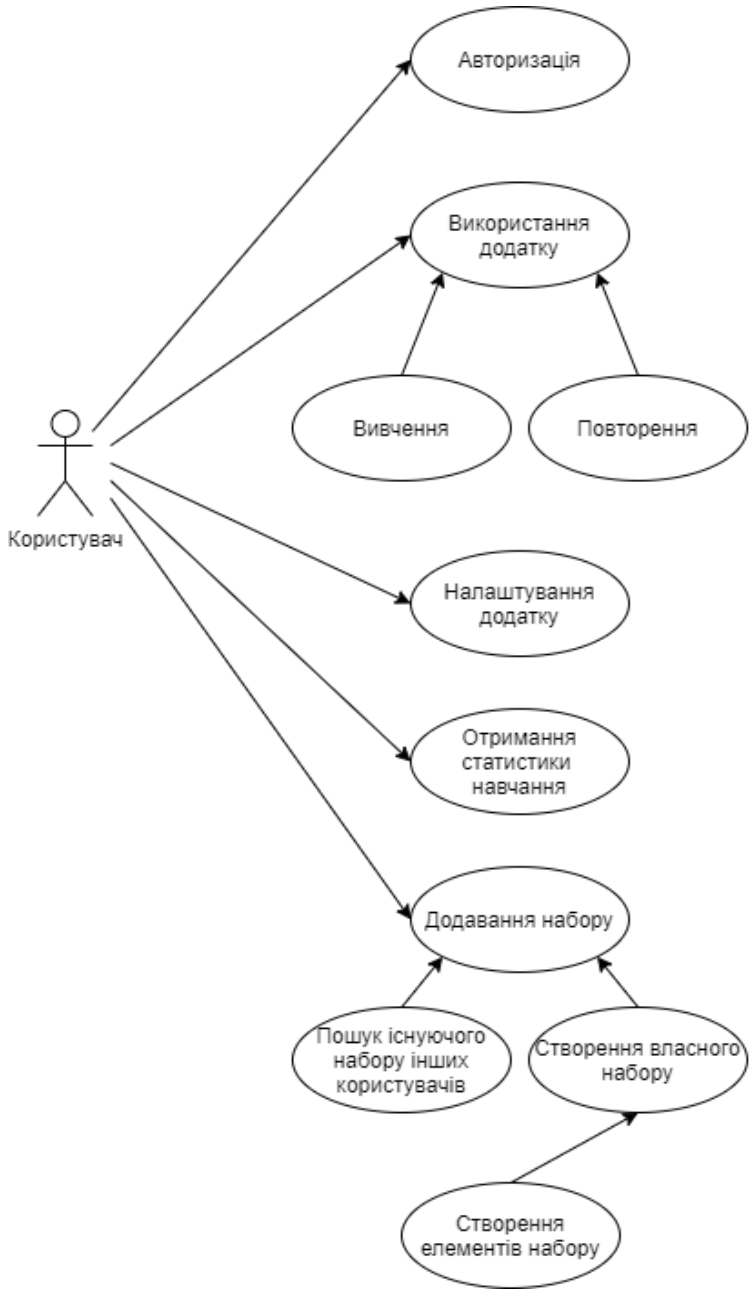


Рисунок 3.1 – Діаграма варіантів використання

3.2 Структура додатку

Додаток складається з багатьох вікон. Вікна в додатку називаються “сценами”. Кожна сцена має своє призначення і складається з певних компонентів. Для можливості перемикати сцени було створено компонент “роутер”. Цей компонент відображає обрану в даний час сцену. Реалізована можливість змінювати поточну сцену, зберігати сцени в історії та повертатися до минулої сцени. Даний функціонал реалізований за допомогою бібліотеки Redux. Кожному компоненту, в якому необхідно мати можливість змінювати сцену надається дії pushScene, popScene або setScene. Ці дії надають можливість змінити поточну сцену на будь-яку іншу і передати в неї будь-яку інформацію. Після виклику однієї з цих дій за допомогою диспетчера ініціюється зміна стану в сховищі і оновлений стан передається в роутер, після чого роутер відображає обрану сцену. Усього в додатку одинадцять сцен, а саме: Авторизація, Головна, Пошук, Профіль, Опції, Навчання, Повторення, Результат, Набір, Редагування набору та Редагування елемента. Структуру навігації в додатку показано на рисунку 3.1.

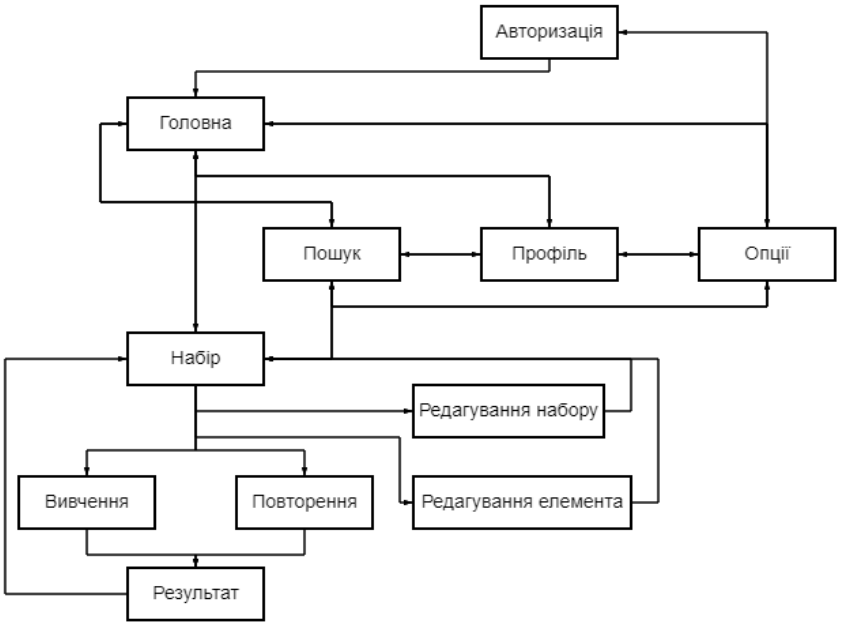


Рисунок 3.2 – Структура навігації в додатку

3.3 Сцена авторизації

Сцена авторизації дозволяє користувачу увійти в свій аккаунт або створити новий. На цій сцені є форма, яку користувач може заповнити своїми реквізитами для входу або реєстрації. Авторизація реалізована за допомогою бібліотеки Firebase Authentication, а саме за допомогою функцій signInWithEmailAndPassword та createUserWithEmailAndPassword. За допомогою цих функцій можна авторизувати користувача, або створити новий аккаунт з автоматично згенерованим унікальним id користувача. Firebase Authentication надає можливість надсилати листи на пошту користувачам, тому також були реалізовані функції підтвердження пошти та відновлення паролю за допомогою посилання, що надсилається на електронну пошту користувачу. Сцену авторизації показано на рисунку 3.3.

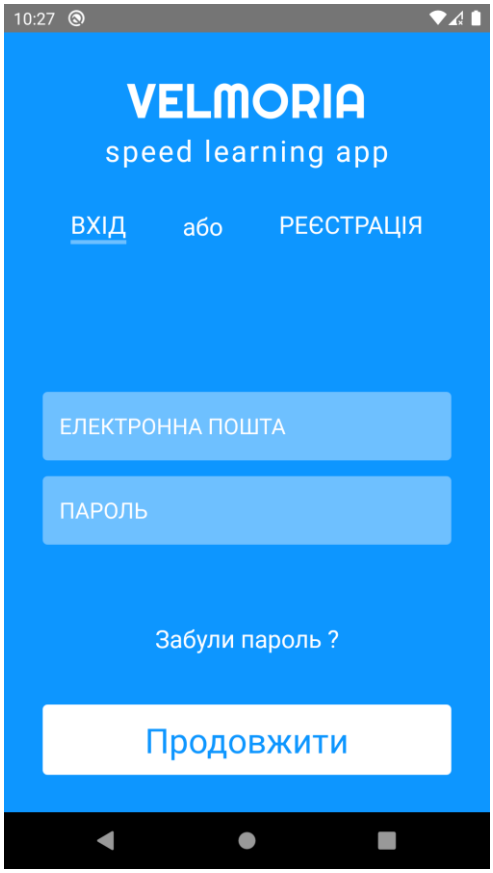


Рисунок 3.3 – Сцена авторизації

JSX код сцени авторизації:

```
<AppContainer>
    <Logo />
    <AuthSelector/>
    <View>
        <AuthError />
        <AuthForm />
        <AuthForgotPassword />
    </View>
    {button}
</AppContainer>
```

3.4 Головна сцена

На головній сцені знаходится список всіх наборів флеш-карток. Для кожного набору показується його назва, кількість флеш-карток, що називаються просто “елементами набору”, кількість вивчених елементів, відсоток вивчення набору, а також кнопка, що викликає модальне вікно зі списком всіх дій, що можна виконати з набором, а саме: відкрити, редагувати, видалити. Окрім списку наборів на сцені є кнопка, що дозволяє створити новий набір. Також зверху відображається назва поточної сцени, а знизу є меню для переходу між основними сценами. JSX код головної сцени:

```
<AppContainer>
    <ContentContainer>
        {this.state.sets.length<=0?<Placeholder text={ "You do not have any sets
        yet." />:null}
        {this.state.sets.length>0?
        <ScrollContainer>
            {this.state.sets.map((value, index) => {
             return <Set
```

```
                    set={value}
                    key={index}>
                  </Set>
                })}
              </ScrollContainer>:null}
              <SquareButton text={ "Add new set} onPress={()=>{this.addNewSet();}}
              />
        </ContentContainer>
</AppContainer>
```

Головну сцену показано на рисунку 3.4.

