Міністерство освіти і науки України
Сумський державний університет
Навчально-науковий інститут бізнесу, економіки та менеджменту
Кафедра економічної кібернетики

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Інтелектуалізація каналів оптимізації
освітньої діяльності в Україні»

Виконав студент 4 курсу (номер курсу), групи ЕК-71а (шифр групи)
Спеціальності 051 «Економіка»
(«Економічна кібернетика»)
Ревенко А. В.
(прізвище та ініціали студента)
Керівник: професор, д.е.н., Кузьменко О. В.
(посада, науковий ступінь, прізвище та ініціали керівника)

Суми – 2021 рік

РЕФЕРАТ

кваліфікаційної роботи бакалавра на тему

«Інтелектуалізація каналів оптимізації освітньої діяльності в Україні»

студента Ревенка Артема Володимировича

(прізвище, ім'я, по батькові студента)

Актуальність теми, обраної для дослідження, визначається тим, що у сфері освітньої діяльності України постійно проводиться взаємодія з закордонними колегами для поширення та покращення міжнародних зв'язків шляхом проведення наукових заходів англійською мовою. В Україні все більше осіб, котрі безпосередню беруть участь в освітній діяльності володіють англійською мовою на низькому рівні. Все більше користувачів гаджетами надають перевагу саме мобільним додаткам замість комп'ютерних програм. Можливість об'єднання процесу користування девайсами з рівнем володіння англійською мовою.

Мета кваліфікаційної роботи полягає у розробці прототипу мобільного додатку для вивчення англійської мови.

Об'єктом дослідження є відносини що виникають між студентами та викладачами в процесі надання освітніх послуг в Україні.

Предметом дослідження є інструменти інтелектуалізації та оптимізації освітньої діяльності.

Задачами дослідження є:

– охарактеризувати освітню діяльність України;

– провести аналіз сучасного стану інтелектуалізації освітньої діяльності;

– сформувати вимоги до системи та зробити вибір технології розробки;

– змоделювати та спроектувати архітектуру мобільного додатку;

– реалізувати інформаційне забезпечення прикладної програми

– розробити контрольний приклад та інструкцію щодо використання мобільного додатку.

Для досягнення поставленої мети та задач дослідження були використані такі методи дослідження: дедукція, аналіз, синтез, порівняння, системний аналіз, проектування та моделювання.

Інформаційною базою кваліфікаційної роботи є матеріали ПП «БАС», аналітичні огляди та наукові публікації зарубіжних авторів, присвячених вивченню англійської мови

Основний науковий результат кваліфікаційної роботи полягає у такому: проаналізовано ринок освітніх мобільних додатків, побудовано власний мобільний додаток, який дозволяє самому налаштовувати асоціативні зв'язки між словом та картинкою, адаптувати словник під себе.

Одержані результати можуть бути використані фірмами для навчання своїх працівників (підвищення кваліфікації), студентами, школярами – всіма, хто хоче вивчити англійські мову.

Апробація кваліфікаційної бакалаврської роботи здійснена на ПП «БАС» (довідка № 23 від 19 травня 2021 року).

Ключові слова: інтелектуалізація, автоматизація, мобільний додаток, прикладна програма, освітня діяльність, прототип, англійська мова.

Зміст кваліфікаційної роботи викладено на 48 сторінках. Список використаних джерел із 42 найменувань, розміщений на 5 сторінках. Робота містить 3 таблиці, 47 рисунків, а також 6 додатків, розміщених на 33 сторінках.

Рік виконання кваліфікаційної роботи – 2021 рік.

Рік захисту роботи – 2021 рік.

Міністерство освіти і науки України
Сумський державний університет
Навчально-науковий інститут бізнесу, економіки та менеджменту
Кафедра економічної кібернетики

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.е.н., професор
_____ О.В. Кузьменко
“__” _____ 2021 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА
спеціальність 051 «Економіка» (Економічна кібернетика)
студента 4 курсу, групи ЕК-71а

Ревенко Артем Володимирович
(прізвище, ім’я, по батькові студента)

1. Тема роботи: Інтелектуалізація каналів оптимізації освітньої діяльності в Україні
затверджена наказом по університету від «__»_____ 2021 року №_____
2. Термін подання студентом закінченої роботи «__» _____ 2021 року
3. Мета кваліфікаційної роботи: розробка прототипу мобільного додатку для вивчення англійської мови.
4. Об’єкт дослідження: відносини що виникають між студентами та викладачами в процесі надання освітніх послуг в Україні.
5. Предмет дослідження: інструменти інтелектуалізації та оптимізації освітньої діяльності.
6. Кваліфікаційна робота виконується на матеріалах: Кваліфікаційна робота виконується на матеріалах ПП "БАС", аналітичних оглядів та наукових публікацій вітчизняних та зарубіжних авторів, присвячених вивченню англійської мови
7. Орієнтовний план кваліфікаційної роботи, терміни подання розділів керівникові та зміст завдань для виконання поставленої мети
Розділ 1. АНАЛІЗ СУЧАСНОГО СТАНУ ІНТЕЛЕКТУАЛІЗАЦІЇ ОСВІТНЬОЇ ДІЯЛЬНОСТІ В УКРАЇНІ – 11 травня 2021 року
(назва – термін подання)

У розділі 1. Характеристика освітньої діяльності, аналіз сучасного стану інтелектуалізації освітньої діяльності, формування вимог до системи та вибір технології розробки
(зміст конкретних завдань до розділу, які повинен виконати студент)

Розділ 2. РЕАЛІЗАЦІЯ ПРОТОТИПУ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ – 17 травня 2021 року
(назва – термін подання)

У розділі 2. Проектування архітектури мобільного додатку, реалізація інформаційного забезпечення мобільного додатку, контрольний приклад та інструкція щодо використання, очікуваний ефект від використання мобільного додатку

(зміст конкретних завдань до розділу, які повинен виконати студент)

8. Консультації з роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1			
2			

9. Дата видачі завдання: «01» березня 2021 року

Керівник кваліфікаційної роботи _____ Кузьменко О. В.
(підпис) (ініціали, прізвище)

Завдання до виконання одержав _____ Ревенко А. В.
(підпис) (ініціали, прізвище)

ЗМІСТ

ВСТУП

Кожен учасник освітньої діяльності має намір покращувати власні навички для отримання вищого навчального ступеня, яка підтвердить рівень майстерності. До подібних навичок так само і відноситься рівень володіння іноземними мовами, а саме – англійською мовою. Англійська мова – друга за популярністю, після китайської, але перша за кількістю способів застосування, міжнародна мова. Вона має величезну кількість застосувань. І чим краще людина нею володіє, тим вищий статус вона займає. Таким чином, кожен учасник освітньої діяльності має намір постійно покращувати рівень володіння поданою мовою і використовувати її як у робочих процесах, так і у повсякденному житті.

Нині існують велика кількість методів вивчення англійської мови, починаючи від занять з репетитором по Скайпу, закінчуючи інтерактивним мобільним додатком. Розвиток технологій не стоїть на місті, тому можливостей покращення рівня володіння поданою мовою все більше й більше.

У 2021 році змінився ритм життя. Тепер кожна людина має смартфон, за допомогою якого для нього відкриваються двері до всього, чого він хоче. І виникає питання: «Так чому ж не скористатися цим захопленням багатьох людей і не створити новий продукт, який допоможе їм опанувати англійську мову?»

Застосовувати англійську мову можна як для створення і поліпшення зв'язків з іноземними партнерами або колегами, для саморозвитку, так і для подорожей, за допомогою яких відбувається духовне збагачення кожної людини.

Отже відбувається процес інтелектуалізації – неперервного та нескінченного динамічного процесу, який супроводжує діяльність соціально-економічної системи будь-якого рівня в усіх сферах її активності, забезпечує

виникнення та розвиток інтелекту, формує людський ресурс, здатний до творчих аналізу та синтезу інформації, оцінювання й організації виробничих процесів, генерування і реалізації власних та залучених ідей [3].

Для постійного процесу самовдосконалення необхідні певні інструменти, котрі будуть допомагати постійно розвиватися та вчитися.

І так, як ми живемо у світі швидких інновацій та розвитку технологій, існує можливість створення власного інструменту для досягнення поданої мети – постійного розвитку володінням англійською мовою.

Подана можливість допоможе об’єднати повсякденний та рутинний процес користування девайсами з покращенням рівня англійської мови та пришвидшити процес оптимізації освітньої діяльності в Україні.

Таким чином, об’єктом кваліфікаційної роботи виступають відносини що виникають між студентами та викладачами в процесі надання освітніх послуг в Україні.

Предметом кваліфікаційної роботи є інструменти інтелектуалізації та оптимізації освітньої діяльності.

Мета: розробка прототипу мобільного додатку для вивчення англійської мови.

Завдання кваліфікаційної роботи:

– охарактеризувати освітню діяльність України;

– провести аналіз сучасного стану інтелектуалізації освітньої діяльності;

– сформувати вимоги до системи;

– змоделювати та спроектувати архітектуру мобільного додатку;

– реалізувати інформаційне забезпечення прикладної програми;

– розробити контрольний приклад та інструкцію щодо використання мобільного додатку

– охарактеризувати очікуваний ефект від використання мобільного додатку.

РОЗДІЛ 1. АНАЛІЗ СУЧАСНОГО СТАНУ ІНТЕЛЕКТУАЛІЗАЦІЇ ОСВІТНЬОЇ ДІЯЛЬНОСТІ В УКРАЇНІ

1.1. Характеристика освітньої діяльності

За терміном Закону України про освіту, «освітня діяльність – це діяльність суб'єкта освітньої діяльності, спрямована на організацію, забезпечення та реалізацію освітнього процесу у формальній та/або неформальній освіті» [6].

У свою чергу, «суб'єкт освітньої діяльності – це фізична або юридична особа (заклад освіти, підприємство, установа, організація), що проводить освітню діяльність» [7].

Основною метою освітньої діяльності є набуття та задоволення здобувачами вищої освіти та інших осіб актуальними освітніми потребами, серед котрих є володіння англійською мовою на рівні вище середнього.

За Загальноєвропейськими рекомендаціями з мовної освіти або кваліфікаційні документи (диплом про вищу освіту, науковий ступінь), пов’язані з використанням зазначеної мови одним із членів проектної групи під час провадження підготовки іноземців та осіб без громадянства, вимагають володіти англійською мовою на рівні B2 [5].

Рівні володіння англійською мовою [14, 28]:

A1 – початкова англійська [2]. Особа має змогу розуміти та використовувати звичні повсякденні слова та фрази, необхідних для задоволення потреб, що прийняли специфічну форму відповідно до культурного рівня та індивідуальності людини. Може представити себе та дати коротку інформацію про себе.

A2 – елементарна англійська. Особа може розуміти речення та фрази, які часто вживаються у сфері безпосереднього значення або певної тематики: географія, сім’я, традиції, особиста інформація, сфера зайнятості і т.д. Має

можливість брати участь у простих дискусіях, котрі передбачають простий обмін інформацією. Може описати аспекти свого досвіду, найближчого оточення та питань, що стосуються безпосередніх потреб.

B1 – середня англійська. Особа такого рівня вважається незалежним користувачем англійською мовою. Може розуміти основні моменти стандартного ведення питань, які регулярно використовуються у побуті. Має змогу вирішувати проблеми, які можуть виникнути під час подорожі закордоном (у країні, мешканці якого вільно спілкуються англійською). Уміє писати тексти на популярні теми та особисті інтереси.

B2 – англійська вище середнього [9]. Особа може розуміти основні ідеї складного тексту як на конкретні, так і на абстрактні теми, включаючи спеціалізовані дискусії. Може взаємодіяти з носіями англійської мови без напруги з будь-якої сторони. Уміє писати детальні тексти з широкого кола предметів та може пояснити особисту точку зору на певну проблему, користуючись її перевагами та недоліками.

C1 – англійська високого рівня. Особа поданого рівня мови є досвідченим користувачем іноземної мови. Може розуміти широкий спектр складних та спеціалізованих текстів та розпізнавати наявний зміст. Може вільно та без попередньої підготовки висловлювати особисту думку. Уміє ефективно користуватись мовою в професійних, соціальних та академічних напрямках. Може писати структурований, чіткий, детальний текст на складні теми, використовуючи зв’язні пристрої.

