Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Факультет інформатики

Кафедра мультимедійних систем

Розробка мобільного застосунку для навчання за допомогою флеш-карток

Текстова частина до курсової роботи
за спеціальністю «Комп’ютерні науки»

Керівник курсової роботи:
Калітовський Б. В.

Виконала студентка:
Оленин С.

Київ 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Жежерун О.П.

_____

(підпис)

“____”_____2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студентки Оленин Софії Ігорівни факультету інформатики 3 курсу

Тема: Розробка мобільного застосунку для навчання за допомогою флеш-карток.

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

1. Аналіз предметної області. Постановка завдання курсової роботи

2. Теоретичні відомості

3. Опис реалізації додатку

Висновки

Перелік використаних джерел

Дата видачі “___” _____ 2021 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	20.10.2020	
2.	Огляд технічної літератури за темою роботи.	10.11.2020	
3.	Аналіз актуальності та дослідження аналогів.	24.02.2021	
4.	Написання практичної частини.	05.03.2001	
5.	Написання пояснювальної роботи.	20.04.2021	
5.	Попередня демонстрація роботи науковому керівнику	09.05.2021	
7.	Створення презентації	15.05.2021	
8.	Захист курсової роботи		

Студент _____

Керівник _____

“_____”_____

Зміст

Перелік використаних скорочень та термінів

Флеш-картка - це картка, що містить інформацію з обох сторін. З одної термін чи слово, з іншої відповідно визначення чи переклад. Використовується для запам'ятовування інформації.

ОС – операційна система.

JS – JavaScript.

TS – TypeScript.

СКБД - система керування базами даних.

БД – база даних.

Анотація

У роботі розглянуто популярні додатки для навчання, завдяки чому визначено ключові функції застосунку.

Також проаналізовано найпоширеніші засоби розробки мобільних додатків, обґрунтовано вибір тих, що будуть використовуватися. А саме обрано наступний стек технологій: фреймворк React Native, мову програмування Java Script, середовище розробки WebStorm, системе контролю версій Git, платформу Expo. В якості бази даних використовується Firebase Realtime Database.

Розроблено навчальний сервіс, який надає можливість вивчати інформацію за допомогою флеш-карток, а також здійснювати тестування по них.

ВСТУП

Актуальність теми:

Питання вивчення нового матеріалу є завжди актуальним, адже люди продовжують навчатися протягом усього життя: спочатку в школі, потім в університеті, далі підвищують свою кваліфікацію на роботі, дізнаються нове для загального розвитку, тощо. Проблема полягає в тому, як саме ефективно засвоювати нову інформацію. З цим питанням я зіткнулася в університеті, коли матеріалу багато, а часу обмаль. Одним із найпродуктивніших способів для себе, я виділила флеш-картки (прототип карток, де з одного боку розташоване слово\термін, а з іншого пояснення\переклад\тощо). Надзвичайно зручно використовувати не паперовий варіант, а його прототип у мобільному додатку. Це дозволяє структурувати знання та мати доступ до них в будь-який час просто з свого смартфону, що дуже вигідно, бо непотрібно шукати необхідний матеріал у великих зошитах, купі аркушів, особливо це зручно в непридатних для цього місцях, наприклад в транспорті, черзі в поліклініці, тощо.

Мета та завдання дослідження:

Мета: створення сервісу для зберігання і засвоєння інформації, що полегшить користувачам навчання, а також зробить цей процес зручним та ефективним.

Завдання: розробка навчального мобільного додатку із зручним інтерфейсом, де користувачі зможуть вивчати інформацію за допомогою флеш-карток, а також здійснювати тестування по них.

***Об’єкт дослідження*:**

Інші сервіси, що надають змогу структурувати і вивчати інформацію за допомогою флеш-карток.

Методи дослідження.

- спостереження – пошук сервісів, що надають змогу структурувати і вивчати інформацію, проходити тестування по ній.

- порівняння – розбір кожного додатку, порівняння зручності, функцій і можливостей.

- аналіз –розкладання зібраної інформації дослідження на складові частини.

В якості джерел порівняння я використала сервіси: Chegg Prep, Quizlet, AnkiApp.

Причиною вибору перших двох сервісів була велика популярність серед користувачів, а також мій досвід користування додатками. Останній я обрала, для того щоб проаналізувати щось нове і не схоже на аналоги.

Робота складається з трьох розділів.

У першому проводиться аналіз предметної області та постановка задачі.

Другий розділ присвячений розгляду теоретичних відомостей щодо розробки мобільних додатків. Обґрунтовано вибір стеку технологій розробки.

У третьому розділі описується реалізація додатку.

Також зроблено висновки щодо роботи.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.

1.1. Аналіз сучасного стану питання та обґрунтування теми

В сучасному світі смартфони стали невід'ємною частиною нашого життя, і майже кожен завжди тримає його під рукою. За допомогою цього девайсу ми маємо можливість спілкуватися, розважатися, а також навчатися. Вивчати нову інформацію за допомогою смартфону зручніше ніж, наприклад, за допомогою ноутбука чи планшету. Адже малі розміри гаджету дозволяють користуватися ним будь-де. Тому наразі існує безліч додатків, які надають змогу ефективно засвоювати нову інформацію за допомогою смартфону. Саме така популярність і необхідність мобільних застосунків послужили причиною вибору моєї теми і я захотіла створити мобільний додаток для навчання.

1.2. Огляд аналогів

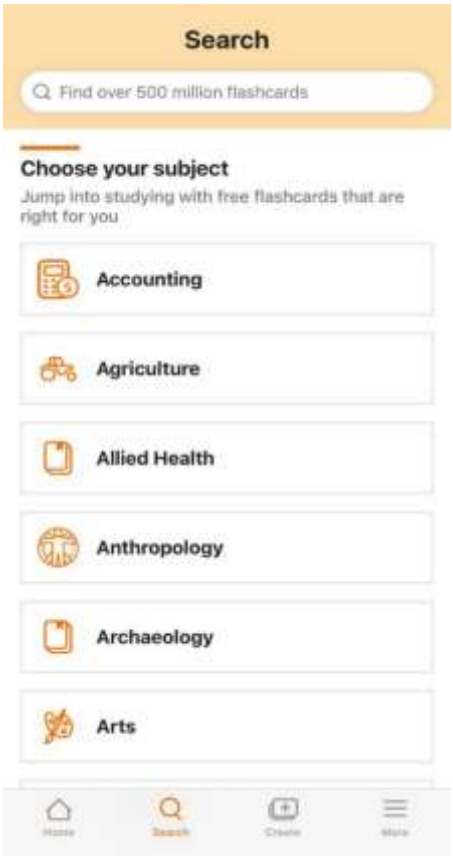
Розпочнемо розгляд аналогів із Chegg Prep (у минулому StudyBlue, проте в 2012 була викуплена компанією Chegg) [1].

Як стверджує офіційний сайт [2]:

Chegg Prep - це потужний безкоштовний інструмент для самонавчання, який можна використовувати для створення, вивчення та обміну картками.

Він надає змогу :

- створювати власні картки або вивчати завдання, зроблені іншими користувачами;
- шукати понад 500 мільйонів колод, щоб знайти ті, які найбільш відповідають вашим потребам;
- бачити бал, щоб знати, з яким успіхом була опрацьована кожна колода, що допомагає відслідковувати прогрес.



Під час користування додатком, найбільшою перевагою я виділила пошук колод по темах. Адже не завжди знаєш, як саме буде називатися збірка карток, а в секції пошуку додатку, можна знайти багато нової інформації за вибраною тематикою (Рис. 1.2.1).

З недоліків: відсутність можливості об'єднувати колоди в колекції. Всі збірки карток зберігаються в одному місці, що часом ускладнює пошук необхідної інформації. Адже структуризація дуже важлива при навчанні.

Рис. 1.2.1 Секція пошуку колод карт

Наступним аналогом є Quizlet, який доступний у вигляді мобільного та веб-додатку. Це найпопулярніший сервіс для вивчення, який дозволяє користувачам вивчати різні теми за допомогою карток та безлічі ігор, тестів [3].

Крім більшості функцій, які наявні в Chegg Prep, Quizlet також надає змогу групувати сети (сукупність карт) у папки, додавати їх до певного класу та вивчати інформацію за допомогою наступних інструментів:

- Gravity. У цьому режимі вивчення, визначення прокручують вертикально вниз по екрану у формі астероїдів. Користувач повинен ввести термін, що відповідає визначенню, перш ніж він досягне нижньої частини екрана. (Доступний лише у веб-додатку).
- Write. У цьому режимі користувачам показується термін, і вони повинні ввести визначення, що відповідають показаному. (Рис. 1.2.2)

- Speller. У цьому режимі термін озвучується вголос, і користувачі повинні правильно його написати. (Доступний лише у веб-додатку)
- Match. У цьому режимі користувачам представлена сітка розрізнених термінів. Користувачі перетягують терміни поверх відповідних визначень, щоб видалити їх із сітки та спробувати очистити сітку якомога швидше. (Рис. 1.2.3)
- Live. У цьому режимі навчання користувач Quizlet (як правило, викладач) розбиває свій клас на команди, щоб грати у гру зі студентами окремо.

Отже, Quizlet це справді потужний сервіс для навчання, який містить багато корисних інструментів. Проте, як на мене, він є дуже зручним і ефективним лише для вивчення іноземних слів. Для засвоєння термінів, нового навчального матеріалу доцільніше використовувати простіші додатки, які не відволікають від основного надмірною рекламою та функціями.