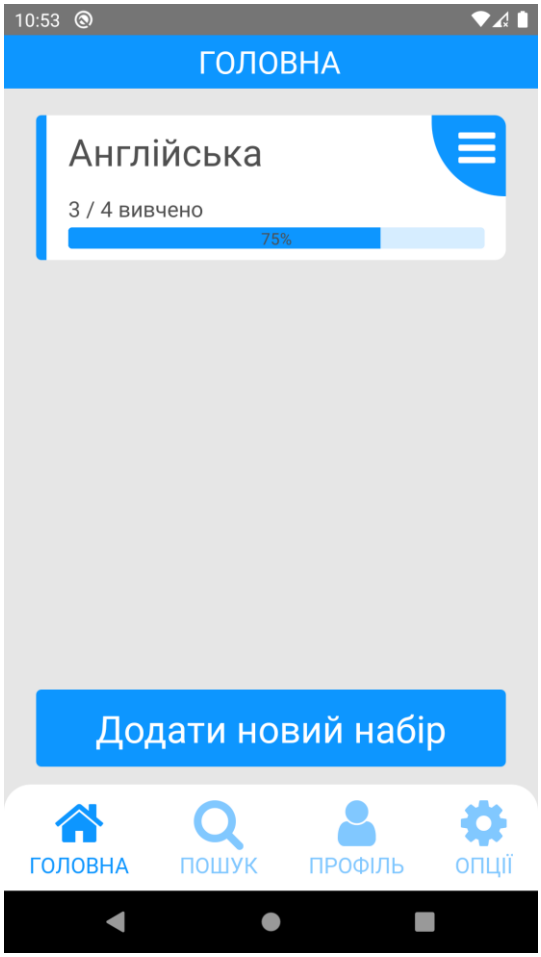


Рисунок 3.4 – Головна сцена

Всі набори, а також вся інформація про них зберігається не локально, а в базі даних. В якості бази даних використовується NoSQL база даних Firebase

Cloud Firestore. В Firebase Cloud Firestore вся інформація зберігається в документах, що групуються в колекції. Набори зберігаються в колекції під назвою “Sets”. Структуру документа набору показано на рисунку 3.4.

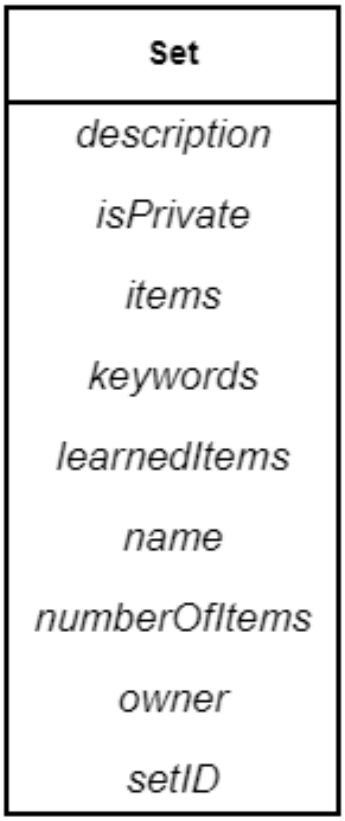


Рисунок 3.5 – Структура документа набору

Документ набору складається з дев'яти полів. В полі “description” зберігається короткий опис набору. Поле “isPrivate” визначає можливість пошуку і використання набору іншими користувачами. В полі “items” зберігається масив усіх елементів набору. Поле “keywords” – це масив ключових слів для пошуку набору. В полях “numberOfItems” та “learnedItems” зберігається загальна кількість елементів та кількість вивчених елементів набору. Поле “name” – це назва набору. В полях “owner” та “setID” зберігаються id власника набору та id самого набору.

3.5 Сцена пошуку

На сцені пошуку знаходиться поле для вводу запиту для пошуку наборів

інших користувачів. Під полем для пошуку відображається список всіх знайдених наборів. Пошук виконується за ключовими словами, що зберігаються в базі даних для кожного набору. Для кожного набору відображається його назва та кнопка, що дозволяє клонувати набір для подальшого використання. Також зверху відображається назва поточної сцени, а знизу є меню для переходу між основними сценами. Сцену пошуку показано на рисунку 3.6.

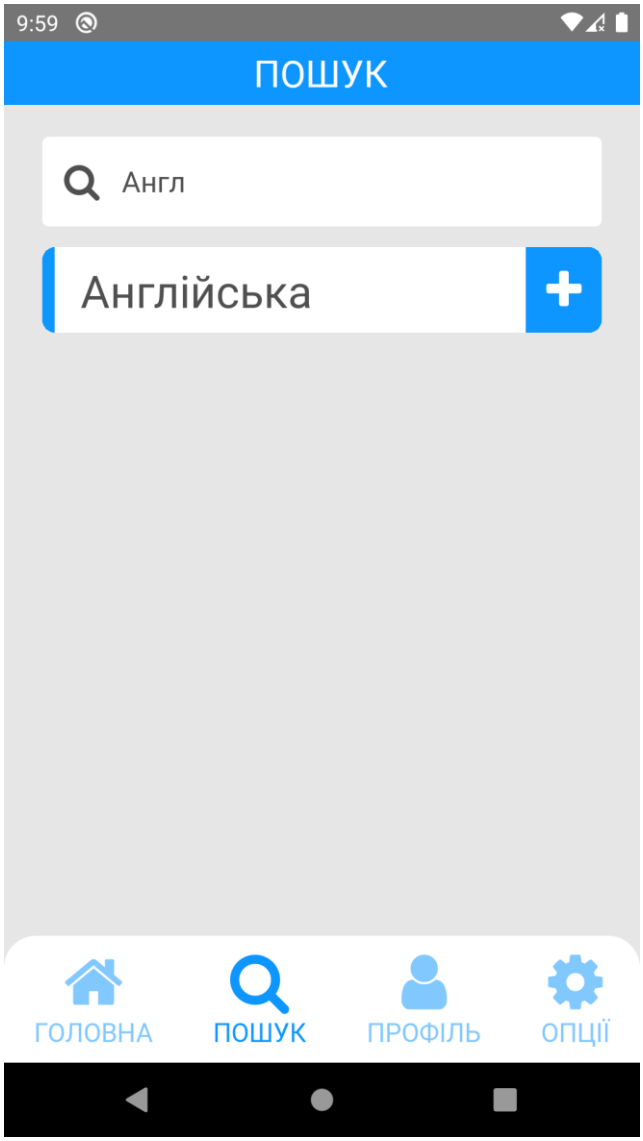


Рисунок 3.6 – Сцена пошуку

JSX код сцени пошуку:

```
<AppContainer>
    <ContentContainer>
```

```
            <Input value={this.state.searchQuery}
            onChange={(query)=>{this.findSets(query);}} showIcon={true}
            icon='search' placeholder={ "Search} />
                    <ScrollContainer>
                    {this.props.FoundSets.length>0?this.props.FoundSets.map((value,
            index) => {
                        return <FoundSet key={index} set={value} />})
                    :<Placeholder  text={ "Nothing found."} />}
                    </ScrollContainer>
        </ContentContainer>
</AppContainer>
```

3.6 Сцена профілю

На сцені профілю знаходиться фото та ім'я користувача, а також основна інформація та статистика навчання. На сцені відображається така інформація як рахунок користувача, тобто кількість балів що користувач заробив під час навчання, кількість вивчених елементів, кількість сесій вивчення та повторення та час витрачений на навчання. Також зверху відображається назва поточної сцени, а знизу є меню для переходу між основними сценами.

JSX код сцени профілю:

```
<AppContainer>
        <ContentContainer>
          <ScrollContainer>
            <User user={this.state.user} />
            <View style={styles.Statistics}>
            // Statistics
            </View>
            <Schedule />
            <Progress />
```

```
            <Leaderboard/>
          </ScrollContainer>
        </ContentContainer>
    </AppContainer>
```

Приклад JSX коду блоку статистики:

```
<View style={styles.StatisticsBlock}>
        <Text style={styles.StatisticsItem}>{"Score: + score}</Text>
</View>
```

Сцену профілю показано на рисунку 3.7.