C2 – рівень володіння носія. Особа з легкістю розуміє та сприймає прочитану або усну інформацію. Може узагальнювати інформацію з різних усних та письмових джерел, відновлюючи аргументи у послідовній презентації. Уміє спонтанно висловити особисту думку, дуже плавно і точно, розуміти відтінки значень у складних ситуаціях.

Беручи до уваги дослідження компанією Education First [27], Україна посіла 49 місце зі 100 країн, котрі знаходяться у рейтингу володіння англійською мовою. Наша країна отримала індекс 52.13, тим самим потрапила

до групи низького володіння іноземною мовою. А за Загальноєвропейською школою мовної компетенції (CEFR) це відповідає рівню B1 [15].

CEFR (Common European Framework of Reference) – це спеціально розроблена для того, щоб можна було оцінити рівень володіння певним європейським мовою, віднісши навички певним критеріям [21].

Тому постає питання: для чого потрібно вивчати англійську мову?

Найпоширеніша відповідь на подане запитання: англійська – це «мова можливостей», тому що нею говорять майже в усіх країнах світу. Нині подана іноземна мова є офіційною мовою більш, ніж у 50-ти країнах світу, котрою володіють більш ніж 1.5 мільярда людей.

Основні переваги володіння англійською мовою [12]:

1) Англійська – друга найпоширеніша мова у світі.

2) Англійська мова є однією з найдоступніших мов для вивчення. Вона має просту систему дієслів та базовий алфавіт та вважається однією з найлегших для сприйняття.

3) Англійська є мовою навчання. Найбільша кількість книг написано саме поданою мовою. Це є вагомою перевагою для продовження навчання та читання улюблених книг в оригіналі.

4) Англійська – мова бізнесу. В умовах пандемії все більше людей починає працювати з роботодавцями з-за кордону, використовуючи саме універсальну мову спілкування – англійську.

5) Англійська є мовою подорожей. Вище було сказано, що поданою мовою володіє понад 1.5 млрд людей, що є одним з найважливіших ключових моментів подорожей. Англійська є вашим основним і єдиним інструментом у спілкуванні з людьми в іншій країні.

6) Англійська мова є мовою саморозвитку. Майже всі світові генії користуються поданою мовою для повідомлення найновітніших та найголовніших відкриттів сьогодення.

7) Англійська – ваш основний інструмент у програмуванні. Подана перевага є найважливішою у сфері інформаційних технологій. Нині кодинг є

основним джерелом заробітку та саморозвитку, головним інструментом якого є саме англійська мова.

Таким чином, англійська мова – це основний інструмент для вашого розвитку, бізнесу та відпочинку.

Виходячи з вищесказаної інформації, а саме з результату компанії Education First, при якому рівень володіння англійською мовою громадянами України відповідає рівню B1 та списку переваг поданої мови, мною було обрано тему дипломної роботи «Інтелектуалізація каналів оптимізації освітньої діяльності в Україні».

Виходячи зі статті Святослава та Галини Кісь «Явище інтелектуалізації у діяльності соціально економічних систем», інтелектуалізація – це неперервний, нескінченний, динамічний процес, який супроводжує діяльність соціально-економічної системи будь-якого рівня в усіх сферах її активності, забезпечує виникнення та розвиток інтелекту, функціонування індивідуального та суспільного інтелекту, покращує якісні характеристики організації зі зворотним ефектом, формує людський ресурс, здатний до творчих аналізу та синтезу інформації, оцінювання й організації виробничих процесів, генерування і реалізації власних та залучених ідей [3].

А термін оптимізація, згідно з тлумачного словника української мови [10, 34] – це надання чому-небудь оптимальних, найбільш сприятливих властивостей, співвідношень. Вибір найкращого (оптимального) варіанта з великої кількості можливих. Покращання характеристик системи. Визначення значень економічних показників, при яких досягається оптимум, тобто оптимальний, найкращий стан системи; найчастіше оптимуму відповідає досягнення найвищого результату при даних витратах ресурсів або досягнення заданого результату при мінімальних ресурсних витратах.

Таким чином, актуальність та сучасність кваліфікаційної роботи дає можливість неперервно та нескінченно супроводжувати діяльність суб’єктів освітньої діяльності, яка націлена на досягнення найкращого стану володіння

англійською мовою, сприяти покращенню розумових характеристик здобувачів освіти та покращенню пам’яті.

1.2. Аналіз сучасного стану інтелектуалізації освітньої діяльності

Провівши дослідження ринку мобільних додатків на рахунок існування альтернатив для вивчення англійської мови, мною було знайдено декілька варіантів: LinguaLeo, Duolingo та Easy Ten.

Перша альтернатива LinguaLeo представляє собою платформу, що працює від комп'ютера до смартфона, основним плюсом якого є ігровий метод вивчення іноземної мови. Основними недоліками програми виступає те, що найкорисніші та найефективніші методики навчання є платними, а так само те, що додаток для операційної системи Андроїд сильно поступається комп'ютерної версії. І так, як кількість користувачів цієї ОС перевищує кількість користувачів IOS, це розв’язує руки для завоювання даного сегменту [30].

Другим конкуруючим додатком є Duolingo. Дана альтернатива є умовно безкоштовною, що гарантує доступ до всіх матеріалів і ресурсів. Це його найбільша перевага. Іншими плюсами, на думку багатьох користувачів, є зрозумілий та інтуїтивний інтерфейс і можливість почати заняття з нульовим рівнем знань. Основними недоліками цього додатка виступають його переоцінена значущість, неможливість роботи без інтернету і просування мало використовуваної лексики. Спираючись на думку користувачів, можна зробити висновок про те, що хоч додаток і досить гарний для тренування лексики з англійської мови, але дійсна його користь явно переоцінена [25, 38].

Третьою альтернативою є додаток під назвою Easy Ten. Воно пропонує користувачам щодня вивчати по 10 нових слів і підходить для всіх рівнів володіння англійська мова. Основним плюсом є те, що існує зручна система, яка допомагає відслідковувати прогрес вивчення слів, а так само приємний

інтерфейс. Основним недоліком виступає те, що безкоштовна підписка триває всього-на-всього 7 днів, а подальше використання дороге [26].

Аналізуючи переваги та недоліки вищезгаданих мобільних додатків, можна зробити висновок про те, що існує реальна можливість зайняти сегмент ринку членів освітньої діяльності, котрі заінтересовані у вивченні та вдосконаленні англійської мови.

Подану ідею можна досягти шляхом створення мобільного додатку, котрий буде виключати недоліки конкуруючих мобільних додатків та мати власні унікальні переваги.

Таким чином, мобільний додаток «AnyWord» – це прикладна програма, створена для досягнення мети покращення володіння англійською мовою серед членів освітньої діяльності України шляхом набуття нової для них лексики.

1.3. Формування вимог до системи та вибір технології розробки

За своїм призначенням система створення мобільного додатку повинна забезпечувати швидкий, зручний, безпечний та взаємодіючий з іншими мовами програмування процес створення прикладної програми, яка забезпечить виконання основного завдання – допомога при вивченні англійської мови членами освітньої діяльності.

Основними вимогами до системи є:

– орієнтованість на користувачів з операційною системою Android;

– можливість створення власного словника та додавання або видалення нових слів, котрі певний користувач має бажання вивчити;

– можливість додавання користувацьких зображень для формування унікальних асоціативних зв’язків під кожного користувача;

– зрозумілий та інтуїтивний інтерфейс.

Порівняння основних інструментів для створення додатку

Одними з найпопулярніших інструментів для створення мобільних додатків є ReactNative, Xcode та Android Studio.

1) ReactNative – це фреймворк, котрий має відкритий вихідний код для створення мобільних додатків на мовах JavaScript та TypeScript, який підтримує такі мобільні операційні системи, як Android та IOS одночасно. Це є значною перевагою поданого фреймворку, бо пишучи один код, створюється 2 мобільні додатки для різних ОС. Основними недоліками поданого методу є те, що ReactNative ще є «молодим» фреймворком (це означає, що не всі потрібні для створення мобільного додатку компоненти існують, що потрібно постійно слідкувати за версіями використаних бібліотек), і, як наслідок, у певний момент, без своєчасного втручання, мобільний додаток може перестати бути активним. Другим основним недоліком є те, що адаптація дизайну для різних девайсів з операційною системою Android є надзвичайно складною задачею, що негативно позначиться на його роботі в майбутньому [37, 39].

2) Xcode – це інтегроване середовище розробки програмного забезпечення для платформи IOS. Є основним інструментом для створення мобільних додатків для смартфонів компанії Apple. На сьогодні основною мовою програмування поданого середовище є мова Swift. Swift – це відкрита нативна мова програмування для розробки прикладних програм для операційної системи IOS. Основним перевагами є легкість освоєння та застосування поданої мови, її функціональна безпечність, наявність вихідного коду, взаємодія та інтеграція попередньої мови розробки (Objective-C), надзвичайно великий потенціал за рахунок крос-платформеності (у майбутньому планується доступ навіть на Windows) та її функціональна сумісність. До недоліків відносяться наступні пункти: обмежений кадровий потенціал (доступ лише з операційної системи macOS), мала кількість бібліотек (подана мова програмування є «молодою») та її нестабільність, що проявляється при оновленні версій поданої мови [31, 42].

3) Android Studio – це інтегроване середовище розробки для роботи з операційною системою Android. Є основним інструментом для створення мобільних додатків для смартфонів на Android. На сьогодні основною мовою програмування поданого середовище є мова Kotlin [19, 33].

За визначенням Мартіна Хеллера (редактора та доповідача InfoWorld – журналу в сфері інформаційних технологій, колишнього консультанту з питань Windows та віце-президента з питань технологій та освіти в Alpha Software), Kotlin – це загальна, безкоштовна, з відкритим кодом, статично набрана «прагматична» мова програмування, спочатку розроблена для JVM (Java Virtual Machine) та Android, яка поєднує в собі об'єктно-орієнтовані та функціональні функції програмування [30, 32].

Проте досі актуальною є й попередня мова програмування Java. Для того, щоб обрати найкращий з двох варіантів, потрібно провести порівняння.

Android-розробники надають перевагу мові програмування Kotlin. Використовувати подану мову простіше та зручніше. Це дає можливість писати більш зрозумілий та простий код, зменшує вірогідність виникнення помилок та оптимізує час створення мобільного додатку. Також існує думка, що в майбутньому Kotlin витіснить мову Java, так як Kotlin повністю сумісний з Java [22].

Kotlin виглядає як більш стисла і впорядкована версія Java. На прикладі рисунку 1.1, де відбулась автоматизація зразку коду Java в Kotlin. Звертаючи увагу на повторення, властиве ініціюванню змінних Java, можна помітити, що воно зникло та відбулось спрощення коду, що надзвичайно корисно для роботи програміста.

Ідіома Java:

```
StringBuilder sb = new StringBuilder();
```

Стає у Kotlin:

```
val sb = StringBuilder()
```

ExampleJava.java

```java
package com.example.anyword.examples;

import ...

public class ExampleJava {
    public String toJSON(Collection<Integer> collection) {
        StringBuilder sb = new StringBuilder();
        sb.append("[");
        Iterator<Integer> iterator = collection.iterator();
        while (iterator.hasNext()) {
            Integer element = iterator.next();
            sb.append(element);
            if (iterator.hasNext()) {
                sb.append(", ");
            }
        }
        sb.append("]");
        return sb.toString();
    }
}
```

ExampleKotlin.kt

```kotlin
package com.example.anyword.examples

class ExampleKotlin {
    fun toJSON(collection: Collection<Int?>): String {
        val sb = StringBuilder()
        sb.append("[")
        val iterator = collection.iterator()
        while (iterator.hasNext()) {
            val element = iterator.next()
            sb.append(element)
            if (iterator.hasNext()) {
                sb.append(", ")
            }
        }
        sb.append("]")
        return sb.toString()
    }
}
```

Рисунок 1.1 – Порівняння коду Java та Kotlin

У мові програмування Kotlin функції ініціалізуються за допомогою ключового слова «fun», а крапка з комою «;» нині не є потрібною у застосуванні. Ключове слово «val» може бути використане лише для властивості зчитування. У свою чергу, ключове слово «var» має змінну властивість або локальну змінну

Kotlin – надзвичайно типізована мова програмування. Змінні, індексовані за допомогою ключових слів «val» і «var» можуть використовуватися лише при вказаному типі. На рисунку 1.2 можна побачити декілька прикладів ініціалізації та присвоєння змінних.

```kotlin
val exampleValFirst: String = "Перший приклад типізації val"
val exampleValSecond = "Другий приклад типізації val"
var exampleVar: String? = null

fun toJSON(collection: Collection<Int?>): String {
    exampleVar = "Перший приклад типізації var"
```

Рисунок 1.2 – Приклади ініціалізації та присвоєння змінних

Змінна «exampleValFirst» має тип String. Так, як тип поданої змінної «val», то необхідно одразу ж надати їй значення, яке буде записане за допомогою лапок. Лапки відповідають типу String.