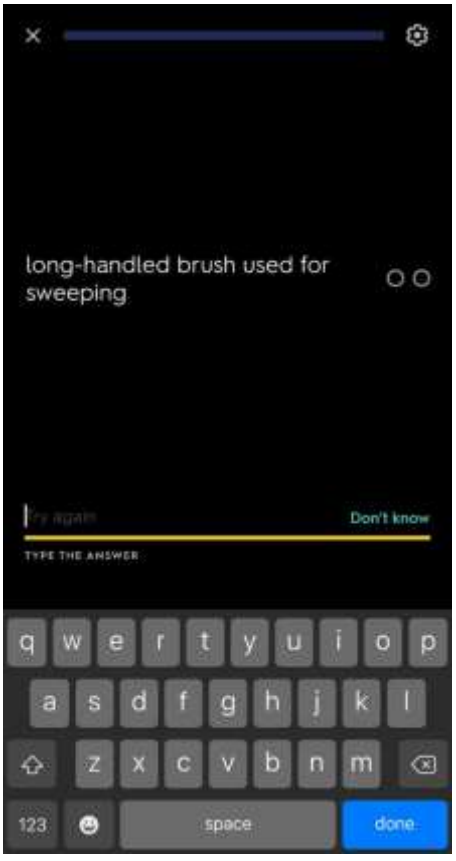


Рис. 1.2.2 – навчання у режимі Write

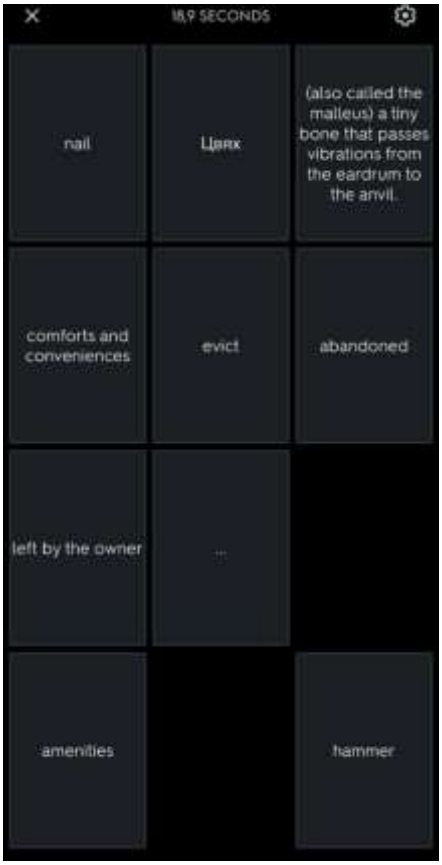


Рис. 1.2.3 – навчання у режимі Match

Останнім аналогом, який я хочу розглянути, є AnkiApp

Згідно з офіційним сайтом [4] додаток надає можливість створювати картки з текстом, звуком та зображенням або додавати вже готові. Навчання є надзвичайно ефективним завдяки унікальному алгоритму. Сервіс автоматично робить резервні копії та синхронізує всі пристрої .

Після використання, я виявила такі переваги: під час вивчення користувач не просто вказує, чи вивчив чи ні термін, а оцінює свої знання по шкалі (Рис. 1.2.4.). Після чого алгоритм виявляє, наскільки добре користувач знає кожну флеш-карту, потім визначає їх пріоритетом, тож вивчається те, знання чого мінімальні, не витрачаючи дорогоцінного часу на те, що вже вивчено. (Рис 1.2.5)

Рис 1.2.4 – процес вивчення картки

Рис 1.2.5 – статистика після вивчення

Ще однією перевагою є можливість додавати не тільки текст чи картинку, а й звук.

Проте, під час використання, я зіткнулася з дуже вагомим недоліком: незручний і незрозумілий інтерфейс, а також неприємний, застарілий дизайн (Рис.1.2.5, Рис.1.2.6), які переважують переваги і відбивають бажання користуватися додатком.

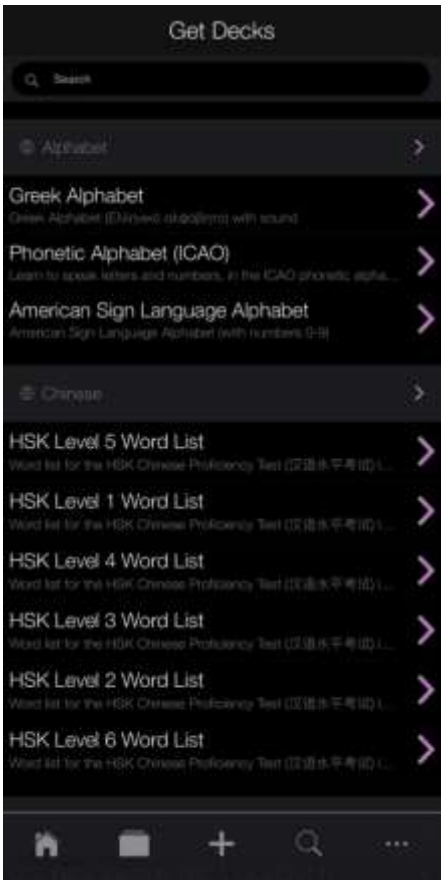


Рис. 1.2.6 - створення картки

Рис. 1.2.7 – пошук вже створених карток

Проаналізувавши три додатки можна зробити порівняльний аналіз. Отже, для вивчення нового матеріалу, як на мене, найкраще підходить Chegg Prep, адже це лаконічний, зручний сервіс, проте в ньому неможливо структурувати колоди карт. Щодо Quizlet, то порівнюючи з першим додатком, він має набагато більший функціонал, і є ефективним, особливо для вивчення та повторення іноземних слів.

Третій застосунок вирізняється наявністю алгоритму для оцінки рівня знань, проте має незручний інтерфейс.

1.3. Постановка завдання

Як зазначалося вище мобільні додатки для навчання є дуже популярними, проте знайти зручний сервіс без надлишку функцій, реклами не просто. Тому я хочу створити лаконічний мобільний додаток із зручним, інтуїтивно зрозумілим інтерфейсом, приємним дизайном, який матиме наступний функціонал:

- створення та редагувати власних колод з флеш-картами ;
- групування колод в колекції ;
- вивчення колод карт, переглядаючи кожну флеш-карту, відмічаючи чи засвоєно її, на основі чого рахується фінальний бал, наскільки добре користувач знає матеріал ;
- проходити тестування по картках;
- шукати та переглядати колоди інших користувачів за іменем чи тегом ;
- можливість додавати собі колоди інших користувачів ;
- створення та редагування облікового запису користувача.

1.4. Висновок до розділу

Проаналізовано актуальність теми, предметну область, обґрунтовано тему. Розглянуто найпопулярніші аналоги, визначено їхні переваги і недоліки, завдяки чому з'ясовано ключові функції, які необхідні для додатку.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ВІДОМОСТІ.

2.1. Аналіз засобів розробки мобільних додатків.

Після постановки завдання важливо уважно проаналізувати технології розробки, щоб обрати ті, які точно задовольнятимуть потреби сервісу.

Розпочнемо аналіз із Java. Вона є мовою, яка активно використовується для створення додатків Android. Java пропонує безліч бібліотек з відкритим кодом, хорошу документацію та підтримку спільноти, що допомагає розробникам ефективно створювати різні типи програм для Android [5]. Проте критики цієї мови стверджують, що Java потребує багато "шаблонного" коду, щоб виконати просте завдання, і такі поняття як винятки важко зрозуміти [6].

Наступною мовою є Kotlin. Вона використовується для розробки високо досконалих мобільних додатків. Для розробників Android ця мова є сучасною відповіддю на застарілу Java, і є її вдосконаленою версією. Безпека, чіткість та велика підтримка інструментів - це функції, які роблять Kotlin кращим вибором для створення додатків для Android. Trello, Coursera та Evernote - це деякі програми, створені за допомогою Kotlin.[5] Немає сумнівів, що у багатьох випадках, таких як виконання покрокових збірок, вона швидша за Java. Однак Java залишається явним переможцем, коли справа доходить до створення чистих збірок для додатків Android. [7]

Попередньо було розглянуто мови програмування для ОС Android. А от для розробки під iOS найпопулярнішою є Swift, розроблена Apple Inc. для macOS, watchOS, tvOS, Linux, iPadOS та деяких інших платформ. Мова цілеспрямовано працює з фреймворками Cocoa та Cocoa Touch та кодами C та Objective C, написаними виключно для розробки програм для iOS. Swift використовує безпечний шаблон програмування та пропонує виняткові функції, щоб зробити повний процес розробки мобільних додатків iOS плавним. [5] Проте вона має ряд недоліків. По-перше, часто Swift вважається нестабільною через серйозні зміни, які

вносяться з кожним оновленням. Однією з основних проблем, зазначену багатьма розробниками, є відсутність зворотної сумісності зі старішими мовними версіями. Тобто розробники змушені повністю переписати свої проекти, якщо хочуть перейти на останню версію Swift. По-друге, відсутність підтримки попередніх версій iOS. Swift не можна використовувати для застарілих проектів, що працюють на старих версіях операційної системи.[8]

Вище було проаналізовано технології для нативних додатків, тобто розроблених для використання для конкретної платформи. Але свій додаток я вирішила зробити гібридним (або кросплатформленим), тобто створеним для декількох платформ одночасно, при цьому маючи однакову функціональність. Мій вибір спричинений тим що:

- додаток працює одразу на декількох платформах;
- більше людей можуть ним користуватися;
- швидка розробка.