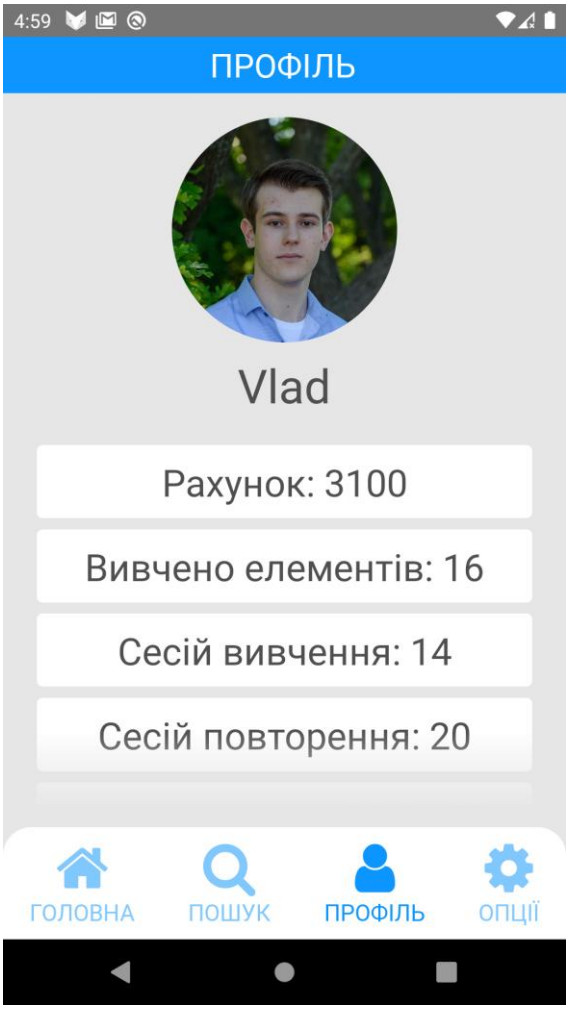


Рисунок 3.7 – Сцена профілю

Крім цього на сцені ще відображаються компоненти “розклад”, “прогрес” та “таблиця лідерів”. Компонент “розклад” показує усі дні в які користувач

використовував додаток. Компонент “розклад” показано на рисунку 3.8.

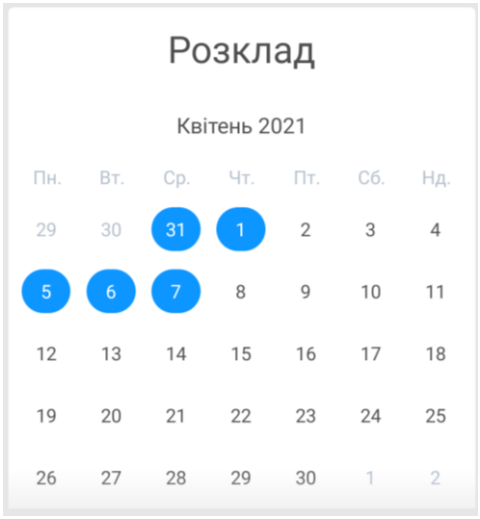


Рисунок 3.8 – Компонент “розклад”

Компонент “прогрес” показує користувачу графік вивчення нових елементів за поточний тиждень. Компонент “прогрес” показано на рисунку 3.9.

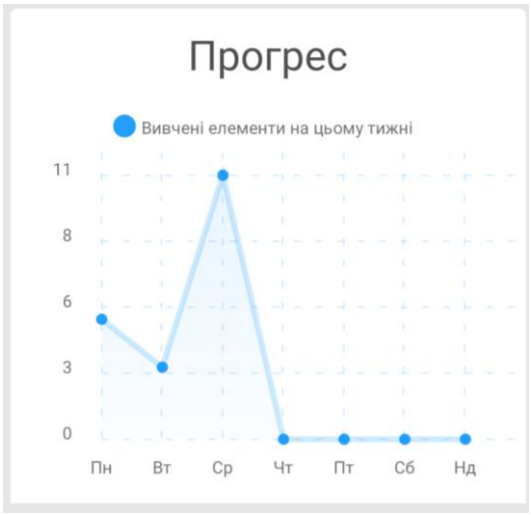


Рисунок 3.9 – Компонент “прогрес”

Компонент “таблиця лідерів” показує користувачу список найбільш успішних користувачів додатку з найбільшою кількістю балів. Компонент “таблиця лідерів” показано на рисунку 3.10.

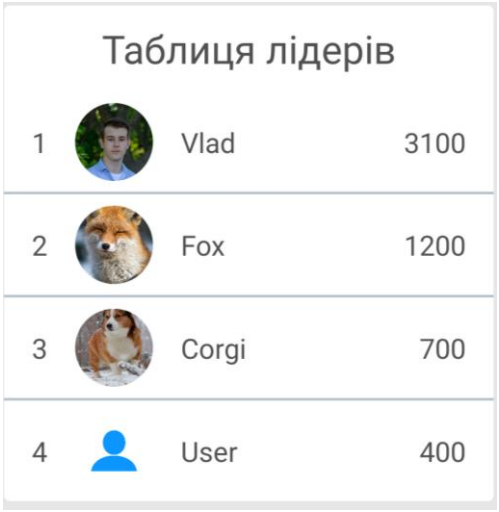


Рисунок 3.10 – Компонент “таблиця лідерів”

Вся інформація, що відображається на сцені профілю зберігається в базі даних Firebase Cloud Firestore. Документи що містять інформацію про користувача зберігаються в колекції під назвою “Users”. Структуру документа користувача показано на рисунку 3.11.

Рисунок 3.11 – Структура документа користувача

Документ користувача складається з дев'яти полів. В полі “email” зберігається електронна пошта користувача. В полі “itemsLearned” зберігається кількість вивчених користувачем елементів. В полі “name” зберігається ім'я користувача. В полі “photoURL” зберігається посилання на фото профілю користувача. В полях “learnSessions” та “reviewSessions” зберігається кількість сесій навчання та повторення. Поле “uid” – це унікальний ідентифікатор користувача згенерований за допомогою бібліотеки Firebase Authentication. В полі “score” зберігається кількість зароблених користувачем балів, а в полі “timeLearning” зберігається час, що користувач витратив на навчання.

Документи що містять статистику навчання користувача зберігаються в колекції під назвою “Statistics”. Документ статистики створюються для кожного користувача кожного дня, якщо користувач використовує додаток і за день пройшов хоча б одну сесію навчання або повторення. Структуру документа статистики показано на рисунку 3.12.

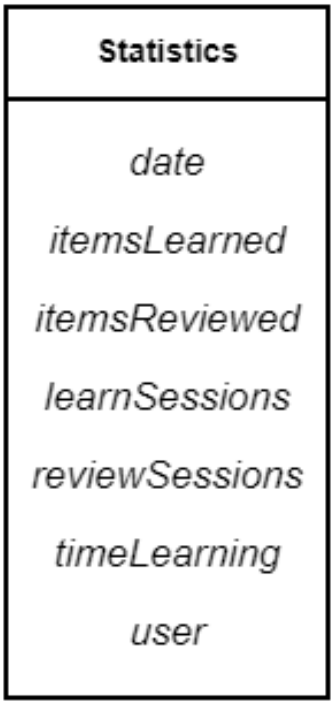


Рисунок 3.12 – Структура документа статистики

Документ статистики складається з семи полів. В полі “date” зберігається

дата створення документа. В полях “learnSessions” та “reviewSessions” зберігається кількість сесій навчання та повторення за поточний день. Поле “user” – це унікальний ідентифікатор користувача статистика якого зберігається в даному документі. В полях “itemsLearned” та “itemsReviewed” зберігається кількість вивчених та повторених елементів набору за поточний день. В полі “timeLearning” зберігається час, що користувач витратив на навчання за поточний день.

3.7 Сцена опцій

На сцені опцій відображаються усі можливі налаштування додатку. На цій сцені є декілька полів різного типу. Два поля для введення тексту дозволяють задати максимальну кількість елементів набору для повторення або навчання за одну сесію. Три інші поля при натисканні викликають модальні вікна в яких відображаються усі можливі варіанти вибору. Одне з цих полів дозволяє змінити локалізацію додату, а два інших дозволяють змінити налаштування звуку та вібрації. Крім, цього на сцені відображаються дві кнопки, що дозволяють змінити або видалити свій аккаунт. Також зверху відображається назва поточної сцени, а знизу є меню для переходу між основними сценами.