Змінна «exampleValSecond» також має тип String, проте записана вже без його визначення. Необхідності у цьому немає, бо лапки типізують змінну до потрібного нам значення.

Змінна «exampleVar» є локальною змінною, тому при її ініціалізації обов’язково потрібно вказувати тип. І вже при використанні змінної проводиться присвоєння значення.

Властивість типізації і манера її використання покращується і спрощується з кожним оновленням мови Kotlin.

Також Kotlin послабила вимогу Java, при якій функції були членами класу. У Kotlin функції можуть бути оголошені на верхньому рівні у файлі, локально всередині інших функцій, як функція-член в середині класу чи об'єкта та як функція розширення.

Ще однією відмінністю від Java є те, що Kotlin не має ключового слова new. Для того, щоб створити екземпляр класу, відтепер можливо викликати конструктор так само, як і звичайну функцію.

Класи Kotlin повинні бути позначені ключовим словом open, щоб дозволити іншим класам успадковувати їх. Класи Java успадковуються, якщо вони не позначені ключовим словом final. Щоб замінити метод суперкласу, сам метод повинен бути позначений open, а метод підкласу – позначений override.

Таким чином, проаналізувавши всі інструменти створення мобільних додатків було прийняте рішення використовувати інтегроване середовище розробки Android Studio з нативною мовою програмування Kotlin.

РОЗДІЛ 2. РЕАЛІЗАЦІЯ ПРОТОТИПУ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ

2.1. Проектування архітектури мобільного додатку

Моделювання – це метод дослідження реальних і абстрактних об’єктів-прототипів на умовних образах, схемах, фізичних об’єктах, що відрізняються від прототипу, але аналогічні йому за будовою чи типом поведінки, із застосуванням методів аналогії, теорії подібності й теорії обробки даних експерименту [4, 35].

У свою чергу, проектування – це процес визначення архітектури, компонентів, інтерфейсів та інших характеристик системи або її частини. Результатом проектування є проект – цілісна сукупність моделей, властивостей або характеристик, описаних у формі, придатній для реалізації системи [1, 35].

Архітектура мобільного додатку – це спосіб структурування прикладної програми [8], абстракція елементів системи на певній фазі її роботи. Система може складатись з кількох рівнів абстракції і мати багато фаз роботи, кожна з яких може мати окрему архітектуру [29].

Таким чином, за допомогою сайту app.diagrams.net було створено структуру мобільного додатку для того, щоб мати попереднє уявлення про мобільний додаток, його візуальне відображення та логіку, що зображено на рисунку 2.1.

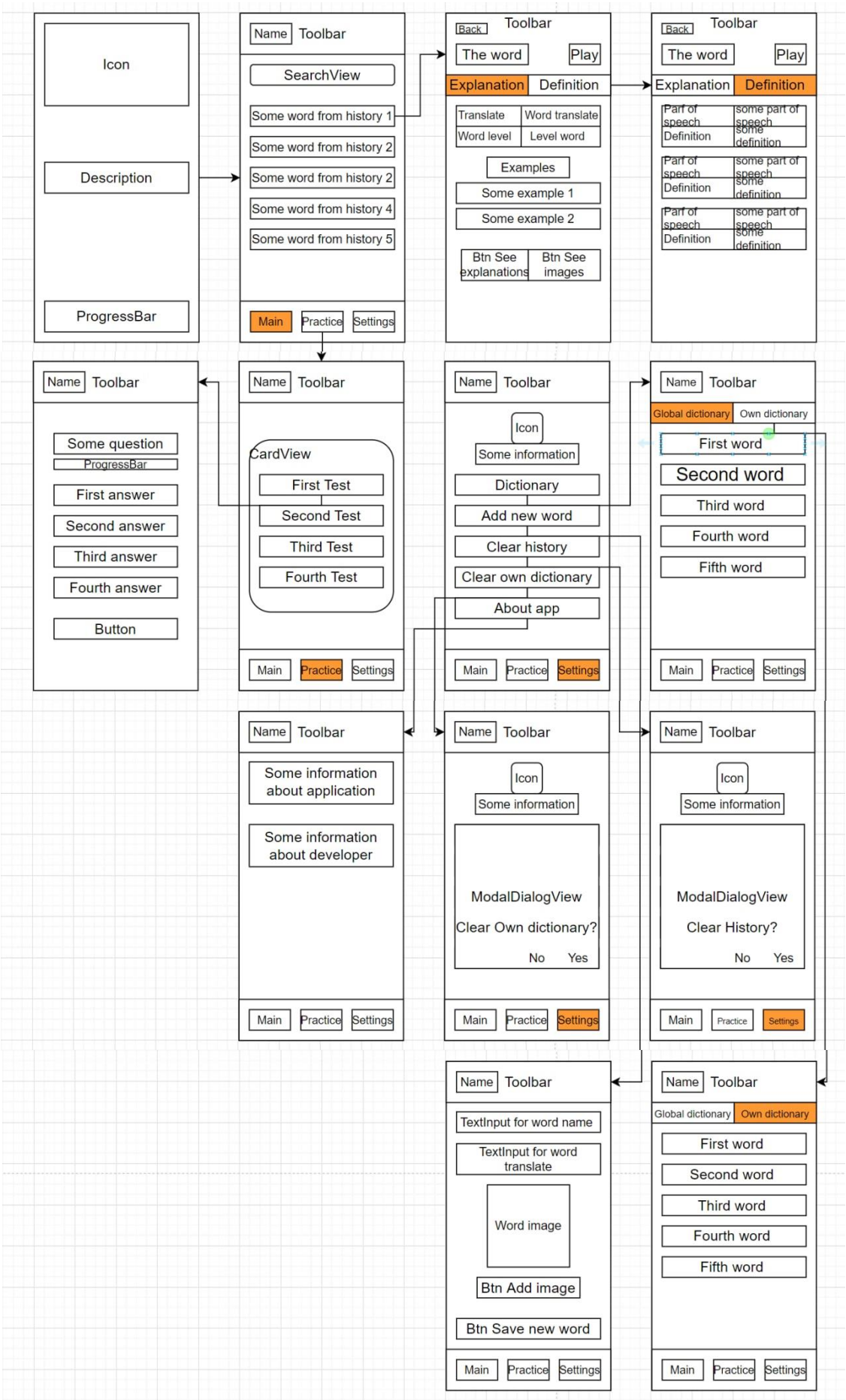

Рисунок 2.1 – Архітектура мобільного додатку AnyWord

Для забезпечення роботи back-end мобільного додатку, була створена база даних під назвою «dictionary.db». Подана БД містить 3 таблиці: «words», «words_new» та «history» (рис. 2.2-2.4)

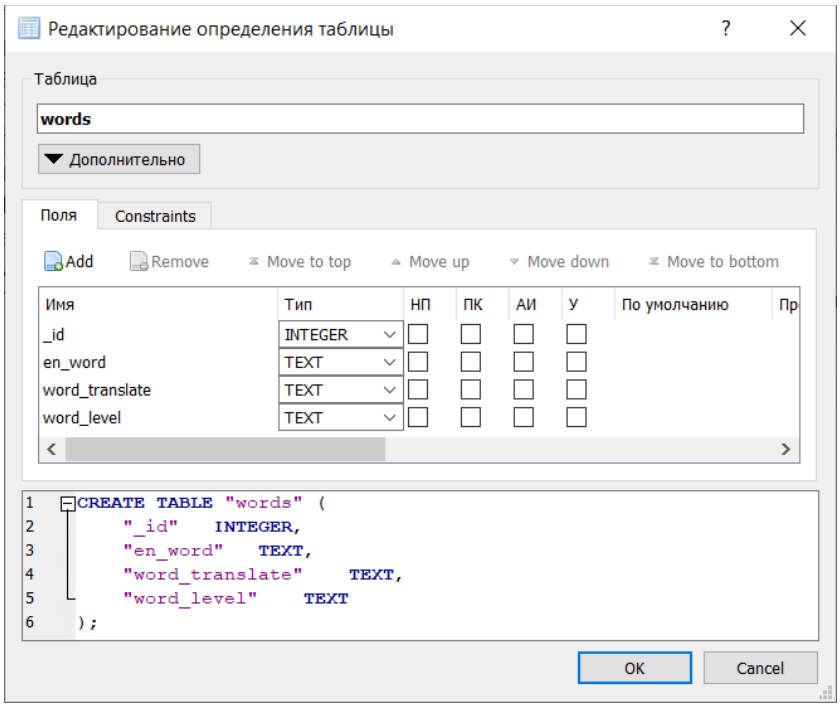

Рисунок 2.2 – Створення таблиці «words»

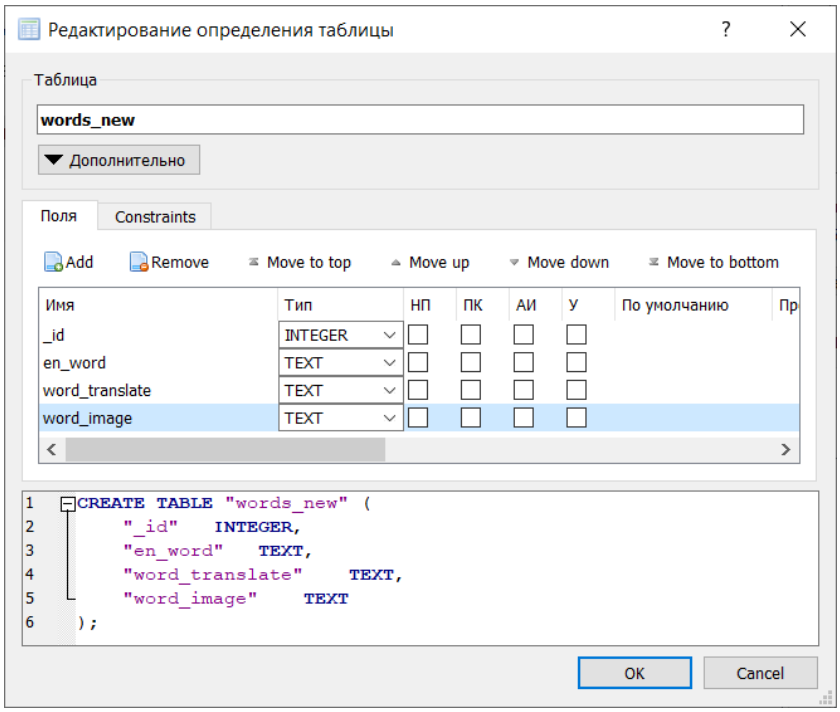

Рисунок 2.3 – Створення таблиці «words_new»

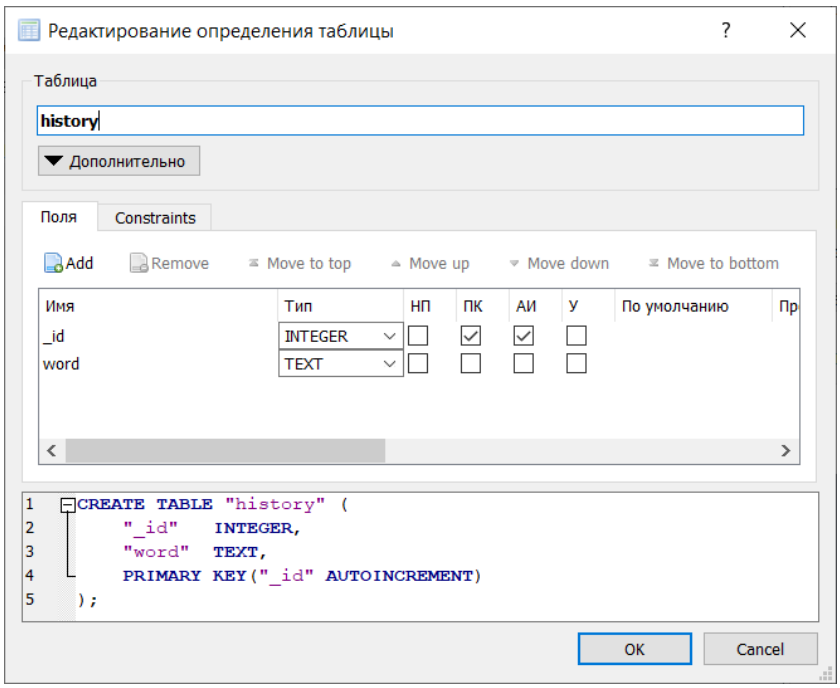

Рисунок 2.4 – Створення таблиці «history»

Розглянемо детальніше кожну зі створених вище таблиць.