Тепер розгляньмо варіанти для розробки гібридних додатків.

Розпочнемо з Xamarin – це інструмент, що використовується для розробки міжплатформенних мобільних додатків. Вона створена, щоб розробники .NET могли створити сучасні та ефективні програми для Android, iOS та Windows. Xamarin відповідає за управління зв'язком між загальним кодом і кодом платформи. Завдяки ньому розробники можуть розподіляти близько 90% своїх програм на всіх основних платформах. Така можливість означає, що програмісти можуть писати всю свою бізнес-логіку, використовуючи одну мову, одночасно отримуючи відчуття, подібні до рідного, на кожній платформі. Серед переваг можна виділити: повна технічна підтримка (наприклад, камера, GPS), сумісність з архітектурами MVC та MVVM, наявність повної екосистеми розробки (C #, .NET та Visual Studio), також Xamarin з відкритим кодом та безкоштовний. Щодо недоліків, то він має мінімальну підтримку спільноти, великий розмір додатку, не є хорошим вибором

для програм зі складним інтерфейсом, бо розробка користувальницького інтерфейсу займає багато часу та не є зручною для мобільних пристроїв.[9]

Наступним є React Native, фреймворк з відкритим кодом, що використовується для розробки кросплатформенних додатків для iOS, Android, Web та UWP. React Native використовує ті самі основні блоки інтерфейсу користувача, що і звичайні програми для iOS та Android, а розробники складають блоки разом за допомогою JavaScript і React. React Native представляє новий, радикальний та високофункціональний підхід до побудови користувальницьких інтерфейсів. З ним логіка додатка пишеться та працює у JavaScript, тоді як користувальницький інтерфейс додатка є повністю нативним (тобто рідним). [8] І саме React Native я обрала для написання свого сервісу, адже він має безліч переваг серед яких[9]: доступ до нативних функцій (наприклад камера), безкоштовний, легкий у вивченні, швидший процес розробки завдяки попередньо встановленим елементам, перегляд результату в режимі реального часу, а також, порівняно з Xamarin, React Native має значно більшу спільноту розробників, про що свідчить статистика на GitHub[10]. Отже, це хороший вибір для написання мого додатку.

Фреймворк дозволяє використовувати JavaScript або TypeScript.

JavaScript це скриптова мова програмування, яка має динамічну типізацію.[11]

Переваги:

- Простота. Мова є порівняно проста у вивченні та реалізації.
- Популярність.
- Висока швидкість роботи.
- Наявність багатьох бібліотек з необхідним функціоналом.

Недоліки:

- Відсутність засобів налагодження (Debugging Facility).
- Одиночне успадкування.

- Клієнтська безпека. Оскільки код виконується на комп'ютері користувачів, в деяких випадках його можна використовувати для зловмисних цілей.

TypeScript - це мова з відкритим кодом, яка базується на JavaScript, додаючи визначення статичного типу.[12]

Переваги:

- Самодокументація - пишучи типи, ви документуєте, які значення ви очікуєте від функції / компонента.
- TS має інтеграцію майже з усіма редакторами.
- Додає структуру до великих додатків.
- Гарантовано дійсний JS - якщо у вас є дійсний TypeScript, ви знаєте, що отримаєте дійсний вихідний код JavaScript, який працюватиме на будь-якому вказаному вами цільовому рівні.

Недоліки:

- Повільніша розробка, оскільки доводиться прописувати типи.
- Додаткові анотації роблять файли TS більшими, ніж ті, що написані простою JS.
- Помилкове почуття безпеки. На мій погляд, найбільшим недоліком TypeScript є те, що він може принести вам хибне відчуття безпеки. Так, це величезна перевага. Однак покладання на те, що мова може перевіряти типи для нас і попереджати нас, коли з нашим кодом щось не так, занадто сильно спричиняє значний ризик: складається враження, що весь код на TS, є точно захищеним.

2.2. Обґрунтування вибору засобів розробки

Для додатка я обрала фреймворк React Native та мову програмування JS, аналіз та причини використання яких було розглянуто вище.

Тепер обґрунтую вибір інших засобів розробки.

Налаштовувати проект збираюся за допомогою Expo. Це інструмент, який допомагає розробляти, будувати, розгортати та швидко переглядати iOS, Android та веб-програми.[13] Обрала його тому що він:

- простий у використанні;
- надає можливість перегляду результату в режимі реального часу;
- має хорошу документацію;
- тестування безпосередньо на пристрої як для iOS, так і для Android.

Для середовища розробки вибрала WebStorm. Це потужне середовище розробки, яке надає допомогу в кодуванні JavaScript, HTML та CSS та широкий спектр сучасних веб-технологій. Інструмент розроблений на основі платформи IntelliJ IDEA та є платним. Але можна оформити безплатну студентську підписку.

Обрала його, адже він забезпечує:

- розумний аналіз коду;
- автозаповнення;
- функції рефакторингу;
- попередження помилок;
- широку навігацію та пошук;
- інтеграцію інструментів.

Для контролю версій використовую Git. А саме обрала веб-сервіс GitHub. Тому що завдяки ньому зручно відслідковувати прогрес написання програми, всі файли зберігаються, можливість повернутися до попереднього “коміту” в разі “поломки” програм під час розробки

2.3. Вибір СКБД

Обравши основні засоби розробки я приступила до одного з найважливіших етапів: вибору СКБД.

СКБД (система керування базами даних) – це програмне забезпечення, за допомогою якого користувачі можуть визначати, створювати та підтримувати базу даних, а також здійснювати контрольований доступ до неї.

Важливість цього вибору полягає в тому, що не зважаючи на те, що всі СКБД забезпечують такі функції:

- дозволяти створювати БД,
- дозволяти додавання, оновлення, видалення та вибірку інформації з БД,
- надавати контрольований доступ до БД,

сам процес виконання цього завдання варіюється в широких межах.

Тому важливо вибрати саме таку, яка чудово задовольнятиме потреби додатку.

Розгляньмо декілька СКБД.

Почнемо з AWS DynamoDB. DynamoDB - це база даних NoSQL на стороні сервера, яка призначена тільки для платформи Amazon Web Service. Ним можна легко керувати, оскільки він повністю децентралізований.

DynamoDB надає безкоштовно перші 25 ГБ, а потім буде здійснюватися плата відповідно до ГБ, споживаної базою даних.[15]

Переваги:

- простий в налаштуванні;
- простий в інтеграції;
- Інтегровано з більшістю служб AWS;
- Практично безкінечне сховище.

Недоліки:

- Відсутність складник запитів. Оскільки БД не дозволяє традиційні запити SQL та шаблони доступу, а це означає, що для таких речей, як пошук вільного тексту або спеціальні запити, ймовірно, доведеться експортувати дані в іншу систему.
- Важко оцінити вартість користування. Важче передбачити витрати і часто можна натрапити на несподівані витрати.[16]

Наступною розглянемо MongoDB. Це орієнтована на документи база даних. Дані організовані як документи JSON (еквівалент рядків) з полями (еквівалент стовпців), які згруповані в колекції (еквівалент таблиць).[14]

Переваги:

- гнучка (дає гнучкість і свободу зберігати дані різних типів);
- має можливість зберігати великі дані, розподіляючи їх на декількох серверах, підключених до програми;
- має високу швидкість запитів;
- наявна підтримка спеціальних запитів;
- просте налаштування навколишнього середовища.

Недоліки:

- не підтримує об'єднання (як у реляційних БД);
- зберігає імена ключів для кожної пари значень, через що існує надмірність даних, що призводить до збільшення непотрібного використання пам'яті.[18]

Розглянемо ще Firebase Realtime Database. Це хмарна база даних NoSQL, яка дозволяє зберігати та синхронізувати дані між своїми користувачами в режимі реального часу.

Переваги:

- може синхронізувати всі дані в режимі реального часу для всіх клієнтів одночасно, що важливо для програми, яка переходить у автономний режим через відсутність підключення до Інтернету;
- більшість послуг платформи є безплатними, плата необхідна лише після досягнення певного обсягу пам'яті бази даних;
- легко і зручно інтегрувати;
- пропонує кросплатформенний API, який потребує мінімальних налаштувань, коли він використовується як частина програми. Тепер не потрібен окремий сервер для доступу до даних, оскільки до них можна безпосередньо отримати доступ через програму.

Недоліки:

- обмежена можливість запитів;
- обмежена міграція даних;
- складно визначити точну вартість (так само як в AWS) ;
- не працює в країнах, які не дозволяють Google.

Отже, проаналізувавши вище зазначені СКБД можна зазначити, що всі вони є хорошими продуктами зі своїми плюсами і мінусами. Але опираючись на вимоги мого додатку, я зупинила свій вибір на Firebase Realtime Database. Адже платформа допомагає зручно зберігати та редагувати динамічний вміст, що прискорює процес розробки, дозволяючи отримувати швидші результати.

2.4. Висновок до розділу

Проаналізовано найпопулярніші і найпоширеніші засоби розробки мобільних додатків, обґрунтовано вибір тих, що будуть використовуватися. А саме обрано наступний стек технологій: фреймворк React Native, мову програмування Java Script, середовище розробки WebStorm, системе контролю версій Git, платформу Expo. Також вирішено використовувати базу даних Firebase Realtime Database.