JSX код сцени опцій:

```
<AppContainer>
<ContentContainer>
    <ScrollContainer>
    //Options
    <SquareButton text={"Switch account"} onPress={()=>{this.signOut();}}/>
    <SquareButton text={"Delete account"} onPress={()=>{this.delete();}} />
    </ScrollContainer>
</ContentContainer>
</AppContainer>
```

Приклад JSX коду компонента вводу з варіантами вибору опцій:

<PickerSelect title={"Localization"} options={["ENG", "UKR"]}
value={this.props.Locale}
callback={(result)=>{this.props.saveConfig(this.props.Config, {"locale":result});}} />

Приклад JSX коду компонента вводу опцій:

<Input value={this.props.MaxItemsPerLearnSession.toString()}
onChange={(text)=>{this.props.saveConfig(this.props.Config,
{"maxItemsPerLearnSession": parseInt(text, 10)});}} label={"Max items per learn
session:"/>

Сцену опцій показано на рисунку 3.13.

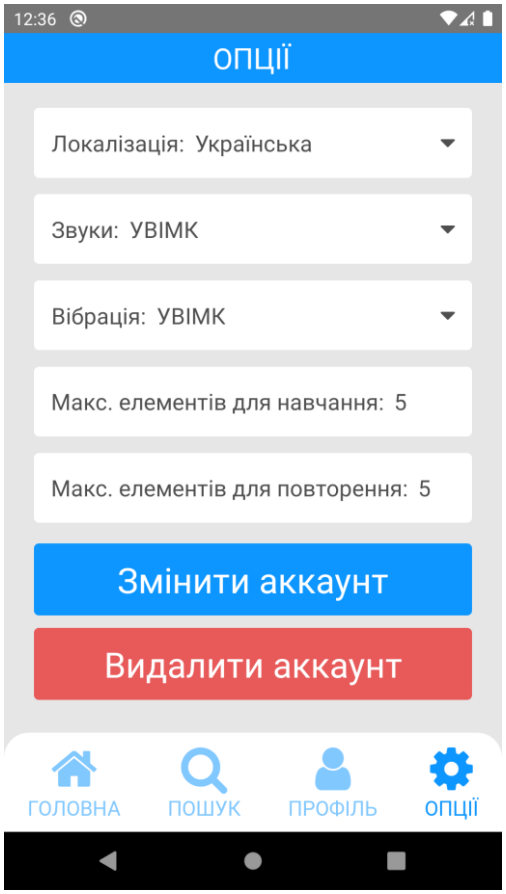


Рисунок 3.13 – Сцена опцій

Видалення та зміна аккаунта відбувається за допомогою бібліотеки Firebase Authentication, а саме за допомогою функцій signOut та delete.

Всі налаштування зберігаються в локальному сховищі додатку на пристрої користувача. Доступ до локального сховища надається за допомогою бібліотеки

“sync-storage”. Дані з локального сховища зберігаються в сховищі додатку та передаються усім компонентам за допомогою бібліотеки “Redux”.

3.8 Сцена набору

На сцені набору відображається вся інформація пов'язана з обраним набором, а саме: назва, короткий опис набору, загальна кількість елементів набору, кількість вивчених елементів та відсоток вивчення набору. Також на сцені відображаються кнопки для старту сесій вивчення та повторення. На цих кнопках в дужках позначено кількість елементів що необхідно вивчити та повторити. Крім цього, на сцені відображається список усіх елементів набору та кнопка що дозволяє додати новий елемент.

JSX код сцени набору:

```
<AppContainer>
<ContentContainer>
<ScrollContainer>
//Info
<View style={styles.SetButtons}><Button onPress={()=>{this.learn()}}
style={styles.SetButton}><Text
style={styles.SetButtonText}>{"Learn"+toLearn)}</Text></Button>
<Button onPress={()=>{this.review()}} style={styles.SetButton}><Text
style={styles.SetButtonText}>{"Review"+toReview)}</Text></Button>
</View>
<View style={styles.Block} />
{items.length>0?items.map((value, index) => {
            return <Item
            item={value}
            set={this.state.set}
           </Item>
}):
```

```
</ScrollContainer>
<SquareButton text={"Add new item"} onPress={()=>{this.createItem();}} />
</ContentContainer>
</AppContainer>
```

Приклад JSX коду блока інформації про набір:

```
<View style={styles.Block}>
<Text style={styles.InfoTextMajor}>{name}</Text>
</View>
<ProgressBar progress={learnedItems/numberOfItems}></ProgressBar>
```

Сцену набору показано на рисунку 3.14.

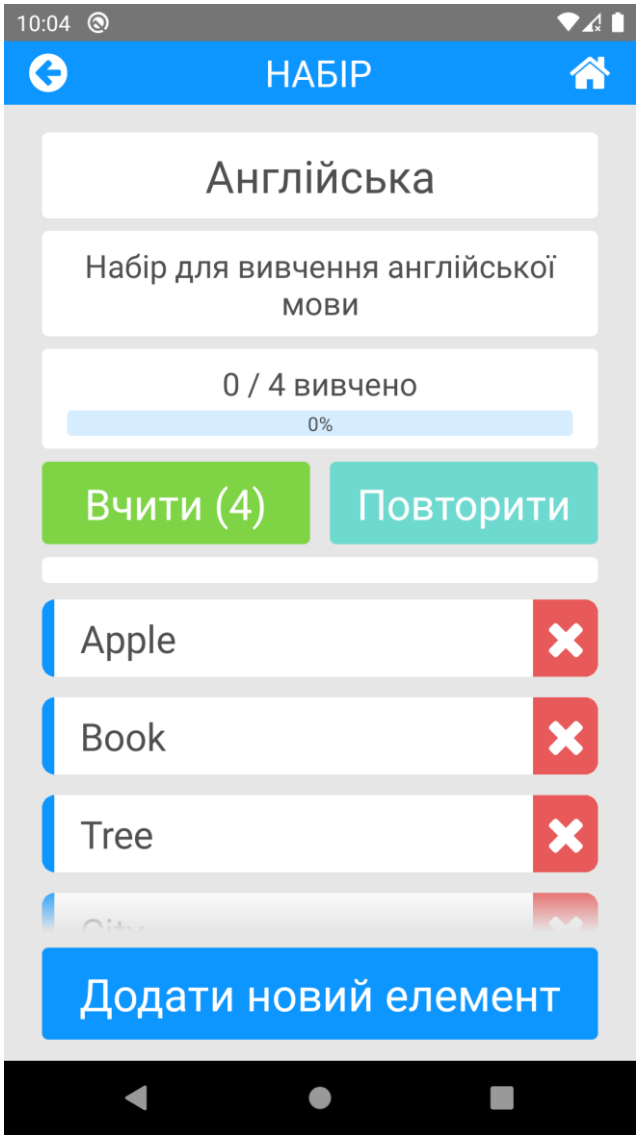


Рисунок 3.14 – Сцена набору

3.9 Сцена редагування набору

Сцена редагування набору дозволяє створювати та вносити зміни в уже створені набори. На сцені відображаються поля для редагування, а саме: поле з ім'ям набору, поле з коротким описом набору та поле, що дозволяє встановити приватність набору. Крім цього на сцені є кнопка, що дозволяє зберегти зміни і відправити оновлену інформацію до бази даних. Також зверху відображається назва поточної сцени та кнопки для навігації між сценами. Сцену редагування набору показано на рисунку 3.15.

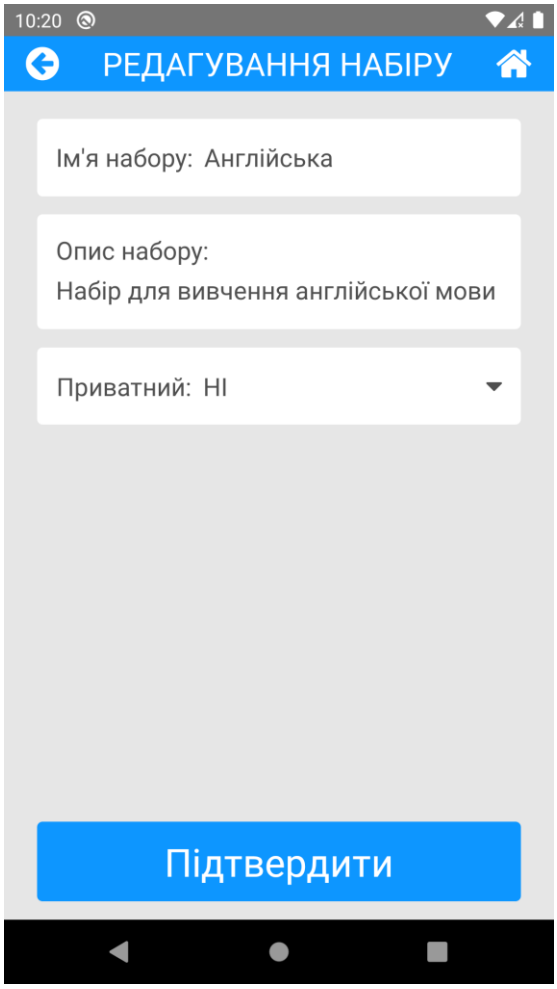


Рисунок 3.15 – Сцена редагування набору

JSX код сцени редагування набору:

<AppContainer>

```
<ContentContainer>
<ScrollContainer>
//Inputs
</ScrollContainer>
{this.props.Props.setID==null?
<SquareButton text={"Confirm"} onPress={()=>{this.createSet();}} /> :
<SquareButton text={"Confirm"} onPress={()=>{this.editSet();}} /> }
</ContentContainer>
</AppContainer>
```

Приклад JSX коду компонента вводу для редагування набору:

```
<Input value={this.state.setName} onChange={(text)=>{this.setState({setName: text});}} label={"Set name:"} placeholder= "New set"}/>
```

3.10 Сцена редагування елемента

Сцена редагування елемента дозволяє створювати та вносити зміни в уже створені елементи набору. На сцені відображаються поля для редагування, а саме: поле з запитанням, відповіддю, поясненням відповіді, поля, що дозволяють додати до елемента аудіо та медіа файли та поля, що дозволяють налаштувати автогенерацію аудіо та медіа. Автогенерація аудіо працює за допомогою бібліотеки react-native-tts, що дозволяє генерувати аудіо за допомогою синтезу тексту в мовлення. Мову автогенерації можна вибрати в окремому полі. Автогенерація медіа працює за допомогою пошуку відповідних до запитання зображень використовуючи відкритий API сервісу pixabay.com, що має велику базу різноманітних зображень. Якщо автогенерація аудіо або медіа вимкнена, то можна завантажити аудіо або медіа файл з локального сховища пристрою. Для цього використовується бібліотека Firebase Storage, що дозволяє завантажувати файли на хмарне сховище Google та отримувати посилання на них для подальшого використання. Крім цього на сцені є кнопка, що дозволяє зберегти зміни і відправити оновлену інформацію до бази даних. Також зверху відображається

назва поточної сцени та кнопки для навігації між сценами. JSX код сцени редагування елемента:

```
<AppContainer>
<ContentContainer>
<ScrollContainer>
//Inputs
</ScrollContainer>
<SquareButton text={"Confirm"} onPress={()=>{this.editItem();}} />
</ContentContainer>
</AppContainer>
```

Приклад JSX коду компонента вводу для редагування елемента набору:

```
<Input value={this.state.question} onChange={(text)=>{this.setState({question:
text});}}
label={"Question:"}
 placeholder={"Your question"}/>
```

Приклад JSX коду компонента вводу для редагування елемента набору з варіантами вибору:

```
<PickerSelect title={"Autogenerate audio"} options={["YES", "NO"]}
value={this.state.autoGenerateAudio}
callback={(result)=>{this.setState({autoGenerateAudio: result});}} />
```

Приклад JSX коду компонента вводу для редагування елемента набору з можливістю вибору файла:

```
<PickerFile
title={"Audio"}
value={audioFileValue}
size={this.state.audio.size}
type={"audio"}
callback={(result)=>{this.pickAudio(result);}} />:null}
```

Сцену редагування елемента показано на рисунку 3.16.

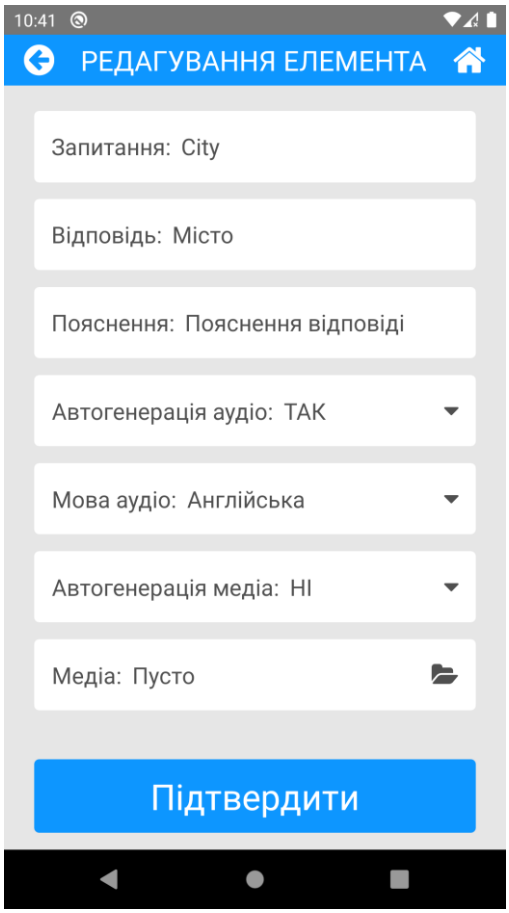


Рисунок 3.16 – Сцена редагування елемента

3.11 Сцена вивчення

Сцена вивчення дозволяє користувачу додатку вивчати інформацію. На сцені відображається уся інформація пов'язана з певним елементом набору, а саме: запитання, відповідь, пояснення відповіді, та відео або зображення. Крім цього на сцені є кнопка, що дозволяє програти аудіо елемента та дві кнопки для завершення сесії навчання та переходу до наступного елемента. Перед переходом переходом до наступного елемента відбувається перехід на сцену повторення для закріплення вивченого матеріалу. Також зверху відображається назва поточної сцени та кнопки для навігації між сценами.

JSX код сцени вивчення:

<AppContainer>

<ContentContainer>

```
{this.state.currentItem!=null&&this.state.otherItems.length!=0&&this.stat
e.showReview==true?
<ReviewItem item={this.state.currentItem.item}
other={this.state.otherItems}
onFinish={()=>{this.finish();}}
onConfirm={(result)=>{this.nextStep(result);}} />:null}
{this.state.currentItem!=null&&this.state.otherItems.length!=0&&this.stat
e.showLearn==true?
<LearnItem item={this.state.currentItem.item}
onFinish={()=>{this.finish();}}
onNext={()=>{this.reviewLearned();}} />:null}
</ContentContainer>
</AppContainer>
```

Сцену вивчення показано на рисунку 3.17.

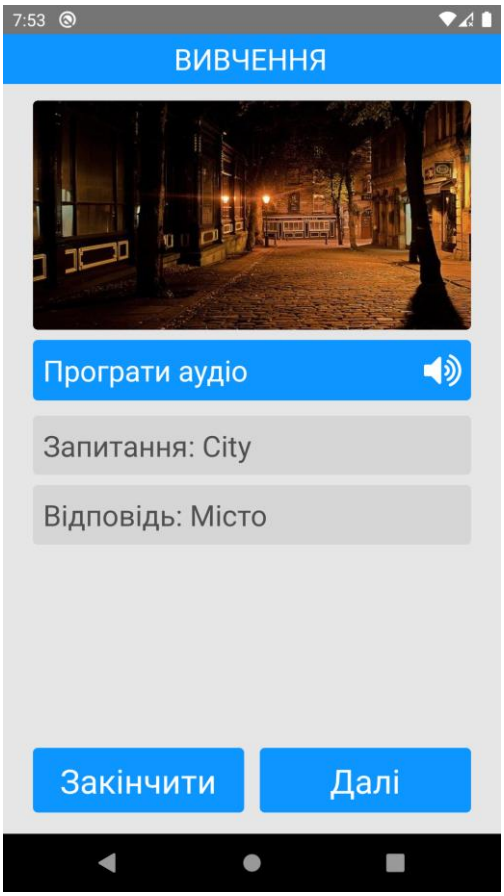


Рисунок 3.17 – Сцена вивчення

3.12 Сцена повторення

Сцена повторення дозволяє користувачу додатку повторювати вивчену інформацію. На сцені відображається питання та варіанти відповідей. Питанням може бути сам текст запитання, або пов'язаний з елементом медіа або аудіо файл. Тип запитання вибирається випадково. Варіанти відповідей також вибираються випадково з-поміж інших елементів набору. Крім цього на сцені є дві кнопки для завершення сесії повторення та підтвердження вибору. Після підтвердження відповіді відображається правильна відповідь, а після паузи – наступне запитання. Також зверху відображається назва поточної сцени та кнопки для навігації між сценами. Сцену повторення показано на рисунку 3.18.