Структура таблиці «words» зберігає ідентифікаційний номер слова, саме слово англійською мовою, його переклад та рівень складності слова.

Таблиця 2.1 – Структура таблиці «words»

Номер атрибуту	Назва атрибуту	Тип даних
1	_id	INTEGER
2	en_word	TEXT
3	word_translate	TEXT
4	word_level	TEXT

Артибут «_id» має унікальне значення й визначає кожен з рядків таблиці.

Подана таблиця унікальна тим, що вона вже заповнена 3-ма тисячами слів рівнем складності від A1 до B2, які користувач може використовувати для отримання перекладу слова без потреби в інтернеті, що зображено на рисунку 2.5.

Таблица: words

	_id	en_word	word_translate	word_level
	Фи...	Фильтр	Фильтр	Фильтр
831	1202	GYM	спортзал	A1
832	2280	SATURDAY	субота	A1
833	2568	SUGAR	цукор	A1
834	2970	YOUNG	молодий	A1
835	2647	TEXT	текст	A1
836	674	DECIDE	вирішити	A1
837	1226	HEALTH	здоров'я	A1
838	2496	SPORT	спорт	A1
839	312	BOTH	обидва	A1
840	2757	TRY	спробуй	A1
841	2414	SIXTEEN	шістнадцять	A1
842	2758	TUBE	трубки	A1
843	916	EURO	євро	A1
844	2499	SPRING	весна	A1
845	1214	HARD	важко	A1
846	2295	SCIENTIST	учений	A1
847	2576	SUMMER	літо	A1
848	2805	UNUSUAL	незвичний	A2
849	2335	SERVE	подавати	A2
850	808	DRUG	ліки	A2

Рисунок 2.5 – Заповнена даними таблиця «words»

Структура таблиці «word_new» зберігає ідентифікаційний номер слова, слово англійською мовою, його переклад та зображення, котре може обрати користувач для найкращого опису даного слова.

Таблиця 2.2 – Структура таблиці «words_new»

Номер атрибуту	Назва атрибуту	Тип даних
1	_id	INTEGER
2	en_word	TEXT
3	word_translate	TEXT
4	word_image	TEXT

Останньою таблицею є таблиця «history». Вона зберігає ідентифікаційний номер слова, та саме слово. Дана таблиця необхідна для формування списку історіє перегляду слів.

Таблиця 2.3 – Структура таблиці «history»

Номер атрибуту	Назва атрибуту	Тип даних
1	_id	INTEGER
2	word	TEXT

Для подальшої роботи з мобільним додатком також були створені схеми алгоритму дій [16]. Це точні розпорядження послідовності певних дій, необхідних для досягнення певної мети.

Отже, загальний алгоритм роботи з мобільним додатком має наступний вигляд (рис. 2.6):

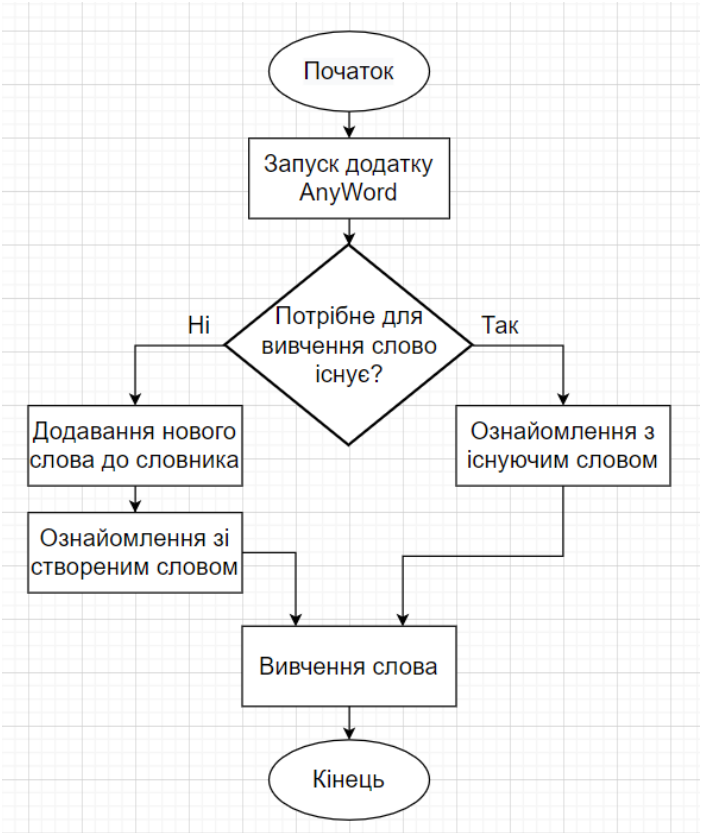

Рисунок 2.6 – Загальний алгоритм роботи з додатком AnyWord

Вищезгаданий алгоритм дій відображає інтуїтивне управління мобільним додатком для досягнення основної мети – покращення володіння англійською мовою шляхом поповнення власного лексичного словника.

Тепер потрібно проаналізувати процес додавання нового слова до словника. Поданий алгоритм зображений на рисунку 2.7.

Рисунок 2.7 – Алгоритм додавання нового слова до власного словника

Виконання поданого алгоритму дій забезпечить додавання нового слова до власного словника, що гарантує подальшу роботу зі словом.

Аналізуючи подану схему, потрібно звернути увагу на пункт «Вибір зображення». Поданий процес є унікальним у даному мобільному додатку. Саме тому був розроблений ще один алгоритм дій для поданого пункту (рис. 2.8).

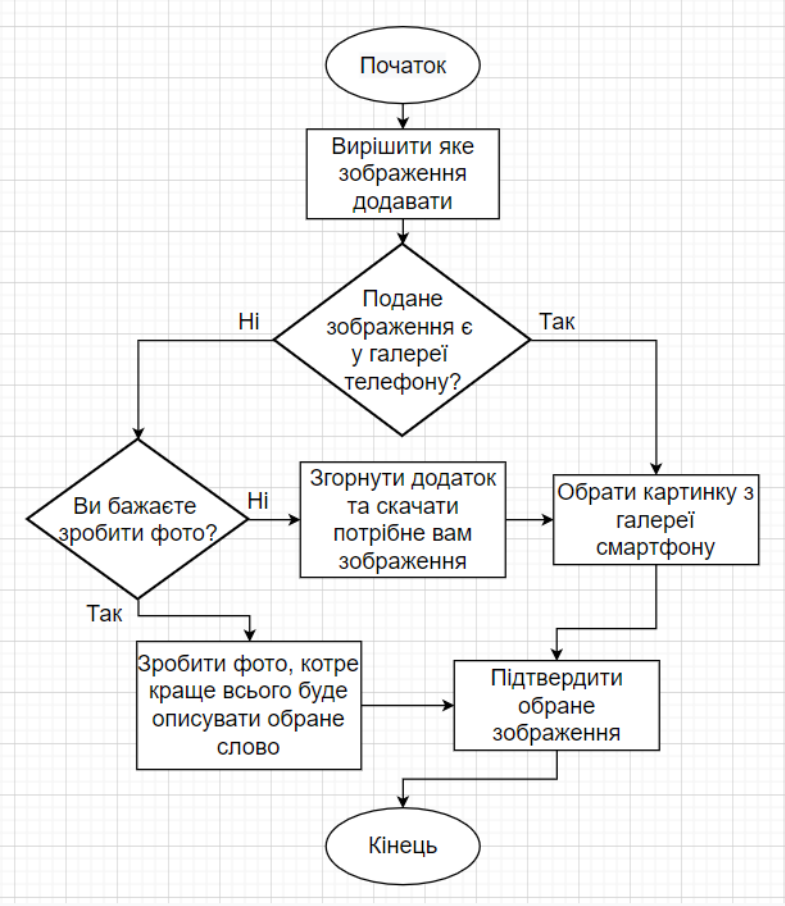

Рисунок 2.8 – Алгоритм додавання зображення до слова

Аналізуючи поданий алгоритм дій можна зробити висновок про те, що у мобільному додатку реалізовано 2 способи додавання зображення до слова. Перший спосіб представляє собою вибір зображення, які містить у собі Галерея смартфону. Другий спосіб має можливість роботи фотографії у режимі реального часу. Користувач має можливість вибору способу додавання зображення до слова.

Тепер, коли користувач вже має як загальний так і власний словники з заповненими словами, він може почати процес ознайомлення зі словом, котрий зображений на рисунку 2.9.

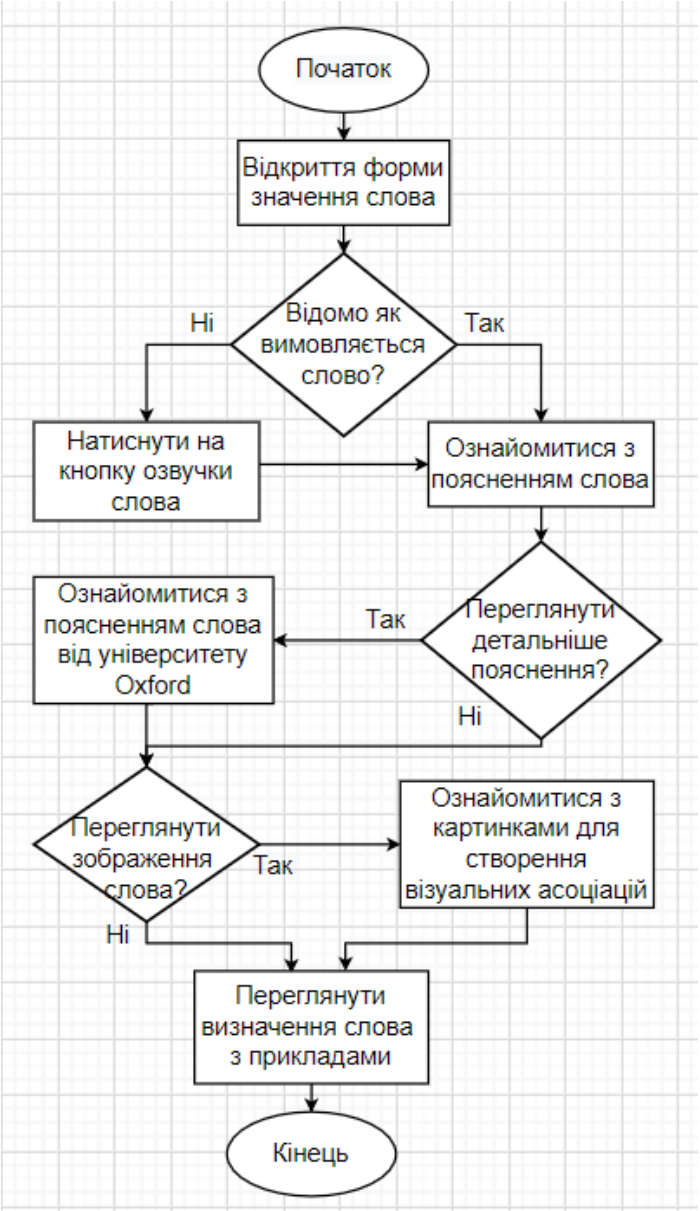

Рисунок 2.9 – Алгоритм ознайомлення зі словом

Виконання поданого алгоритму гарантує ознайомлення з раніше обраним словом користувачем. Даний процес містить декілька етапів: озвучення слова, його переклад та рівень складності (за умови, що слово з загального словника), приклади його застосування та пояснення при набутті слова різних частинах мови, детальний опис на сайті Oxford Learner's Dictionaries, величезна кількість картинок з Google Images та власне зображення слова (при умові, що під час додавання слова до власного словника було обране це саме зображення).