РОЗДІЛ 3. ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТУ.

3.1. Аналіз технічного завдання

Проаналізуймо технічне завдання описане в пункті 1.3.

Користувач має змогу створювати колоду карт із флеш-картами. Він заповнює дані про кожну картку: передню і задню сторону картки (текст і/або зображення). Тобто термін – визначення, слово – переклад, тощо. Під час створення колоди, є можливість редагувати чи видаляти ці дані. Також користувач заповнює поле із назвою колоди і вказує, чи вона буде видимою для інших чи ні.

Користувач може переглянути всі свої колоди (створені ним або додані від інших користувачів). При натиску на одну з них, він потрапляє на екран колоди, де має можливість редагувати або видалити її, переглянути вже існуючі картки, редагувати їх або ж видалити. Вказати бажану сталу кількість карток для вивчення. Також там зберігається оцінка, яка свідчить про те, як добре користувач вивчив дану дошку. Оцінка вираховується як відсоток вивчених карток від їх загальної кількості.

Користувач може опрацювати флеш-картки, зазначаючи чи вивчив він картку чи ні. Також може опрацювати лише ті, які не були вивчені. Також він має змогу опрацьовувати лише ті, які знає найменше. Вони визначаються спеціально створеним для цього алгоритмом.

Також крім опрацювання користувач може пройти тестування: у вигляді тесту, або ж у вигляді написання відповідей з клавіатури.

Користувач може створити колекцію, дати їй назву і добавити туди свої колоди карт, таким чином структурувавши матеріал.

Також у користувачів є можливість шукати колоду карт за потрібним іменем чи тегом, переглядати її і додати до списку своїх колекцій.

Користувач реєструється в системі вводячи ім’я, пошту, пароль.

Інтерфейс додатку має бути інтуїтивно зрозумілим, а дизайн лаконічним. Кнопки і основні елементи, повинні бути виділеними і гарно поєднуватися з фоном. Необхідно обрати кольорову гаму в межах п’яти кольорів, щоб процес навчання був приємний і не було відволікаючих факторів.

3.2. Опис розробки додатку

Проаналізувавши технічне завдання, було перейдено до розробки додатку, яка почалася з планування архітектури, схему якої зображено на схемі. (Рис. 3.2.1)

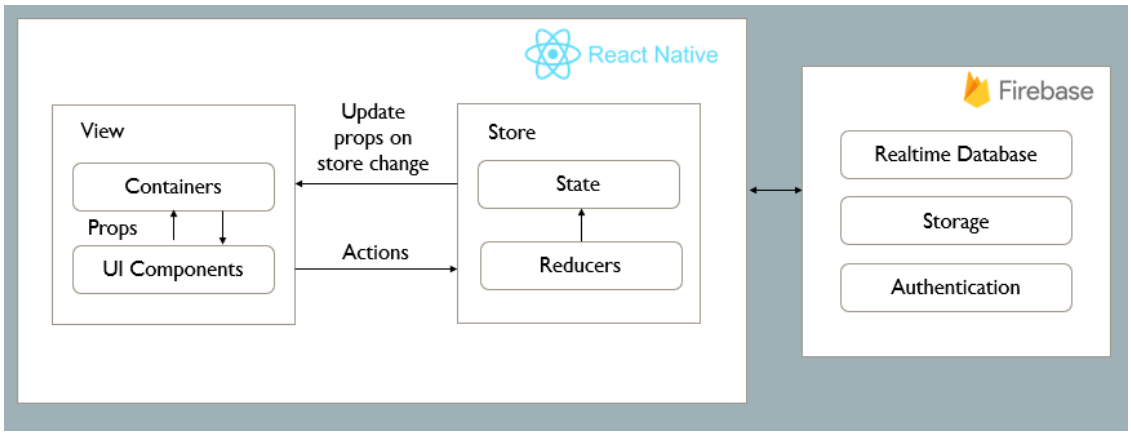


Рис. 3.2.1 – архітектура додатку.

Опісля розпочато написання коду. Розглянемо структуру проекту.

Програма починає роботу з файлу App.js, який містить NavigationContainer із стеком більшості екранів.

У файлі BottomTabNavigation.js додається BottomTabNavigator до основних екранів: головного (з колоди і колекціями користувача), створення нової колоди, пошуку, профайлу користувача.

firebase.js – файл для з налаштуваннями для взаємодії із Firebase.

У папці screens містяться наступні файли:

- AboutUsScreen.js – містить екран, з інформацією та короткою інструкцією додатку.
- DeckInSearchScreen.js – містить екран, який містить колоду іншого користувача при її перегляді у пошуку.

- EditCardInMyScreen.js – містить екран редагування флеш-картки у вже існуючій колоді.
- EditCardWhenAddScreen.js - містить екран редагування флеш-картки при додаванні нової колоди.
- EditCollectionScreen.js – містить екран для редагування колекції.
- EditDeckScreen.js - містить екран для редагування колоди.
- EditUserScreen.js - містить екран для редагування користувача.
- HomeScreen.js – містить екран з усіма колодами і колекціями користувача.
- LoginScreen.js – містить екран для входу в систему.
- MyCollectionScreen.js – містить екран для перегляду колекції.
- MyDeckScreen.js – містить екран для перегляду колоди.
- MyProfileScreen.js – містить екран для перегляду профілю і виходу із системи.
- NewCardScreen.js - містить екран для створення нової картки при створенні нової колоди.
- NewCardToExistingDeckScreen.js - містить екран для створення нової картки для вже створеної колоди.
- NewCollectionScreen.js - містить екран для створення нової колекції.
- NewDeckScreen.js - містить екран для створення нової колоди.
- PracticeCardScreen.js – містить екран для опрацювання флеш-карток із колекції.
- RegisterScreen.js – містить екран реєстрації.
- SearchScreen.js – містить екран пошуку колод.
- TestingByWritingCardsScreen.js – містить екран для тестування шляхом вводу відповіді з клавіатури.
- TestingCardsScreen.js – містить екран для тестування шляхом тесту.

У папці components містяться наступні файли:

- Card.js – містить компоненту флеш-картки, яка міститься у колоді при пошуку.
- CardAdd.js – містить компоненту флеш-картки, яка міститься у дошці при додаванні нової колоди.
- CardInDeck.js – містить компоненту флеш-картки, яка міститься у колоді при перегляді.
- CollectionLine.js – містить компоненту, яка використовується для відображення короткої інформації про колекцію при перегляді своїх колекції.
- Deck.js – містить компоненту, яка використовується для відображення короткої інформації про колоду при перегляді своїх колод.
- DeckLine.js - містить компоненту, яка використовується для відображення короткої інформації про колоду при пошуку.
- HeaderForMyProfile.js - містить компоненту з інформацією профілю.
- SearchField.js – містить компоненту для пошуку колод.

Також у проекті є папка node_modules, яка містять всі стандартні бібліотеки, які є необхідними для роботи проекту.

Файл package. json, який знаходиться у корені проекть, містить метадані, що стосуються проекту, і використовується для управління залежностями (dependencies) проекту, версією.

Авторизація здійснюється за допомогою сервісу Firebase Authentication.

Він надає серверні сервіси, прості у використанні SDK та готові бібліотеки інтерфейсу для автентифікації користувачів у додатку. Він підтримує автентифікацію за допомогою паролів, телефонних номерів, популярних постачальників об'єднаних ідентифікаційних даних, таких як Google.

Щоб увійти до додатку, спочатку користувач вводить облікові дані для автентифікації. Цими обліковими даними є адреса електронної пошти та пароль користувача. Потім ці облікові дані передаються в SDK для автентифікації Firebase. Потім серверні служби Firebase перевірять ці облікові дані та повертають

клієнту відповідь. Після успішного входу користувач може отримати доступ до основної інформації профілю користувача та контролювати доступ користувача до даних, що зберігаються в БД Firebase. [20]

Також додана можливість входу в систему за допомогою акаунту Google.

Використання Firebase Authentication дуже зручне, адже не потрібно турбуватися про те, яким чином, буде зберігатися інформація про користувачів (хешування паролів, тощо).

Також Firebase пропонує сервіс Firebase Storage, який дозволяє завантажувати та ділитися даними, такими як зображення та відео, що дозволяє вбудовувати мультимедійний вміст у програми. Його я використала для збереження зображень на флеш-картках.

За допомогою ImagePicker обрієтться необхідне зображення із галереї. А потім воно завантажується в Firebase Storage за допомогою наступного методу:

```
const uploadImage = async (ID,uri) => {
    const response = await fetch(uri);
    const blob = await response.blob();
    let ref = firebase.storage().ref().child(ID);
    return ref.put(blob);
}
```

Давайте розглянемо структуру БД.

В колекції “decks” зберігаються всі колоди флеш-карт. Кожна з яких містить наступну інформацію:

- deck_id: ID колоди;
- name: назва;
- tags: набір тегів;
- added: набуває значення true, якщо колода додана від іншого користувача, і false, якщо створена користувачем;
- cards: масив флеш-карток, кожна з яких містить наступну інформацію:
    - front: текст передньої сторони картки;
    - frontImage: ID зображення передньої сторони картки;

- back: текст задньої сторони картки;
- backImage: ID зображення задньої сторони картки;
- lastSeen: дата, коли востаннє було переглянуто картку:
- box: номер категорії картки. Набуває значень від 1 до 4 в залежності від того, як добре вона була засвоєна.
- learned: набуває значення true, якщо користувач відмітив картку, як вивчено останнього разу, і false, як не вивчено. Якщо користувач не вивчав ще картку, то значення буде null;

- user_id: ID користувача, чия це колода;
- user_id_creator: ID користувача, який її створив;
- visible: набуває значення true, якщо колода доступна для перегляду іншим користувача, і false, якщо недоступна;
- score: оцінка (0-100) наскільки користувач опрацював колоду, якщо користувач не опрацьовував колоду раніше, то значення буде null.