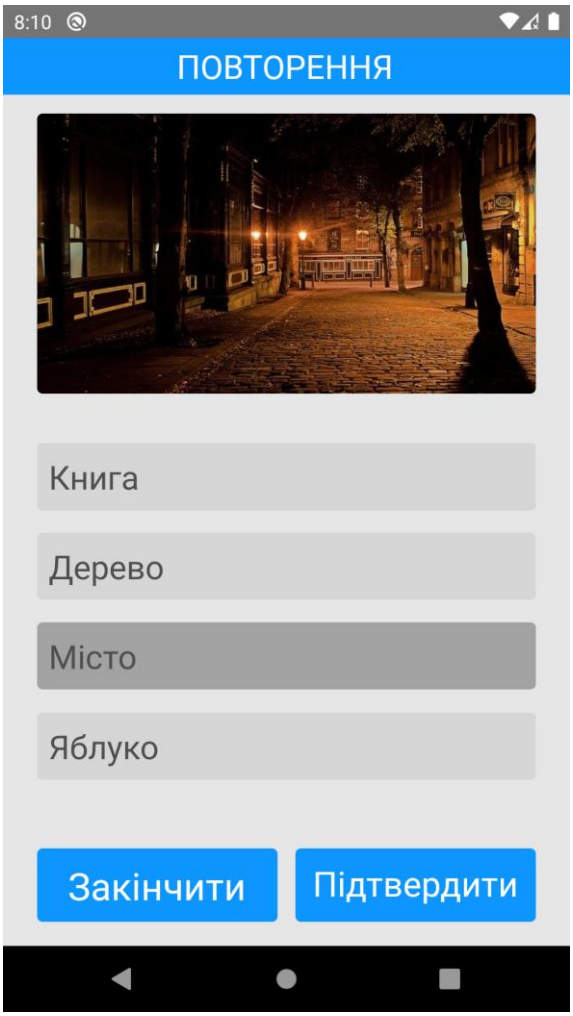


Рисунок 3.18 – Сцена повторення

JSX код сцени повторення:

```
<AppContainer>
    <ContentContainer>
        {this.state.currentItem!=null&&this.state.otherItems.length!=0&&this.stat
        e.showReview==true?
        <ReviewItem
        item={this.state.currentItem.item}
        other={this.state.otherItems}
        onFinish={()=>{this.finish();}}
        onConfirm={(result)=>{this.nextStep(result);}} />:null}
    </ContentContainer>
</AppContainer>
```

3.13 Сцена результату

Сцена результату показує результат проходження сесії навчання або повторення. На сцені відображається кількість повторених елементів, кількість правильних та неправильних відповідей, точність, витрачений час, середній час відповіді на запитання та рахунок. За кожну правильну відповідь користувач отримує 100 балів. Крім цього, на сцені відображається кнопка завершення сесії навчання або повторення. Після натискання цієї кнопки результат сесії надсилається до бази даних і статистика користувача оновлюється. Також зверху відображається назва поточної сцени та кнопки для навігації між сценами.

JSX код сцени результату:

```
<AppContainer>
<ContentContainer>
<ScrollContainer>
<Text style={styles.Header}>
{"Review session result"}
</Text>
```

```
//Results
</ScrollContainer>
  <SquareButton
       text={"Finish"}
       onPress={()=>{this.finish();}} />
</ContentContainer>
</AppContainer>
```

Приклад JSX коду блока результату сесії вивчення або повторення:

```
<View style={styles.Block}><Text style={styles.InfoTex}>{"Items reviewed:
"+(this.state.items-this.state.notAnswered)}</Text></View>
```

Сцену результату показано на рисунку 3.19.

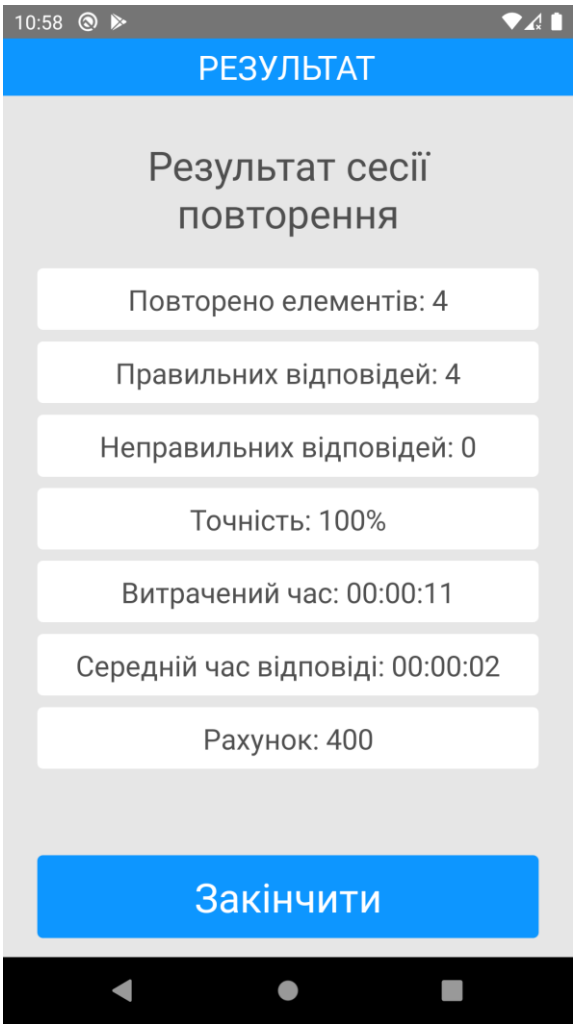


Рисунок 3.19 – Сцена результату

3.14 Висновки до розділу

В даному розділі було детально описано програмну реалізацію додатку та її особливості. Була детально розглянута структура та вся функціональність додатку. Було показано діаграму варіантів використання та структуру навігації в додатку. Було розглянуто структуру документів в базі даних, що використовуються в додатку для зберігання усіх даних користувачів додатку та усі основні компоненти додатку, а саме: роутер, розклад, прогрес, таблицю лідерів, сцену авторизації, головну сцену, сцену пошуку, сцену профілю, сцену опцій, сцену набору, сцену редагування набору, сцену редагування елемента, сцену вивчення, сцену повторення та сцену результату. Для кожного з наведених компонентів було подано детальний опис структури компонанта та усіх елементів з яких він складається, детальний опис функціоналу та принципу роботи, спосіб використання компонента та зображення, на яких показано загальний зовнішній вигляд компонента. Також було розглянуто особливості реалізації цих компонентів, наведено приклади JSX коду розмітки наведених компонентів та розглянуто особливості використання обраних технологій, програмних засобів та сервісів, що використовувалися під час розробки додатку, а саме: мова промування JavaScript, бібліотека для розробки користувацького інтерфейсу React, фреймворк для розробки мобільних додатків React Native та платформа для розробки кросс-платформних мобільних та веб додатків Firebase.

4 ТЕСТУВАННЯ ТА ЕКСПЛУАТАЦІЯ

4.1 Тестування

Одним із найбільш важливих етапів розробки додатку є тестування. Тестування дозволяє виявити та виправити дефекти додатку, що можуть викликати збої в роботі додатку, змусити додаток працювати неправильно або не працювати зовсім. Якість додатку та відсутність в ньому дефектів є дуже важливим фактором для кінцевого користувача під час вибору додатку для навчання, тому для забезпечення якості було проведено повне тестування додатку з використанням різних видів тестування, а саме:

1. Функціональне тестування. Під час цього тестування була перевірена коректність роботи функціоналу додатку.
2. Інсталяційне тестування. Під час цього тестування була перевірена можливість інсталяції додатку на різні смартфони з різними версіями системи Android
3. Тестування зручності користування. Під час цього тестування було перевірено користувацький досвід використання додатку, а саме: чи достатньо він зрозумілий, логічний та зручний.
4. Тестування продуктивності. Під час цього тестування була перевірена здатність додатку працювати під навантаженням.

4.2 Інсталяція та системні вимоги

Інсталяція додатку на пристрій користувача відбувається за допомогою apk файлу. Для коректної роботи додатку на пристрої користувача повинна бути встановлена операційна система Android будь-якої версії починаючи з версії 4.0.0. Та вище. Також для роботи додатку необхідне підключення до мережі інтернет.

4.3 Інструкція з експлуатації програмного продукту.

Для початку роботи в додатку користувач повинен авторизуватися, або створити новий аккаунт за допомогою електронної пошти та паролю. Після створення нового аккаунту необхідно підтвердити свою пошту за допомогою посилання, що буде надіслано на неї. Після підтвердження пошти аккаунт буде активовано і можна приступати до навчання. Для початку навчання необхідно додати набір флеш-карток для вивчення. Це можна зробити за допомогою вкладки “Пошук”. В цій вкладці можна знайти набір на необхідну тему і додати його до свого аккаунту. Також можна самостійно створити власний набір за допомогою кнопки “Додати новий набір” на головній вкладці. В такому випадку, до набору потрібно буде додати необхідні елементи для вивчення. Для цього потрібно натиснути на кнопку “Додати новий елемент” і заповнити усі необхідні поля. Після цього можна починати сесію навчання. Для цього потрібно натиснути кнопку “Вчити”. Під час сесії користувачу буде показано ще не вивчені елементи та вся інформація, що пов'язана з ними. Після того як користувач запам'ятав поточний елемент можна перйти до наступного за допомогою кнопки “Далі”. Для початку сесії повторення необхідно натиснути на кнопку “Повторити”. Під час сесії повторення користувачу буде показано запитання та варіанти відповідей. Для переходу до наступного запитання потрібно вибрати варіант відповідей та натиснути кнопку “Підтвердити”. Після закінчення сесії навчання або повторення користувачу показується статистика сесії. Загальну статистику користувача можна подивитися на вкладці “Профайл”.

Змінити налаштування додатку можна на вкладці “Опції”.