Також, якщо користувач має намір видалити слово з власного словника, він, звісно ж, має змогу це зробити. Схема дій зображена на рисунку 2.10.

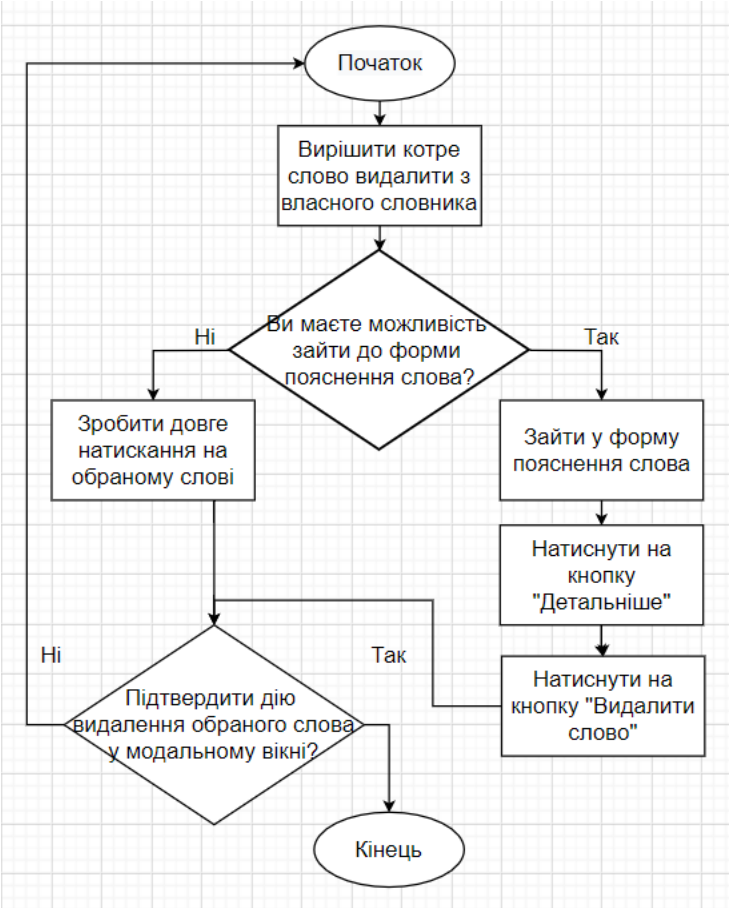

Рисунок 2.10 – Алгоритм видалення слова

Аналізуючи поданий алгоритм, можна помітити, що існує 2 способи видалення слова з власного словника. За допомогою першого методу цю дію можна виконати зі списку всіх раніше доданих слів, шляхом довгого натискання на обране слово. Для виконання другого методу необхідно зайти до форми пояснення слова, і вже в ній виконати видалення слова. Подане розгалуження дає змогу користувачу обирати яким для нього способом зручніше це зробити, що, у свою чергу, збільшує його свободу роботи з мобільним додатком.

2.2. Реалізація інформаційного забезпечення мобільного додатку

Проведемо аналіз основних класів, необхідних для коректного функціонування мобільного додатку.

Першою завантажувальною формою прикладної програми є LoadingActivity. Основний код знаходиться у лістингу Б.1. Розглянемо умовний оператор if, котрий запускається в основній функції життєвого циклу діяльності – onCreate (рис. 2.11).

```
databaseHelper = DatabaseHelper( context: this)
if (databaseHelper?.checkDatabase() == true) {
    openDatabase()
    Log.w( tag: "Form-LoadingActivity",  msg: "databaseHelper?.checkDatabase() == true")
} else {
    val loadDatabaseAsync = LoadDatabaseAsync( context: this)
    loadDatabaseAsync.execute()
    Log.w( tag: "Form-LoadingActivity",  msg: "databaseHelper?.checkDatabase() == false")
}
```

Рисунок 2.11 – Код умовного оператору if

Аналізуючи подану частину коду, можна зробити висновок про те, що при першому запуску мобільного додатку, виконується функція openDatabase(), котра відкриває базу даних. І вже при повторних запусках прикладної програми, буде виконуватися команда loadDatabaseAsync. execute(), котра виконує загрузку БД.

Спочатку розглянемо частину коду, котра виконується після невідповідності вимоги if. Подана частина коду відсилається на клас LoadDatabaseAsync. Основний код якого знаходиться у лістингу Г. 1.

Тепер проаналізуємо код, котрий викликається при відповідності вимоги if. У цьому випадку викликається функція openDatabase(), розташована нижче на формі LoadingActivity (рис. 2.12).

```
@SuppressLint( ...value: "LongLogTag")
private fun openDatabase() {
    try {
        databaseHelper?.openDatabase()
        databaseOpened = true
    } catch (e: SQLException) {
        Log.w( tag: "Form-LoadingActivity-openDatabase-error",  msg: "$e")
    }
}
```

Рисунок 2.12 – Код функції openDatabase()

Функція openDatabase() є асинхронною функцією. Асинхронна функція – це функція, яка виконуються паралельно основному потоку, або в тому ж процесі, або взагалі в іншому процесі [18, 38]. Ключова відмінність асинхронного режиму: паралельне виконання двох і більше гілок процесу. У мові програмування Kotlin подані функції позначаються шляхом застосування блоку try {} catch {}. Подана функція відкриває базу даних, посилаючись на код класу DatabaseHelper з лістингу Г.2.

Отже, викликається функція з класу DatabaseHelper, що зображено на рисунку 2.12.

```
@Throws(SQLException::class)
fun openDatabase() {
    val path = DB_PATH + DB_NAME
    myDatabase = SQLiteDatabase.openDatabase(path,  factory: null, SQLiteDatabase.OPEN_READWRITE)
}
```

Рисунок 2.12 – Функції openDatabase()

Змінна path складається з двох частин: назви БД та шляху знаходження поданої БД. Код поданих змінних розташований нижче (рис. 2.13).

```
companion object {
    private const val DB_NAME = "dictionary.db"
}
init {
    DB_PATH =context.filesDir.path + DB_NAME;
    DB_PATH = "/data/data/" + context.packageName + "/databases/"
    Log.w( tag: "DB_PATH", DB_PATH.toString())
}
```

Рисунок 2.13 – Константи DB_NAME та DB_PATH

Отже, за допомогою вбудованої в AndroidStudio бази даних SQLiteDatabase, було відкрито нашу Database.

Тепер проаналізуємо основні функції класу DatabaseHelper.

Запит getHistory повертає раніше переглянуті слова з загального словника, що допомогає формувати адаптивний масив зі слів-карток на формі MainActivity (рис. 2.14), основний код якої розташований на лістингу Б.2.

```
val getHistory: Cursor?
    get() = myDatabase?.rawQuery(
        sql: "select distinct word, word_translate from history h join words w on h.word==w.en_word order by h._id desc",
        selectionArgs: null
    )
```

Рисунок 2.14 – Запит getHistory

Подана функція вибирає значення «word» з таблиці «history» та «word_translate» з таблиці «words» порівнюючи їх ідентифікаційні номери.

Наступною основною функцією роботи з БД є функції getMeaningFromGlobal та getMeaningFromOwn (рис. 2.15).

```
fun getMeaningFromGlobal(text: String): Cursor? {
    return myDatabase?.rawQuery(
        sql: "SELECT word_translate, word_level FROM words WHERE en_word==UPPER('$text')",
        selectionArgs: null
    )
}
fun getMeaningFromOwn(text: String): Cursor? {
    return myDatabase?.rawQuery(
        sql: "SELECT word_translate, word_image FROM words_new WHERE en_word==UPPER('$text')",
        selectionArgs: null
    )
}
```

Рисунок 2.15 – Код функцій getMeaningFromGlobal та getMeaningFromOwn

Подані запити обирають значення слова з таблиці «words» та «words_new» відповідно, і відображають на формі WordMeaningActivity, основний код якої знаходиться на лістингу В.2.

Тепер розглянемо функцію insertDataIntoOwn. Це основна функція, яка формує запит на додавання нового слова до власного словника (рис. 2.16).

```
fun insertDataIntoOwn(en_word: String, word_translate: String, word_image: String) {
    myDatabase?.execSQL( sql: "INSERT INTO words_new(en_word, word_translate, word_image) " +
            "VALUES (UPPER ('$en_word'), UPPER('$word_translate'), UPPER('$word_image'))")
}
```

Рисунок 2.16 – Функція insertDataIntoOwn

Подана функція має 3 обов’язкові для прийняття параметри типу String, котрі будуть вставлені до таблиці «words_new».

Тепер проведемо аналіз функцій, котрі формують запити на отримання даних з глобального і власного словників (рис. 2.17).

```
val getDictionaryGlobal: Cursor?
    get() = myDatabase?.rawQuery(
        sql: "SELECT  en_word, word_translate FROM words GROUP BY en_word",
        selectionArgs: null
    )
val getDictionaryOwn: Cursor?
    get() = myDatabase?.rawQuery(
        sql: "select distinct en_word, word_translate from words_new order by _id desc",
        selectionArgs: null
    )
```

Рисунок 2.17 – Запити getDictionaryOwn та getDictionaryGlobal

Кожен з поданих запитів формує адаптивний масив, що відображається на формі DictionaryActivity, основний код якої розташований на лістингу Е.1.

Функція видалення слова з власного словника має наступний вигляд:

Функція видалення слова з власного словника приймає значення слова та видаляє його з таблиці «words_new» (рис. 2.18).

```
fun deleteFromNewWords(en_word: String) {
    myDatabase?.execSQL( sql: "DELETE FROM words_new WHERE en_word==UPPER('$en_word')")
}
```

Рисунок 2.18 – Функція deleteFromNewWords

Останніми основними функціями роботи з БД є функції очищення історії перегляду слів та власного словника (рис. 2.19).

```
fun deleteHistory() {
    myDatabase?.execSQL( sql: "DELETE FROM history")
}
fun deleteNewWords() {
    myDatabase?.execSQL( sql: "DELETE FROM words_new")
}
```

Рисунок 2.19 – Функції deleteHistory() та deleteNewWords()

Кожна з поданих функцій має запит, котрий повністю очищує таблиці «history» та «words_new» відповідно.

Тепер розглянемо код класу ApiForMainActivity, повний код якого розташована на лістингу В.1.

Поданий клас приймає 2 обов'язкових параметри, необхідних для подальшої роботи: Intent та FragmentManager. Перший, у свою чергу, передає інформацію про те, на яку діяльність потрібно здійснити перехід після виконання запиту. Другий же необхідний для функціонування діалогового вікна, котрий візуалізує статус «завантаження» (рис. 2.20):

```
class ApiForMainActivity(var intent: Intent, var fragmentManager: FragmentManager?) {
```

Рисунок 2.20 – Клас ApiForMainActivity

Змінні client та dialogProgressIndicator – це клієнт сторонньої бібліотеки для створення запитів та діалогове вікно, створене для відображення статусу завантаження (рис. 2.21).

```
private val client = OkHttpClient()
val dialogProgressIndicator = DialogProgressIndicator()
```

Рисунок 2.21 – Змінні client та dialogProgressIndicator

OkHttp – це стороння бібліотека, розроблена Square для надсилання та отримання мережевих запитів на основі HTTP. Побудований на бібліотеці Okio, яка намагається ефективніше читати та писати дані, ніж стандартні бібліотеки вводу-виводу Java, створюючи спільний пул пам'яті [24, 36].

Функція getWordMeaning містить обов’язковий параметр типу String, котрий формує кінець адреси електронного ресурсу.

Змінна «request» формує запит на сторону API – Application Programming Interface – опис способу взаємодії програми з іншою програмою [20,40] – який містить обов’язкові параметри, як:

– .url – адреса електронного ресурсу;
– .get() – вид запиту, що означає отримання інформації;
– .addHeader – формує ключі допуску до інформації;
– .build() – створення запиту.

У випадку (!response.isSuccessful) виконується перша частина умовного оператору if, котра припиняє діяльність змінної dialogProgressIndicator та інформує про помилку у запиті.