В колекції “collections” зберігаються всі колекції із колодами. Кожна з яких містить наступну інформацію:

- collection_id: ID колекції ;
- name: назва ;
- user_id: ID користувача чия це колекція ;
- decks: масив колод, які входять до колекції. Кожен з елементів масиву містить наступну інформацію:
  - deck_id: ID колоди ;
  - name: назва ;
  - length: кількість флеш-карток у колоді.

Взаємодія із БД здійснюється завдяки сервісам включених у Firebase.

Продемонструю приклади декількох запитів:

Отримання інформації про колоди карт користувача:

```
db.collection("decks").where("user_id", "==", currentUser)
    .get()
```

```
    .then((querySnapshot) => {
        querySnapshot.forEach((doc) => {
            results.push(doc.data())
        });
        setDecks(results);
    })
    .catch((error) => {
        console.log("Error getting documents: ", error);
    });
```

Створення нової колоди карт:

```
db.collection("decks").doc(ID).set({
    name: deckName.trim(),
    user_id: auth.currentUser.uid,
    user_id_creator: auth.currentUser.uid,
    visible: isSelected,
    cards: cardsArr,
    added: false,
    score: null,
    deck_id: ID,
    tags: tagsArr
})
```

Видалення колекції:

```
let query = db.collection('collections')
    .where('collection_id', '==', collection.collection_id);
query.get().then(function (querySnapshot) {
    querySnapshot.forEach(function (doc) {
        doc.ref.delete();
});
});
```

Оновлення колоди:

```
const docRef = db.collection('decks').doc(deck.deck_id);
const update = docRef.update({
    name: deckName,
    visible: isSelected,

});
```

Також розроблено алгоритм для вибору флеш-карток, які користувач знає найменше.

Для цього для кожної картки ми визначаємо одну категорію з чотирьох:

- Перша – картки такої категорії потрібно повторювати кожного дня.
- Друга - картки потрібно повторювати кожних три дні.
- Третя – картки необхідно повторювати щотижня.

- Четверта – картки необхідно повторювати раз на два тижні.

При додаванні нової флеш-картки, вона попадає в першу категорію. Якщо під час практики чи тестування картка була визначена як вивчена, то вона потрапляє до другої категорії і так далі. А при визначенні картки як невивченої, вона повертається до попередньої категорії, меншої за значенням.

До набору карток для практикування потрапляють всі із першої категорії, із другої ті, яких користувач не повторював на протязі 3 днів, із третьої ті, яких користувач не повторював на протязі 7 днів, із четвертої ті, яких користувач не повторював на протязі 14 днів.

Потім потрібно відфільтрувати кількість карток в наборі, щоб вона не була замалою чи завеликою, оскільки користувач задає бажану кількість карток для практики. Якщо набір завеликий, то видаляємо, якщо замалий то додаємо випадково картки доти, доки кількість нас не задовольнятиме.

3.3. Тестування програми і результати її виконання

Рис. 3.3.1 – екран для входу.

При запуску додатку користувач потрапляє на екран для входу в систему. (Рис.3.3.1) Натиснувши на кнопку “Register” переходить на екран для реєстрації (детальніше Додаток А). Користувач заповнює всі поля необхідні при реєстрації, після чого він потрапляє на головний екран. (Рис.3.3.2).

Вже зареєстрований користувач може увійти в систему, якщо на екрані для входу (Рис. 3.3.1) введе необхідні дані і натисне кнопку “Login”. Також можна увійти в систему за допомогою акаунта Google, натиснувши на кнопку нижче.

Опинившись на головному екрані (Рис.3.3.2) користувач може переглянути свої колоди карт. Натиснувши на одну з них, він

Рис. 3.3.2 – головний екран.

потрапляє на екран вибраної колоди (Рис. 3.3.3).

Натиснувши на значок корзинки, висвітлюється повідомлення, для того, щоб підтвердити видалення. Також натиснувши на значок олівця або “+”, користувач може відповідно редагувати або додавати нову флеш-карту до колекції.

У верхньому лівому кутку відображається оцінка освоєння дошки. Збоку наявні три кнопки для перегляду: всіх карток, невивчених, вивчених.

Також натиснувши на кнопку “Not quite” користувач бачить неосвоєні флеш-картки і має можливість опрацьовувати тільки їх.

Натиснувши на кнопку для “Practice all” користувач потрапляє на екран для практикування всіх карт колоди (Рис. 3.3.4). Натиснувши кнопку “Flip” картка обертається і тоді необхідно натиснути на “Not quit” , якщо інформація не засвоєна, або на “Learned”, якщо засвоєна. Натиснувши на кнопку для “Practice” користувач буде опрацьовувати необхідні картки, які підбираються за допомогою алгоритму.

Натиснувши на кнопку “Quiz” користувач може пройти тестування (Рис. 3.3.5). Йому потрібно вибрати відповідь, після чого, якщо вона неправильна - буде висвітлено коректний варіант.

Рис. 3.3.3 – екран колоди.

Рис. 3.3.4 – флеш-картка при опрацюванні.

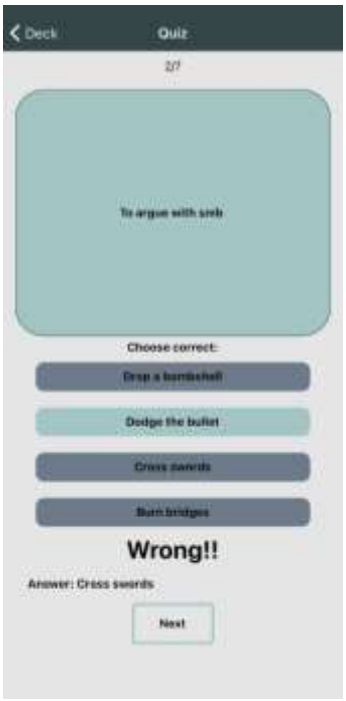


Рис. 3.3.5 – тестування.

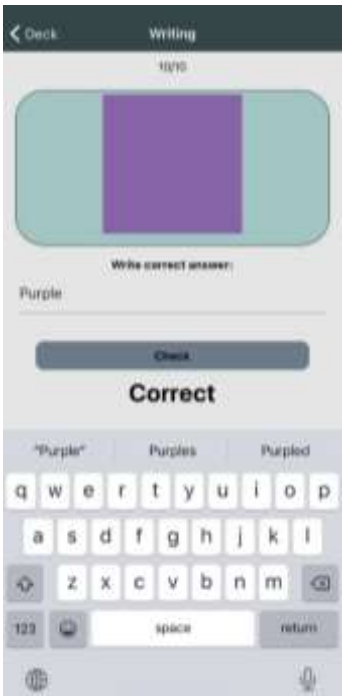


Рис. 3.3.6– тестування.

Натиснувши на кнопку “Write” користувач може пройти ще один вид тестування (Рис 3.3.6). Йому потрібно ввести відповідь із клавіатури, після чого, якщо вона неправильна - буде висвітлено коректний варіант.

Також на головному екрані користувач може переглядати свої колекції.

Натиснувши на обрану потрапляє на екран колоди (Рис. 3.3.3).

Щоб створити нову колекцію необхідно натиснути + у верхньому правому кутку. Тоді відкриється екран створення нової колекції (Рис. 3.3.7).

Щоб створити нову колоду необхідно натиснути на значок із знаком плюс внизу екрана. Після чого користувач опиниться на екрані створення нової колоди.

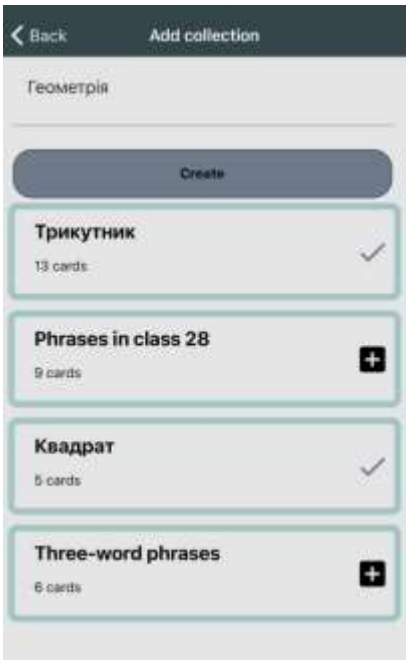


Рис. 3.3.7 – створення нової колекції.

Для того, щоб додати нову картку необхідно натиснути “Add” (Рис. 3.3.8) Щоб відмінити створення – “Reset”. Щоб створити колоду – “Create”.

Для того, щоб знайти колоду за зацікавленою тематикою, необхідно перейти в розділ пошуку внизу екрану. І ввести дані (Рис. 3.3.9). Можна переглянути колоду натиснувши на неї, а також додати собі натиснувши кнопку “Add”.

Для перегляду свого профілю потрібно натиснути кнопку внизу екрану. (Рис. 3.3.10) . При натисканні кнопки “About us” висвітлиться невелика інструкція і інформація про додаток.