4.4 Висновки до розділу

В даному розділі було розглянуто як відбувалося тестування додатку. Було описано всі типи тестування, що використовувалися для забезпечення якості та зручності використання додатку. Було подано опис способу інсталяції додатку та

його системні вимоги. Також було подано детальну інструкцію з експлуатації програмного продукту, в якій покроково розглянуто усі дії які може виконувати користувач під час використання додатку, а саме: авторизація, створення аккаунту та його підтвердження, пошук та створення наборів флеш-карток, створення нових елементів набору, вивчення та повторення елементів , перегляд статистики вивчення та зміна налаштувань додатку.

ВИСНОВКИ

Під час виконання цієї дипломної роботи було проаналізовано існуючі програмні продукти для вивчення інформації і було описано їх детальну характеристику та переваги й недоліки. На основі отриманої інформації було спроектовано та розроблено універсальний мобільний додаток для швидкого вивчення інформації. Даний додаток використовує переваги мобільної системи Android та найбільш дієві методи прискорення запам'ятовування. Для розробки додатку були використані сучасні технології, методи та інструменти розробки. Для забезпечення високої якості та зручності використання додатку, його було ретельно протестовано. В результаті було отримано мобільний додаток, що дозволяє користувачам вивчати будь-яку інформацію максимально швидко та ефективно.

ПЕРЕЛІК ПОСИЛАНЬ

1. Android Open Source Project [Електронний ресурс]. – Режим доступу: https://source.android.com/setup/start/faqs
2. Number of Android apps on Google Play [Електронний ресурс]. – Режим доступу: https://www.appbrain.com/stats/number-of-android-apps
3. How to install the Android SDK [Електронний ресурс]. – Режим доступу: https://www.androidauthority.com/how-to-install-android-sdk-software-development-kit-21137/
4. Kotlin is now Google’s preferred language for Android app development [Електронний ресурс]. – Режим доступу: https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/
5. Android NDK Native APIs [Електронний ресурс]. – Режим доступу: https://developer.android.com/ndk/guides/stable_apis.html
6. Android Common Kernels [Електронний ресурс]. – Режим доступу: https://source.android.com/devices/architecture/kernel/android-common
7. Meet ART, Part 1: The New Super-Fast Android Runtime Google Has Been Working On In Secret For Over 2 Years Debuts In KitKat [Електронний ресурс]. – Режим доступу: https://www.androidpolice.com/2013/11/06/meet-art-part-1-the-new-super-fast-android-runtime-google-has-been-working-on-in-secret-for-over-2-years-debuts-in-kitkat/
8. A JIT Compiler for Android's Dalvik VM [Електронний ресурс]. – Режим доступу: https://web.archive.org/web/20150714081347/http://www.android-app-developer.co.uk/android-app-development-docs/android-jit-compiler-androids-dalvik-vm.pdf
9. Google confirms next Android version will use Oracle’s open-source OpenJDK for Java APIs [Електронний ресурс]. – Режим доступу: https://venturebeat.com/2015/12/29/google-confirms-next-android-version-wont-use-oracles-proprietary-java-apis/

10. JavaScript [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript
11. Usage statistics of JavaScript libraries for websites [Електронний ресурс]. – Режим доступу: https://w3techs.com/technologies/overview/javascript_library
12. React [Електронний ресурс]. – Режим доступу: https://reactjs.org/
13. React: Making faster, smoother UIs for data-driven Web apps [Електронний ресурс]. – Режим доступу: https://www.infoworld.com/article/2608181/react--making-faster--smoother-uis-for-data-driven-web-apps.html
14. JSX Specification [Електронний ресурс]. – Режим доступу: https://facebook.github.io/jsx/
15. In Depth OverView [Електронний ресурс]. – Режим доступу: https://facebook.github.io/flux/
16. In Depth OverView [Електронний ресурс]. – Режим доступу: https://facebook.github.io/flux/
17. React Native [Електронний ресурс]. – Режим доступу: https://reactnative.dev/
18. What Is React Native? [Електронний ресурс]. – Режим доступу: https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html
19. React Native for Android: How we built the first cross-platform React Native app [Електронний ресурс]. – Режим доступу: https://engineering.fb.com/2015/09/14/developer-tools/react-native-for-android-how-we-built-the-first-cross-platform-react-native-app/
20. How React Native Can Empower Your Mobile App Development Process [Електронний ресурс]. – Режим доступу: https://www.xongolab.com/blog/how-react-native-can-empower-your-mobile-app-development-process/

21. Bridging in React Native [Електронний ресурс]. – Режим доступу: https://tadeuzagallo.com/blog/react-native-bridge/
22. Firebase [Електронний ресурс]. – Режим доступу: https://firebase.google.com/
23. Firebase expands to become a unified app platform [Електронний ресурс]. – Режим доступу: https://firebase.googleblog.com/2016/05/firebase-expands-to-become-unified-app-platform.html

ДОДАТОК

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА ANDROID ДОДАТКУ ДЛЯ ШВИДКОГО ВИВЧЕННЯ ІНФОРМАЦІЇ МОВОЮ JS

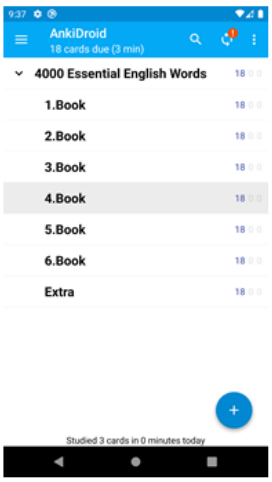
Виконав студент 4 курсу
групи ПД-41
Пащенко В. Ю.
Керівник роботи
Дібрівний О. А.

Київ – 2020

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи:** розробка Android додатку для швидкого вивчення інформації.
- **Об'єкт дослідження:** процес вивчення інформації за допомогою мобільних додатків.
- **Предмет дослідження:** застосування сучасних методів та засобів розробки програмного забезпечення для розробки Android додатку для швидкого вивчення інформації.

ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА АНАЛОГІВ



AnkiDroid

Основні переваги додатку:
- Підтримка різних видів матеріалів для флеш-карток, таких як текст, зображення та аудіо файли.
- Використання методу інтервального повторення.
- Можливість використання синтезу тексту в мовлення.
- Велика кількість безкоштовних наборів флеш-карток.
- Надання користувачу детальної статистики та можливості відслідковувати свій прогрес в навчанні.

Основні недоліки додатку:
- Відсутня підтримка відео файлів в якості матеріалів для флеш-карток.
- Відсутня можливість автоматичного пошуку відповідних медіа матеріалів для флеш-карток.
- Завантаження наборів флеш-карток відбувається через веб версію додатку і займає дуже багато часу.

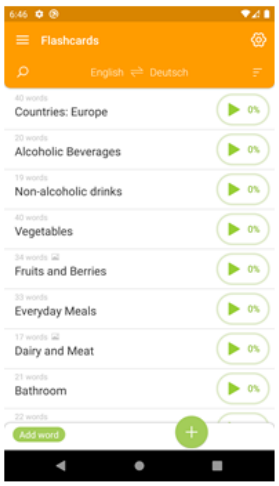
ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА АНАЛОГІВ

Memrise

Основні переваги додатку:
- Підтримка різних видів матеріалів для флеш-карток, таких як текст, зображення та аудіо файли.
- Використання методу інтервального повторення та ігрофікації.
- Синхронізація між різними пристроями.
- Велика кількість курсів для вивчення.

Основні недоліки додатку:
- При використанні android версії додатку відсутня можливість створювати власні курси та користуватись курсами інших користувачів.
- При створенні власних курсів відсутня підтримка відео файлів в якості матеріалів для флеш-карток.
- Відсутня можливість автоматичного пошуку відповідних медіа матеріалів для флеш-карток та синтезу тексту в мовлення.
- Велика частина матеріалів курсів та функціоналу додатку доступа лише з платною підпискою.

ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА АНАЛОГІВ



Lexilize Flashcards

Основні переваги додатку:

- Використання методу інтервального повторення.
- Додаток працює без підключення до Інтернету.
- Можливість використання синтезу тексту в мовлення.
- Можливість автоматичного пошуку відповідних медіа матеріалів для флеш-карток.

Основні недоліки додатку:

- Відсутня підтримка відео та аудіо файлів в якості матеріалів для флеш-карток.
- Статистика доступна тільки з платною підпискою.
- Дуже мала кількість флеш-карток в стартових наборах та відсутність можливості пошуку та використання наборів створених іншими користувачами.
- Відсутня можливість синхронізації наборів флеш-карток між різними пристроями.