Після успішного виконання запиту, необхідно звернути увагу на змінну «json», котра формує об’єкт з отриманою інформацією.

Код успішної операції продемонстровано на рисунку 2.22.

```
val json = JSONObject(response.body()!!.string())
Log.w( tag: "Form-ApiForMainActivity-getWordMeaning-True", json.toString())

if (apiEnd != "examples") {
    val responseArray = json.getJSONArray( name: "definitions")
    wordMeaning = responseArray.toString()
    Log.w( tag: "Form-ApiForMainActivity-getWordMeaning-wordMeaning", wordMeaning)

    apiEnd = "examples"
    getWordMeaning(apiEnd)
} else {
    val responseArray = json.getJSONArray( name: "examples")
    wordExamples = responseArray.toString()
    Log.w( tag: "Form-ApiForMainActivity-getWordExamples-wordExamples", wordExamples)

    dialogProgressIndicator.dismiss()
    intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK
    startActivity(mainContext!!, intent, intent.extras)
}
```

Рисунок 2.22 – Код успішного виконання запиту

Аналізуючи поданий код, потрібно звернути увагу на новий умовний оператор if (apiEnd != "examples"). Після успішного проходження першого запиту, виконується код, що відповідає умові оператору. Таким чином формується масив з даними «wordMeaning», після чого змінна «apiEnd», котра при визові класу набувала значення змінної «end» тепер набуває значення «examples» та заново запускається функція getWordMeaning(apiEnd), проте вже з іншим параметром. Отже, після другого успішного виконання запиту, виконується код., записаний в операторі «else», після чого формується новий масив даних «wordExamples», припиняється діяльність змінної dialogProgressIndicator та відбувається перехід до нової форми, значення якох містить змінна «intent».

Таким чином, за допомогою умовного оператору «if else» було досягнуто спрощення коду, котрий виконує подвійний запит на API.

Інший код прикладної програми знаходиться у додатках Б-Е.

2.3. Контрольний приклад та інструкція щодо використання

Тепер, змоделювавши та спроектувавши архітектуру мобільного додатку, розробивши локальну базу даних, алгоритми дій, що описують функціонал прикладної програми та написавши код, ми маємо контрольний приклад мобільного додатку «AnyWord».

Додаток AnyWord складаються з наступних форм:

1) Форма загрузки. Відображає статус завантаження додатку. Має 2 анімаційних елементи, котрі змінюються з кожною секундою до моменту завантаження StatusBar на 100% (рис. 2.23).

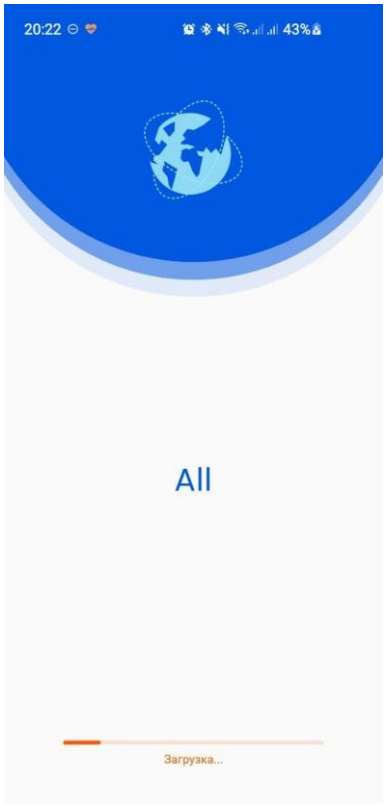

Рисунок 2.23 – LoadingActivity

2) Головна сторінка. Є одною з 3-х головних діяльностей мобільного додатку. При першому запуску містить декілька системних модальних вікон, котрі запрошують доступ до смартфону (рис. 2.24).

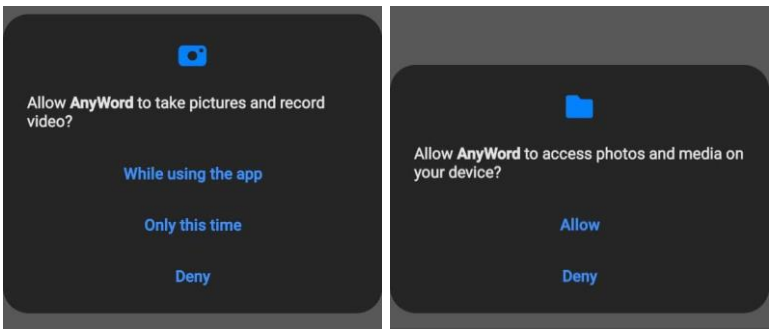

Рисунок 2.24 – Системні модальні вікна

Основна форма має Toolbar з назвою діяльності, SearchView, необхідний для пошуку слів, записаних у базі даних та TextView з повідомленням про те, що не відбувся пошук жодного слова. Після того, як відбувся пошук слова, замість текстового повідомлення відображається список карток, котрі містять коротку інформацію про слова. У нижній частині форми розташований BottomNavigationBar, за допомогою якого існує можливість переходу по головним діяльностям додатку.

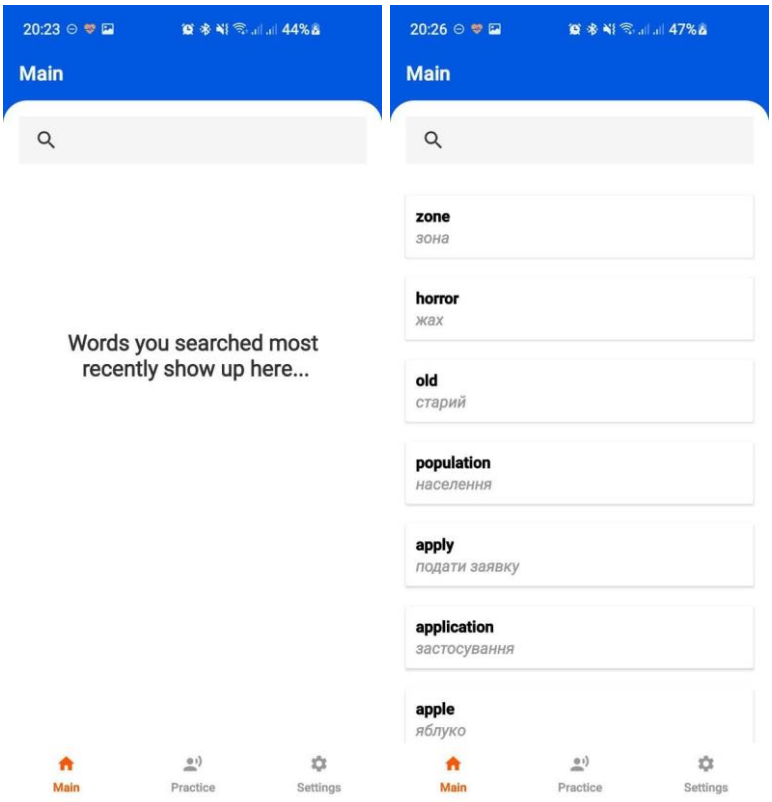

Рисунок 2.25 – MainActivity з пустою та заповненою історією

SearchView має пошуковий функціонал. При виборі певного слова, відбувається перехід на нову форму WordMeaningActivity (рис. 2.26).

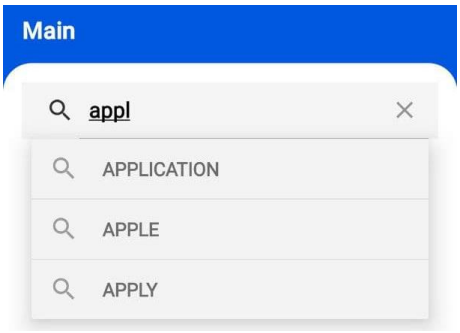

Рисунок 2.26 – SearchView

3) Форма значення слова. Містить Toolbar, на якому розташовано обране у SearchView слово; у правій частині знаходиться кнопка, котра його озвучує. Нижче розташований TabLayout, який розділяє форму на 2 фрагменти: «Word Explanation» (пояснення слова) та «Word Definition» (визначення слова).

Фрагмент Word Explanation містить у собі переклад слова, рівень його складності (подані значення знаходяться у створеній базі даних) та приклади його застосування, котрі є JSON масивом, отриманим після звернення до API. У свою чергу, фрагмент «Word Definition» містить масив з описом частини речення поданого слова та його відповідним визначенням (рис. 2.27).

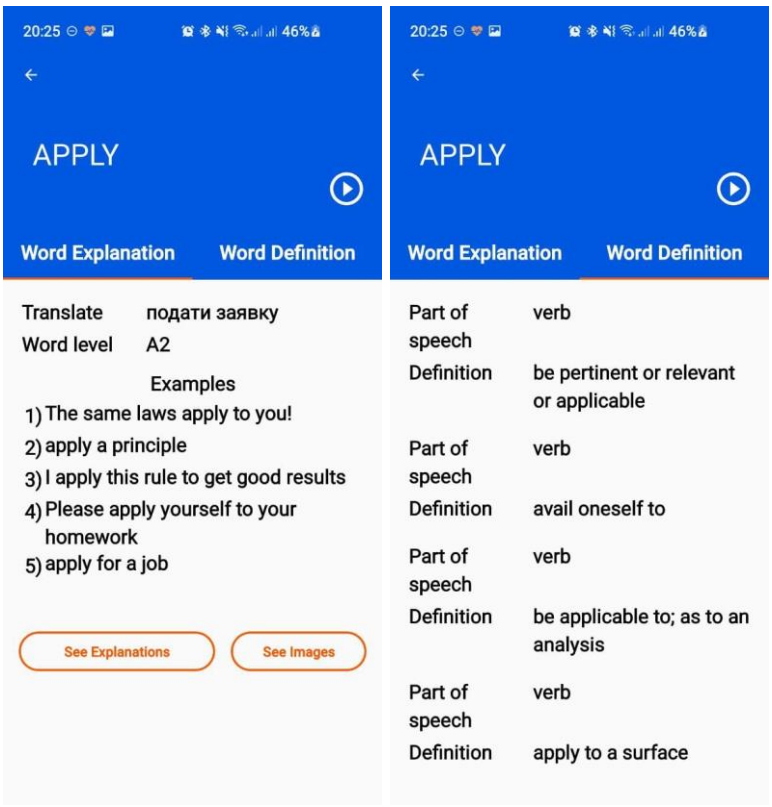

Рисунок 2.27 – Word Explanation

Нижче розташовані кнопки «See Explanations» та «See Images». Перша кнопка містить посилання на сайт Oxford Learner's Dictionaries, на якому міститься детальний опис слова, приклади його застосування та пояснення (рис. 2.28).

Кнопка «See Images» містить Google-images запит. Тому після натиснення на неї відбувається перехід на Google з відображенням усіх можливих візуалізацій.

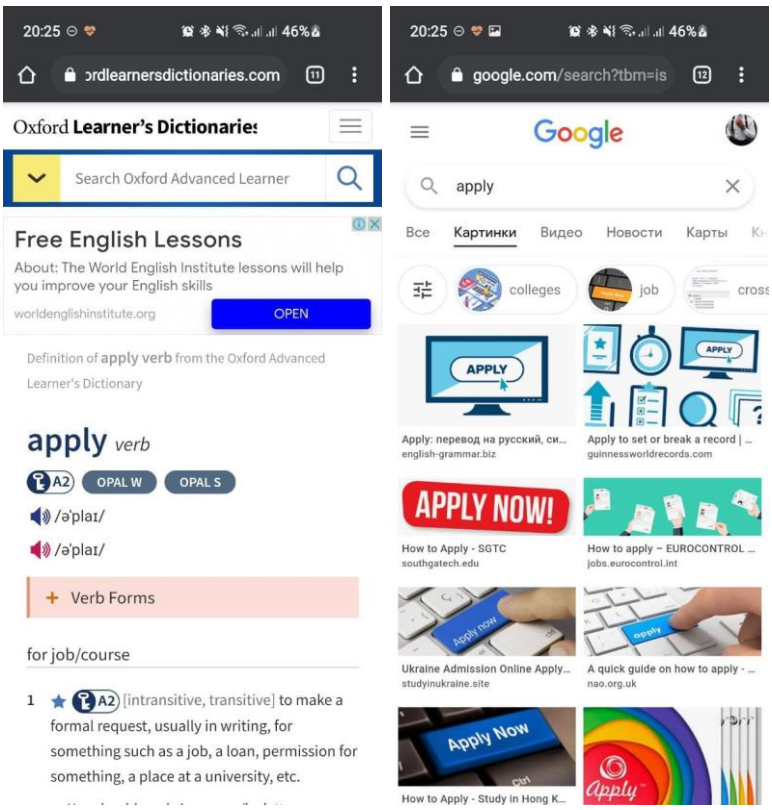

Рисунок 2.28 – Oxford Learner's Dictionaries та Google Images

4) Практика. Друга головна форма мобільного додатку. Подана діяльність відображає список доступних тестів за різними видами складності (від A1 до B2) на перевірку знання слів (рис. 2.29). Також має Toolbar з назвою форми та BottomNavigationBar для навігації по прикладній програмі.