Рис. 3.3.8 – створення нової картки.

Рис. 3.3.9– пошук колоди.

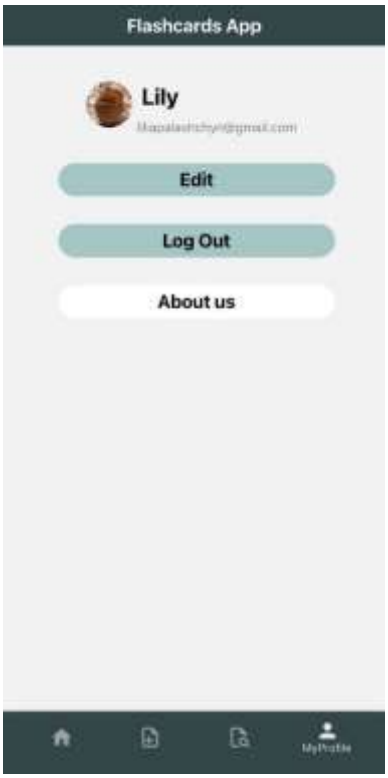


Рис. 3.3.10 – профіль користувача.

3.4. Висновки до розділу

Детально проаналізовано технічне завдання. Продемонстровано архітектуру додатку. Описано структуру бази даних. Розглянуто структуру проекту. Обґрунтовано алгоритм вибору карток для навчання. Продемонстровано результати виконання програми.

ВИСНОВКИ

Отже, у рамках курсової роботи було проаналізовано предметну область та розроблено мобільний додаток для навчання.

Розглянуто сервіси-аналоги та визначено закономірності, що дозволило визначити ключові функції, які необхідно мати додатку такого типу для повноцінної роботи.

Проаналізовано найбільш популярні та вживані технології, що використовуються для розробки мобільних застосунків. Обрано ті, що надають можливість найкраще реалізувати функціонал обраного проекту.

Додаток розроблено за допомогою фреймворку React Native з використанням бази даних Firebase Realtime Database у середовищі розробки WebStorm. Сервіс задовольняє потреби сучасного користувача та дозволяє ефективно опановувати нову інформацію. Розроблено лаконічний дизайн та інтуїтивно зрозумілий інтерфейс.

Створено додаток, який надає можливість структурувати, вивчати новий матеріал, здійснювати тестування по ньому, а також обмінюватися ним з іншими користувачами.

Надалі можна розширити функціонал застосунку додавши можливість прикріпляти аудіо файли до флеш-карток, роздруковувати колоди карт, створити додаткові види тестування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вікіпедія. Інформація про StudyBlue. [Електронний ресурс].
Режим доступу: https://en.wikipedia.org/wiki/StudyBlue

2. Офіційний сайт Chegg-Prep. Інформація про сервіс. [Електронний ресурс].
Режим доступу: https://www.chegg.com/contactus/Chegg-Prep/how-to-use-Chegg-Prep/1459724801/What-is-Chegg-Prep.htm

3. Вікіпедія. Інформація про Quizlet. [Електронний ресурс].
Режим доступу: https://en.wikipedia.org/wiki/Quizlet

4. Офіційний сайт AnkiApp. [Електронний ресурс].
Режим доступу: https://www.ankiapp.com

5. Стаття: 7 Latest Mobile App Development Technologies for Startups. Автор: Jyoti Gupta .[Електронний ресурс].
Режим доступу: https://customerthink.com/7-latest-mobile-app-development-technologies-for-startups/

6. Стаття:Choosing the best programming language for mobile app development.
Автор: Ajay Chebbi. [Електронний ресурс].
Режим доступу: https://developer.ibm.com/technologies/mobile/articles/ choosing-the-best-programming-language-for-mobile-app-development/

7. Стаття: Pros and Cons of the Kotlin Programming Language for Developing Android Apps. Автор: Gabriel Samojło. [Електронний ресурс].
Режим доступу: https://www.netguru.com/blog/kotlin-pros-and-cons

8. Стаття: The pros and cons of programming in Swift. Автор: Vasilisa Sheromova. [Електронний ресурс].
Режим доступу:

https://www.google.com/search?q=pros+and+cons+swift&sxsrf=ALeKk03QkU9S3mKqVvtffvPNEfAFYBLlPw%3A1619890187563&ei=C5CNYPPcIeWGrwSU8LrgBg&oq=pros+and+cons+swift&gs_lcp=Cgdnd3Mtd2l6EAMyBQgAEMsBMgUIABDLATIFCAAQywEyBQgAEMsBMgUIABDLATIFCAAQywEyBQgAEMsBMgYIABAWEB4yBggAEBYQHjIGCAAQFhAeOgcIABBHELADOgIIAFD7xSVYjuIlYIzlJWg1CcAJ4AIABrgaIAaQOkgEJMC4yLjUtMS4xmAEAoAEBqgEHZ3dzLXdpesgBCMABAQ&s

client=gws-wiz&ved=0ahUKEwizjtGZganwAhVlw4sKHRS4DmwQ4dUDCA8&uact=5

9. Стаття: REACT NATIVE VS XAMARIN: PROS AND CONS. Автор: Norbert Kamienski. [Електронний ресурс].
Режим доступу: https://pagepro.co/blog/react-native-vs-xamarin-pros-and- cons/

10. Офіційний сайт GitHub. [Електронний ресурс].
Режим доступу: https://github.com/.

11. Стаття: Pros and Cons of JavaScript – Weigh them and Choose wisely! [Електронний ресурс].
Режим доступу: https://data-flair.training/blogs/advantages-disadvantages-javascript/

12. Стаття: Should You Use TypeScript with React Native? Автор: Spencer Carli. [Електронний ресурс].
Режим доступу: https://www.reactnativeschool.com/should-you-use-typescript-with-react-native

13. Офіційний сайт Expo.[Електронний ресурс].
Режим доступу: https://docs.expo.io/

14. Стаття: Top 5 Databases For React Native App Development. Автор: Nasrullah Patel [Електронний ресурс].
Режим доступу: https://www.business2community.com/tech-gadgets/top-5- databases-for-react-native-app-development-02383205

15. Стаття: The Pros and Cons of DynamoDB. Автор: Matthew Bonig [Електронний ресурс].
Режим доступу: https://matthewbonig.com/2019/06/28/dynamodb-pros-cons/

16. Стаття: Advantages of MongoDB | Disadvantages of MongoDB [Електронний ресурс].
Режим доступу: https://data-flair.training/blogs/advantages-of-mongodb/

17. Стаття: Pros and Cons of Firebase. Автор: Julianna Sykutera [Електронний ресурс].
Режим доступу: https://redvike.com/pros-and-cons-of-firebase/

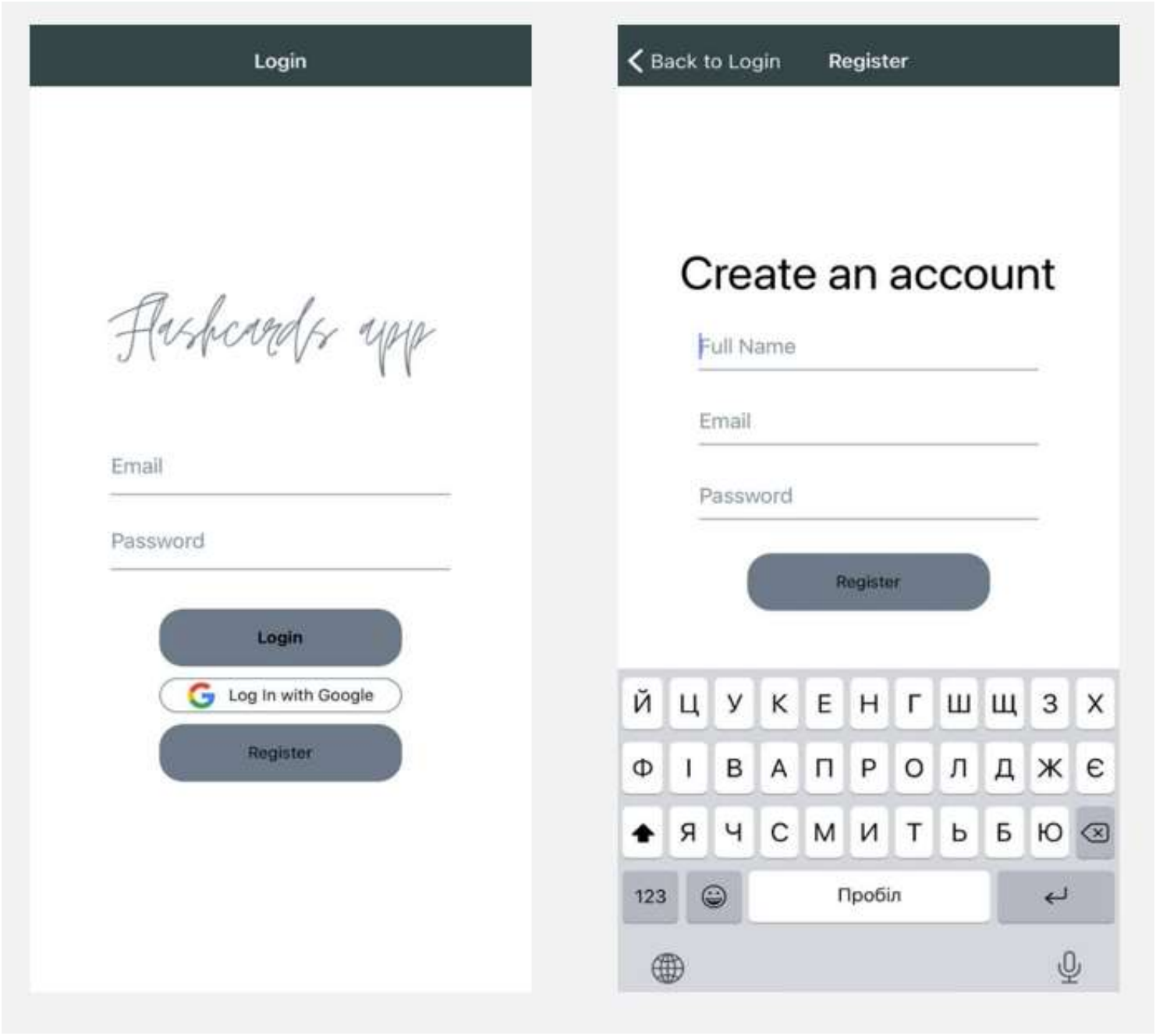
18. Офіційний сайт Firebase [Електронний ресурс].
Режим доступу: https://firebase.google.com/docs/auth