5

ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА АНАЛОГІВ

	AnkiDroid	Memrise	Lexilize Flashcards	Мій додаток
Використовує метод інтервального повторення	Так	Так	Так	Так
Використовує метод гейміфікації	Ні	Так	Ні	Так
Надає готові набори флеш-карток або можливість використовувати набори інших користувачів	Так	Так	Так, але кількість флеш-карток в наборах дуже мала	Так
Підтримує різні види медіа матеріалів	Зображення, аудіо	Зображення, аудіо	Зображення, аудіо	Зображення, аудіо, відео
Автоматично генерує медіа матеріали	Ні	Ні	Так	Так
Використовує синтез тексту в мовлення	Так	Ні	Так	Так
Надає статистику навчання	Так	Тільки з платною підпискою	Тільки з платною підпискою	Так
Синхронізує дані між різними пристроями	Використовує сторонній сервіс	Так	Ні	Так

6

ТЕХНІЧНІ ЗАВДАННЯ

Android додаток для швидкого вивчення інформації.

- *1. Основний функціонал додатку*
- *2. Інтерфейс користувача*
- *3. Створення та інтеграція бази даних*
- *4. Тестування додатку*

7

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Android

Android – це операційна система для сенсорних мобільних пристроїв, розроблена на основі модифікованої версії ядра Linux. Android розробляється консорціумом розробників Open Handset Alliance, і комерційно фінансується компанією Google.

Android є найбільш популярною операційною системою у світі для смартфонів з 2011 року, а для планшетів - з 2013 року. Станом на травень 2017 року операційною системою користуються понад два мільярди активних користувачів і це робить Android системою з найбільшою базою користувачів серед усіх операційних систем.

8

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

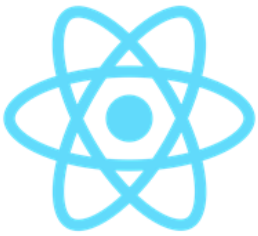
JavaScript

JavaScript – це легка, високорівнева, інтерпретована, об'єктно-орієнтована мова з першокласними функціями і найвідоміша як мова сценаріїв веб-сторінок, але вона використовується і в багатьох середовищах, що не належать до браузера.

Це мова сценаріїв, що базується на прототипах, з використанням багатьох парадигм, яка є динамічною та підтримує об'єктно-орієнтований, імперативний та функціональний стилі програмування.

9

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



React та React Native

React, також відомий як React.js та ReactJS – це JavaScript бібліотека для створення інтерфейсу користувача з відкритим кодом. React підтримується компанією Facebook та спільнотою окремих розробників та компаній. React можна використовувати для розробки веб-додатків або мобільних додатків.

React Native – це фреймворк для розробки мобільних додатків з відкритим кодом, створений компанією Facebook. Він дозволяє розробникам використовувати для створення додатку фреймворк React та усі можливості платформи для якої розробляється додаток.

10

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Firebase

Firebase – це платформа для створення мобільних та веб-додатків, що розробляється компанією Google.
Платформа Firebase надає продукти та рішення які можна використовувати для розробки додатків.

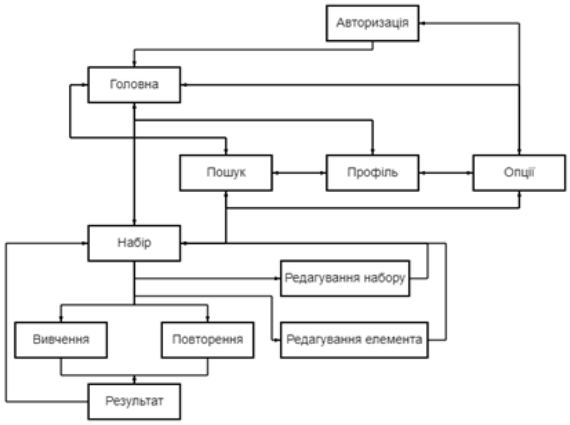
Firebase підтримує велику кількість платформ та мов програмування. За допомогою Firebase можна створювати Android, iOS та веб-додатки використовуючи такі мови програмування, як C++, JavaScript, Java, Kotlin, Objective-C, та Swift.

Firebase

11

ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Діаграма варіантів використання

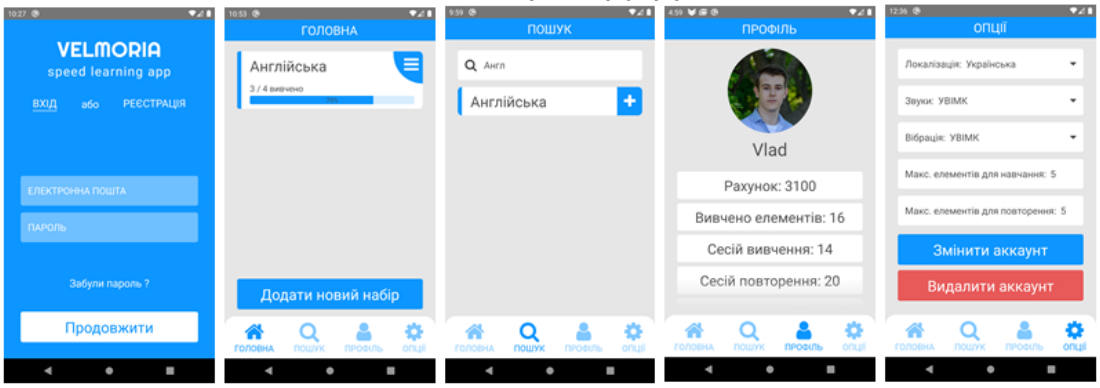


Структура навігації в додатку

12

ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

ОСНОВНІ СЦЕНИ ДОДАТКУ

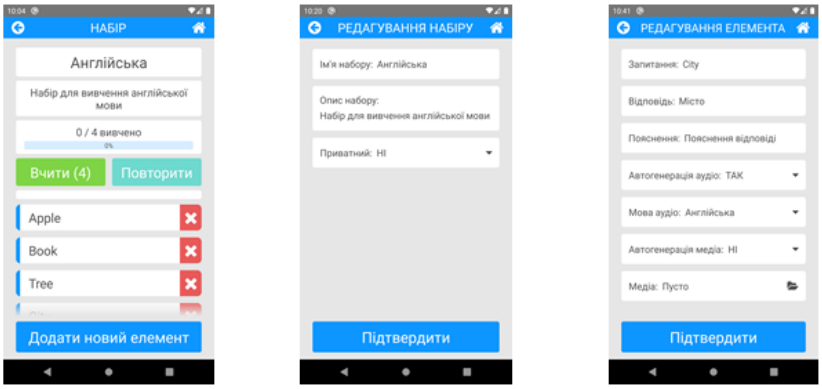


Авторизація | Головна | Пошук | Профіль | Опції

13

ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

СЦЕНИ ДЛЯ СТВОРЕННЯ ТА РЕДАГУВАННЯ НАБОРІВ

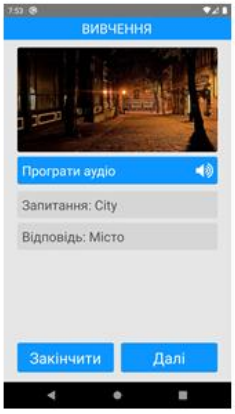


Набір | Редагування набору | Редагування елемента

14

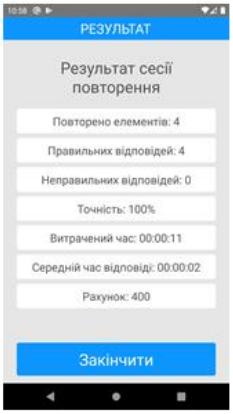
ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

СЦЕНИ ДЛЯ ВИВЧЕННЯ ІНФОРМАЦІЇ



Вивчення

Повторення



Результат

15

ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

СТРУКТУРА ДОКУМЕНТІВ БАЗИ ДАНИХ

Set
description
isPrivate
items
keywords
learnedItems
name
numberOfItems
owner
setID

Документ набору

User
email
itemsLearned
learnSessions
name
photoURL
reviewSessions
score
timeLearning
uid

Документ користувача

Statistics
date
itemsLearned
itemsReviewed
learnSessions
reviewSessions
timeLearning
user

Документ статистики

16

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

- Пащенко В. Ю. Дослідження ефективності використання методів гейміфікації в освітніх мобільних додатках / Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інфокомунікаційних технологіях» : Теорія надійності програмноапаратних систем. Збірник тез. 12.02.2021, ДУТ, м Київ – С 91.

- Пащенко В. Ю. Розробка мобільного додатку для швидкого вивчення інформації мовою програмування JavaScript / Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IOT» : Сучасні інформаційні технології в україні і світі. Збірник тез. 09.04.2021, ДУТ, м Київ – С 23.

17

ВИСНОВКИ

1. Проаналізовано існуючі аналоги і описано їх детальну характеристику та переваги й недоліки.
2. На основі отриманої інформації спроектовано та розроблено мобільний додаток.
3. Для забезпечення якості та зручності використання додатку, його було ретельно протестовано.
4. В результаті було отримано мобільний додаток, що дозволяє користувачам вивчати будь-яку інформацію максимально швидко та ефективно.

У подальшому планується:
1. Розробити версії додатку для інших платформ.
2. Розробити веб версію додатку.
3. Додати можливість автоматичного перекладу флеш-карток.

18

ДЯКУЮ ЗА УВАГУ!