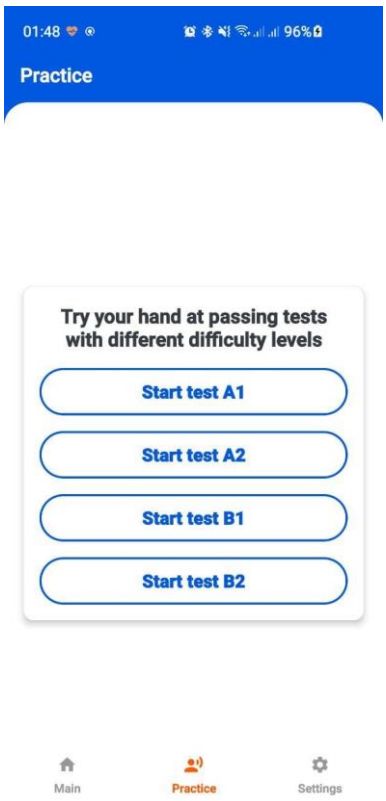

Рисунок 2.29 – PracticeActivity

Після вибору відповідного тесту, відбувається перехід на нову форму із завданням. Тест містить текстовий блок із запитанням, ProgressBar для відслідковування прогресу, 4-ри варіанти відповіді та кнопку перевірки результату. На рис. 2.30 продемонстровано ситуацію, при якій користувач дає правильну відповідь на запитання. На рис2.31 показано приклад, при якій користувач дає не правильну відповідь на запитання. Після проходження тесту відбувається повернення до форми з вибором всіх тестів, на котрій з’являється Toast, який інформує про успішне виконання завдання та результат його проходження (рис. 2.32).

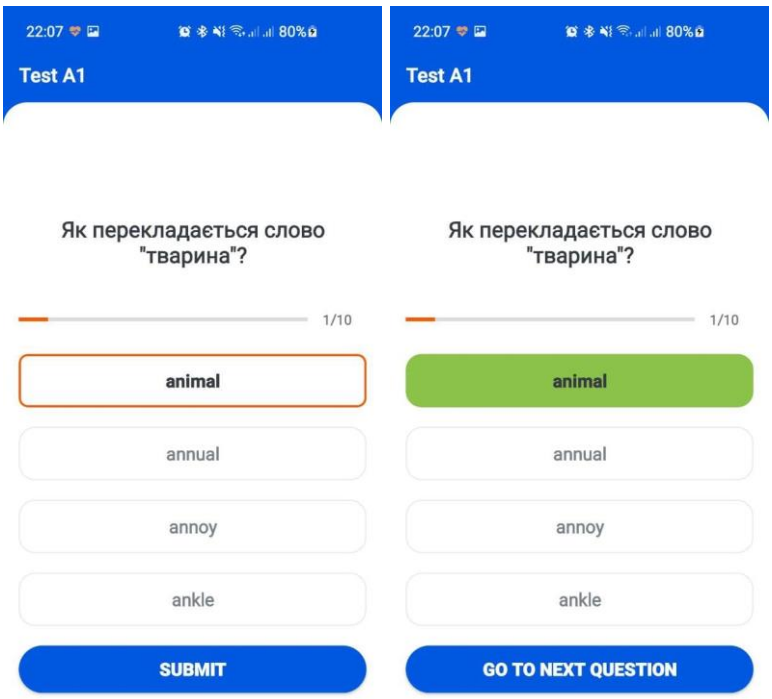

Рисунок 2.30 – Правильна відповідь на запитання

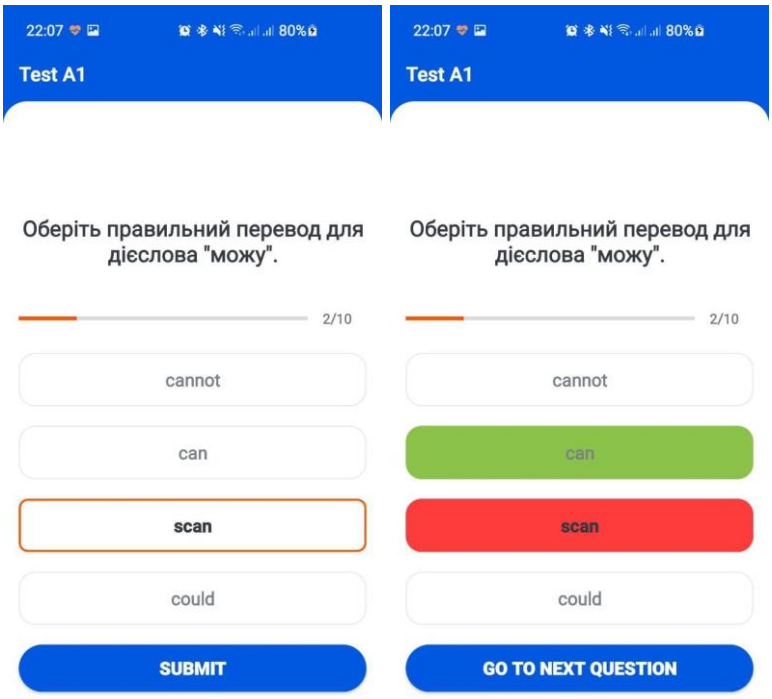

Рисунок 2.31 – Не правильна відповідь на запитання

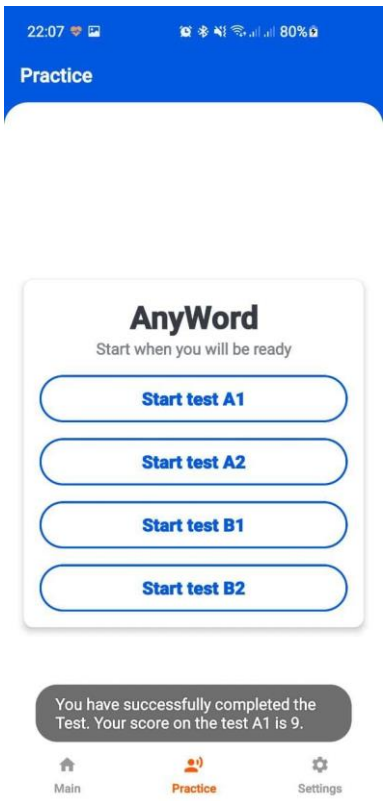

Рисунок 2.32 – Повідомлення з інформацією про пройдений тест

5) Налаштування. Остання з головних діяльностей мобільного додатку. У верхній частині має Toolbar з назвою форми. Під ним розташована іконка прикладної програми с короткою інформацією про номер версії та ким вона розроблена. Основна частина форми – це список кнопок: «Dictionary», «Add new word», «Clear History», «Clear own dictionary» та «About App». Кожна з котрих має певний функціонал, розписаний нижче. Також має BottomNavigationBar для навігації по прикладній програмі (рис. 2.33).

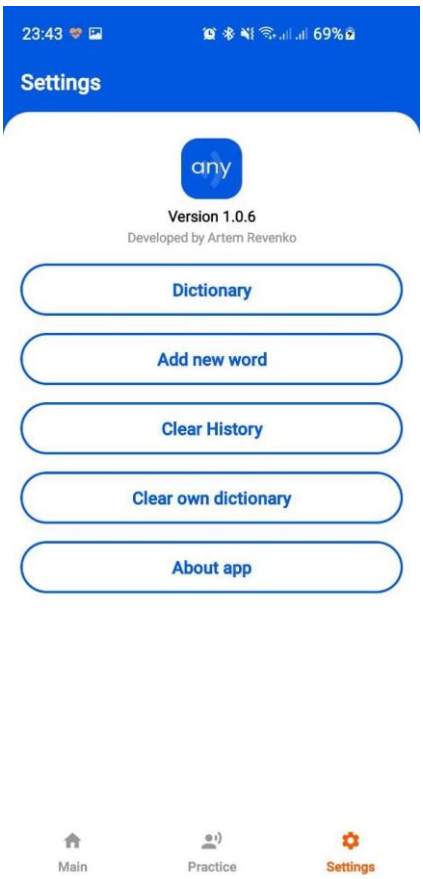

Рисунок 2.33 – SettingsActivity

Спочатку розглянемо функціонал кнопки «Clear History». Результатом натискання на неї буде поява модального діалогового вікна, який буде вимагати підтвердження або відхилення операції (рис. 2.34). Таким чином, при її підтвердженні буде очищена історія пошуку слів, що буде демонструвати пустий список на формі «MainActivity».

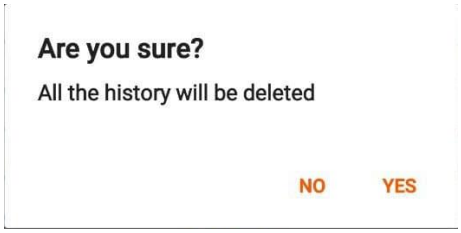

Рисунок 2.34 – Видалення історії пошуку слів

Подібний функціонал містить кнопка «Clear own dictionary». У новому діалоговому вікні потрібно так само підтвердити процес видалення всіх слів з власного словника (рис. 2.35).

Рисунок 2.35 – Очищення всіх слів з власного словника

6) Форма Словника. Після натискання на кнопку «Dictionary» відкривається подана активність. Подана діяльність має TabLayout, який розділяє форму на 2 фрагменти: «Global dictionary» (глобальний словник), який містить у собі приблизно 3000 тисяч слів від рівня елементарної англійської до рівня англійської вище середнього та «My own dictionary» (власний словник), котрий за замовчуванням є пустим, та у подальшому заповнюється власними новими словами (рис. 2.36). При натисканні на кожний з елементів відбувається перехід на форму значення слова (аналогічно до роботи з SearchView або масиву історії пошукових слів).

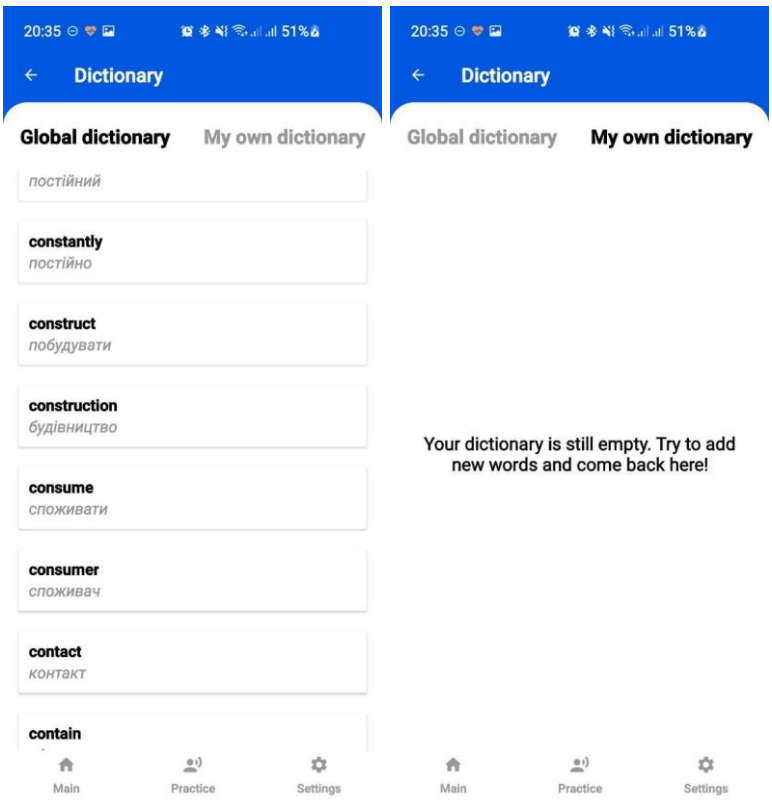

Рисунок 2.36 – DictionaryActivity

7) Активність «Додати нове слово». Кнопка «Add new word» відкриває подану форму. Дана діяльність має 2 обов’язкових для заповнення TextInputs, до яких потрібно записати назву слова на його переклад у відповідні поля. Лише при умові, що текст інпути не є пустими, кнопка «Save new word» стає активною (рис. 2.37). Необов’язковим до використання, проте не останнім за корисністю, є можливість додавання зображення до поданого слова. Поданий метод є корисним для створення візуальних асоціацій під час вивчення слова.