ДОДАТКИ

Додаток А

(Обов’язковий)

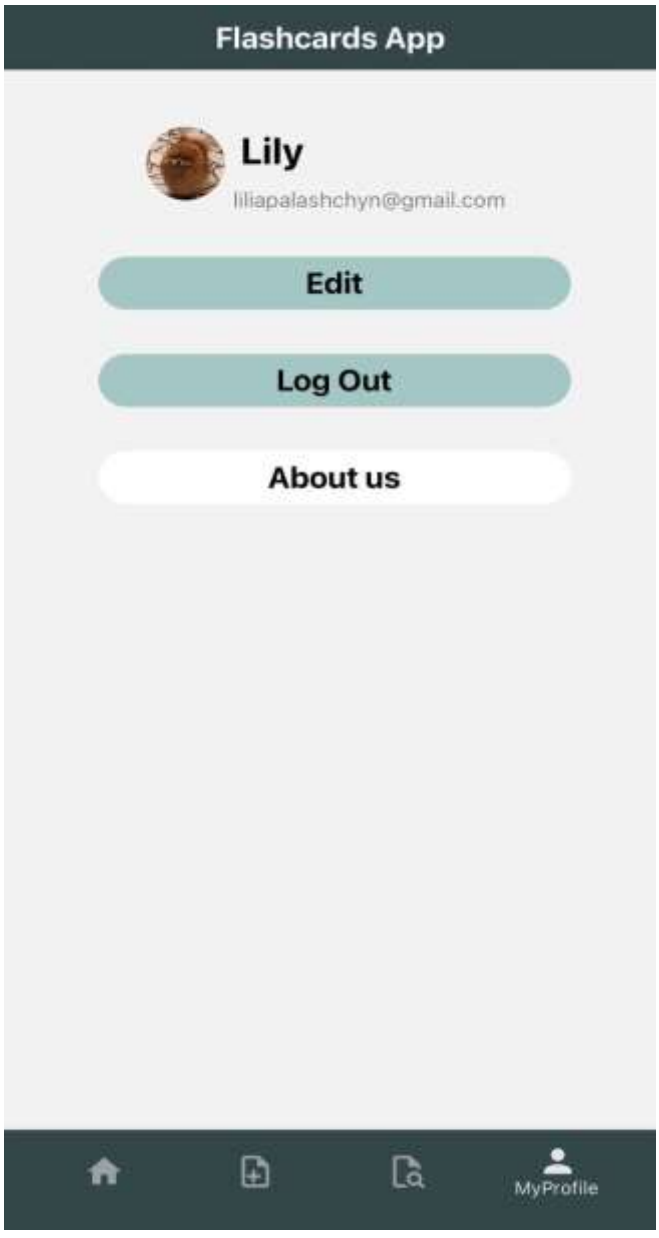
Вхід до системи

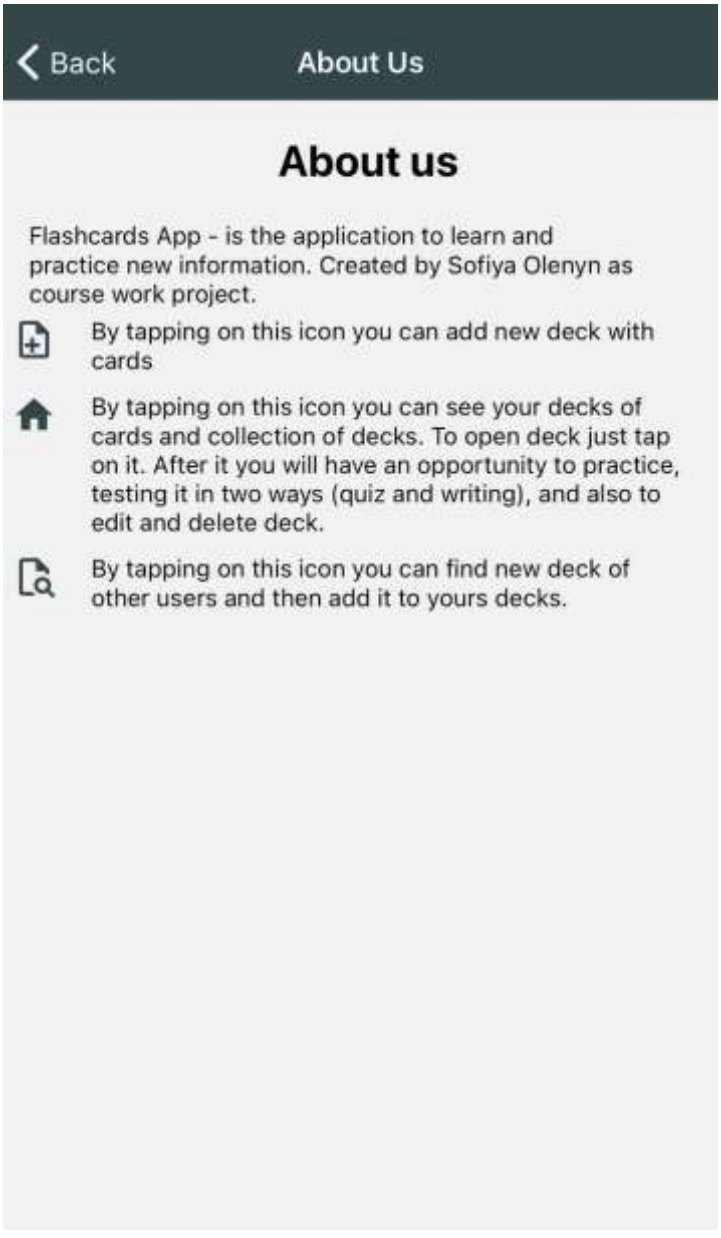


Додаток Б

(Обов’язковий)

Екран користувача та інформація про застосунок

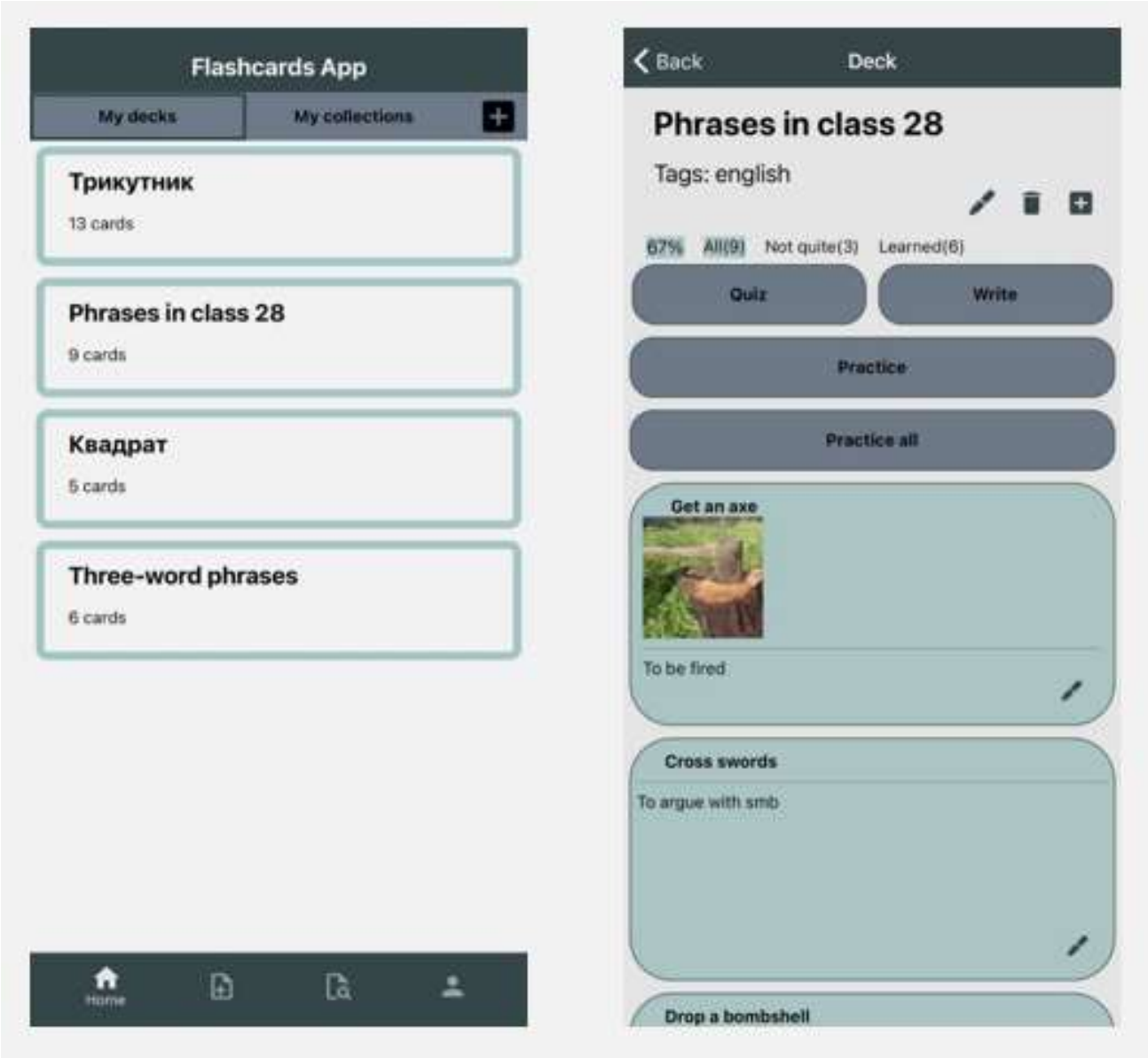




Додаток В

(Обов’язковий)

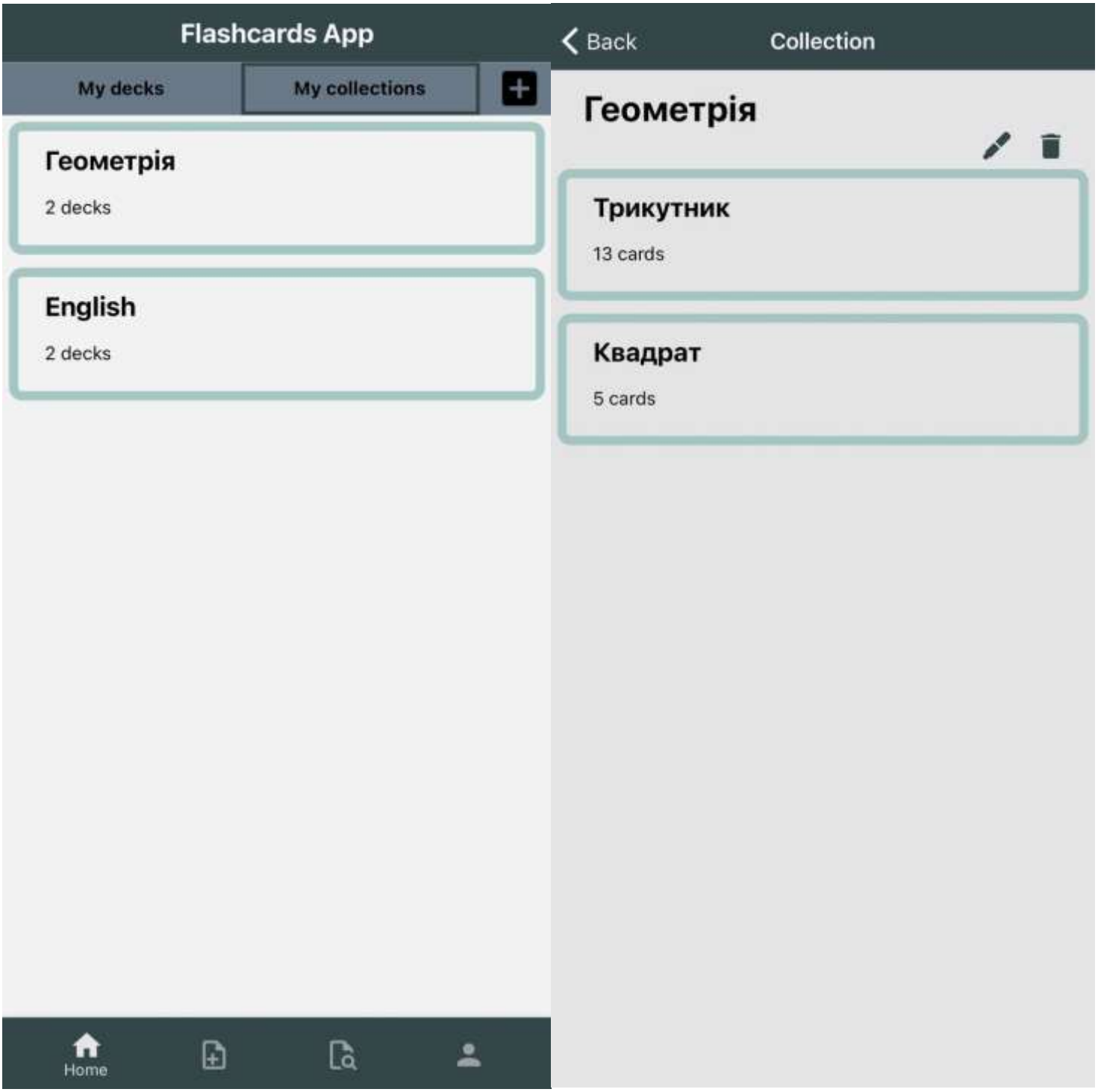
Відображення колод карт



Додаток Г

(Обов’язковий)

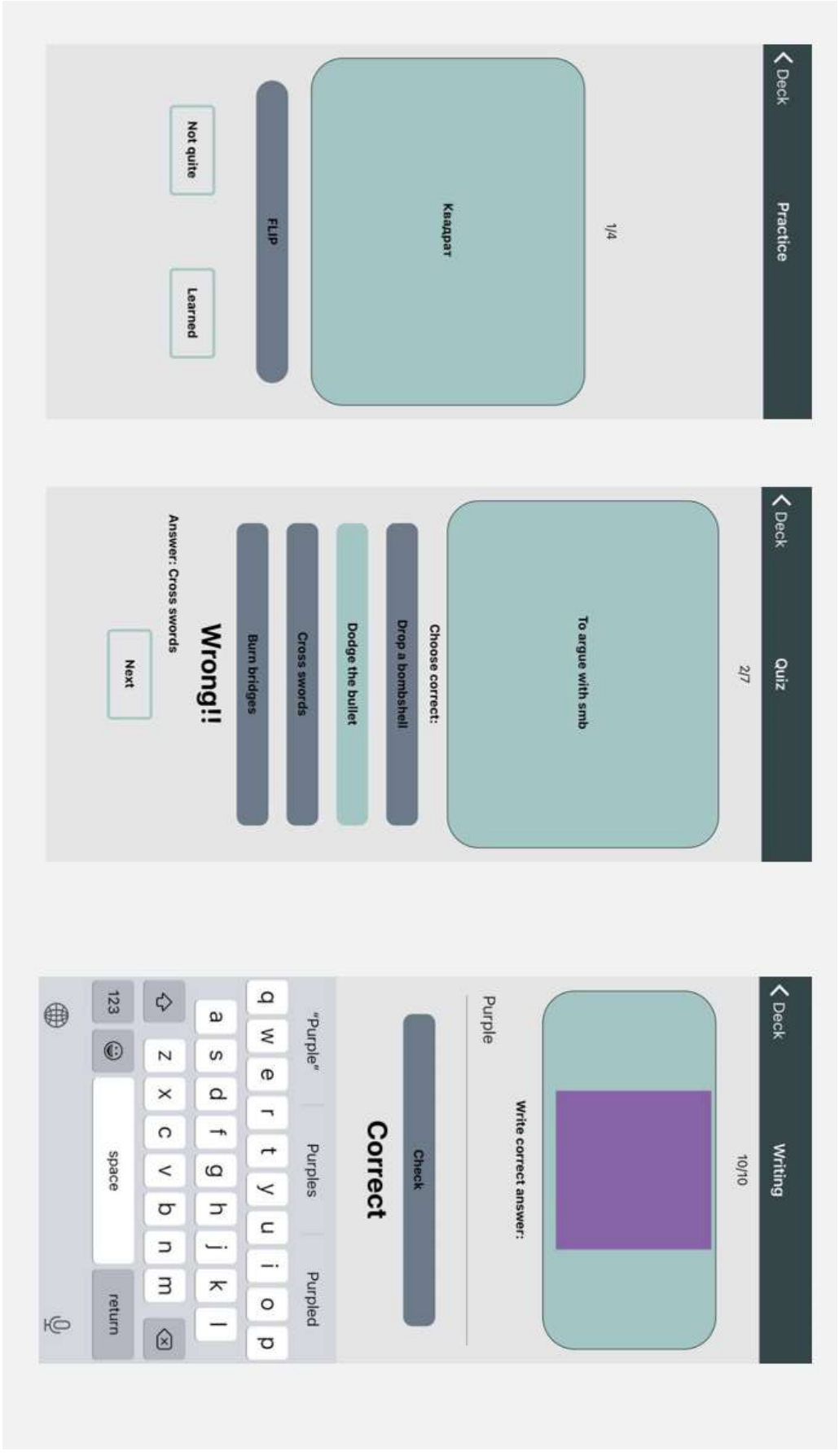
Відображення колекцій



Додаток Д

(Обов’язковий)

Практикування і тестування карток



Додаток Е

(Обов’язковий)

Пошук та сторінка знайденої колоди.

Додаток Ж

(Обов’язковий)

Створення нових колод, карток і колекцій.