Після натискання на кнопку «Add image», з’являється модальне діалогове вікно, на якому є можливість вибрати яким методом можна обрати зображення (рис. 2.38).

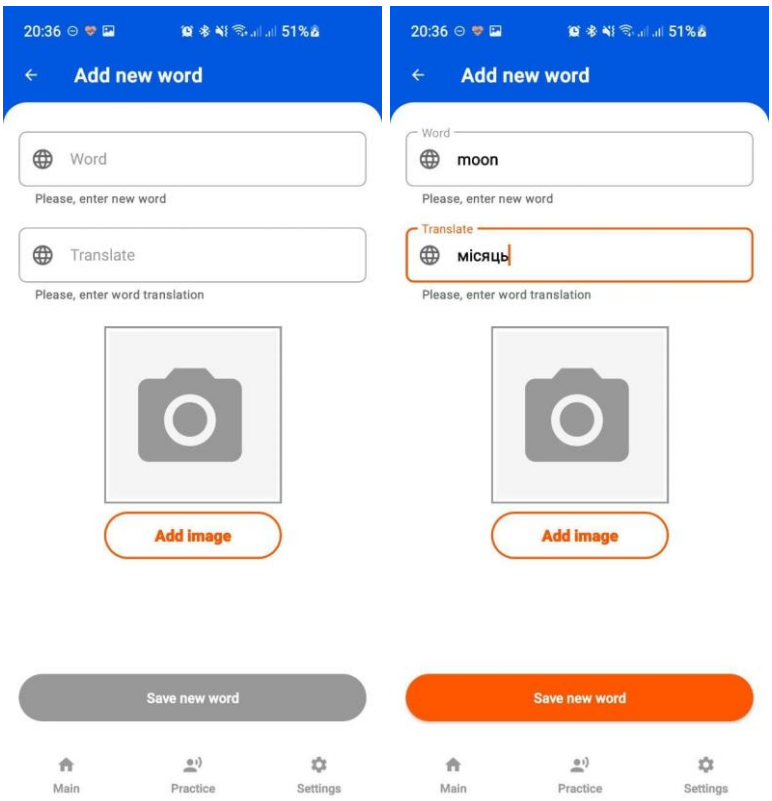

Рисунок 2.37 – AddNewWordActivity

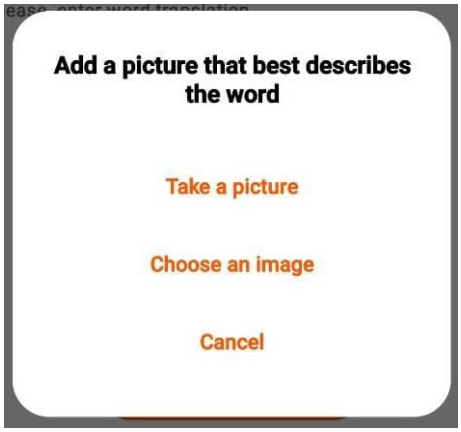

Рисунок 2.38 – Add image

Спочатку розглянемо метод «Choose an image», який передбачає вибір зображення з Галереї смартфону (рис. 2.39).

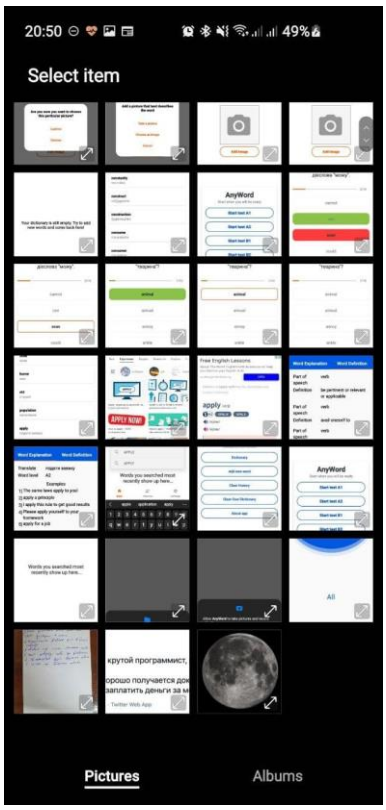

Рисунок 2.39 – Вибір зображення з Галереї

Тепер розглянемо метод «Take a picture». Він передбачає відкриття камери телефону, що дає змогу робити швидкі фотокартки у реальному часі (рис. 2.40).

Таким чином, повністю заповнена форма «Add new word» має вигляд, зображений на рисунку 2.41. Після натиснення на кнопку «Save new word» з’являється BottomModalDialog, котрий сповіщає про успішне виконання операції.

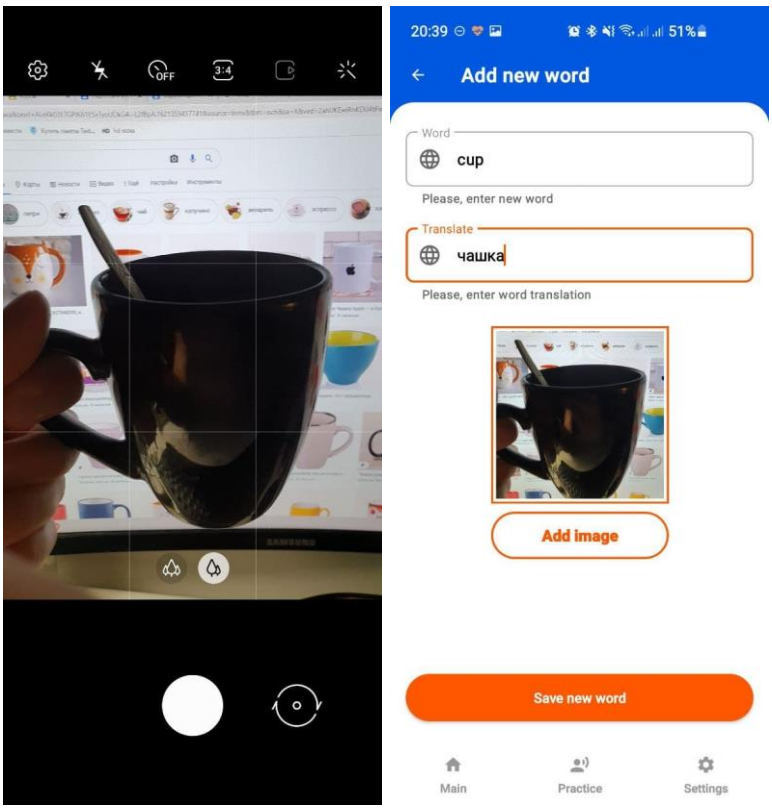

Рисунок 2.40– Додавання зображення за допомогою камери

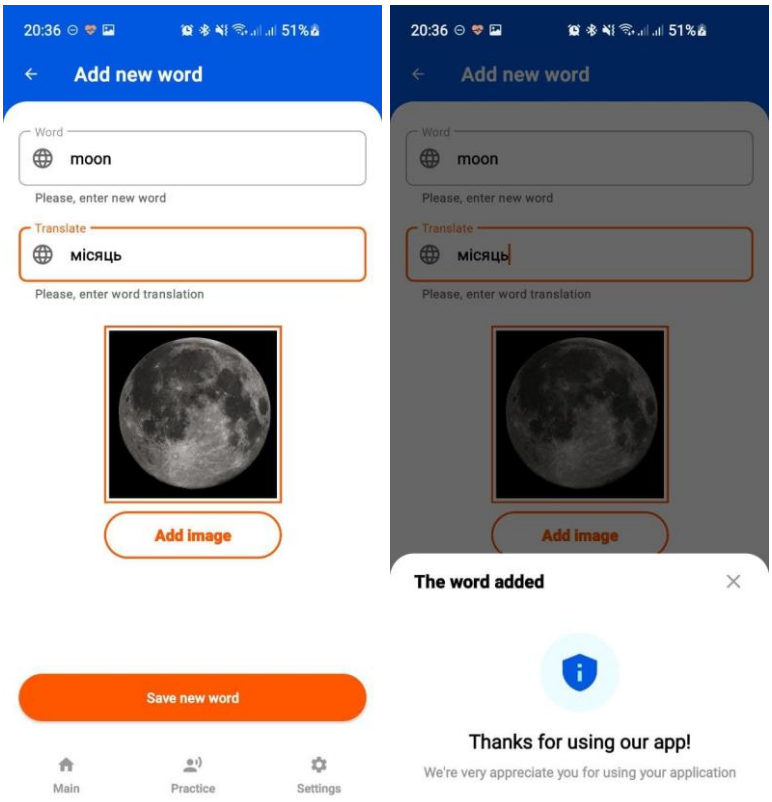

Рисунок 2.41 – Додавання нового слова до власного словника

Таким чином, фрагмент «My own dictionary» вже має заповнені картки нових слів, що зображено на рисунку 2.42.

Рисунок 2.42 – Заповнений «Власний словник»

Тепер постає питання: «Чи відрізняються форми визначення слів з глобального та власного словників?». І відповідь на це питання – «Так».

Існує 2 незначні відмінності, котрі доповнюють функціонал власного словника. Перш за все, при додаванні зображення під час заповнення форми «Add new word», форма «WordMeaning» демонструє подану картинку (рис. 2.43).

Другою відмінністю є можливість видалення слова зі словника. Кнопка «Delete word» з'являється після натискання на кнопку «Додатково», яка розташована у правому верхньому куті (рис. 2.44).

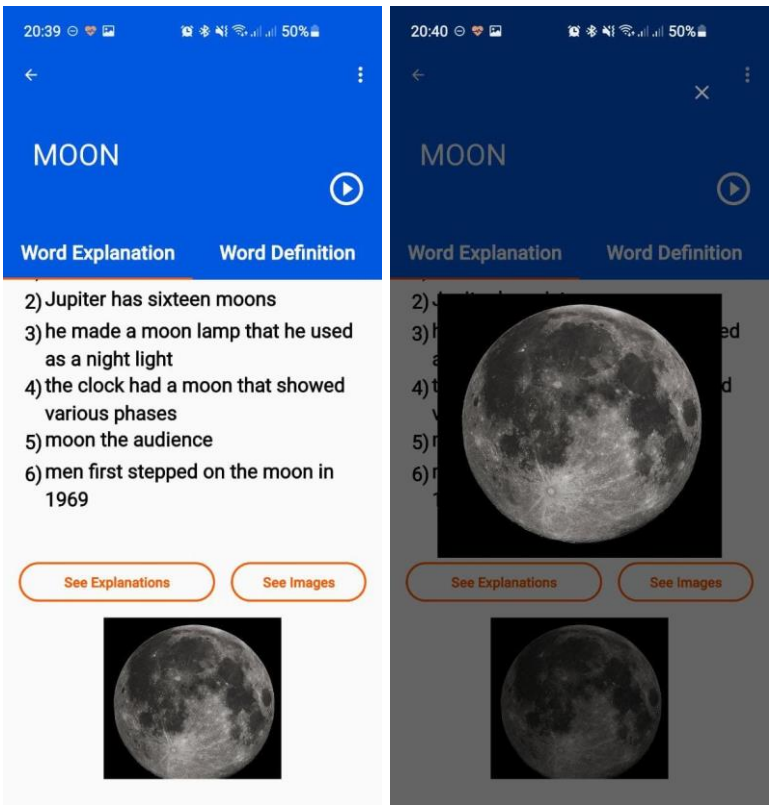

Рисунок 2.43 – Відображення картинки на формі WordMeaning

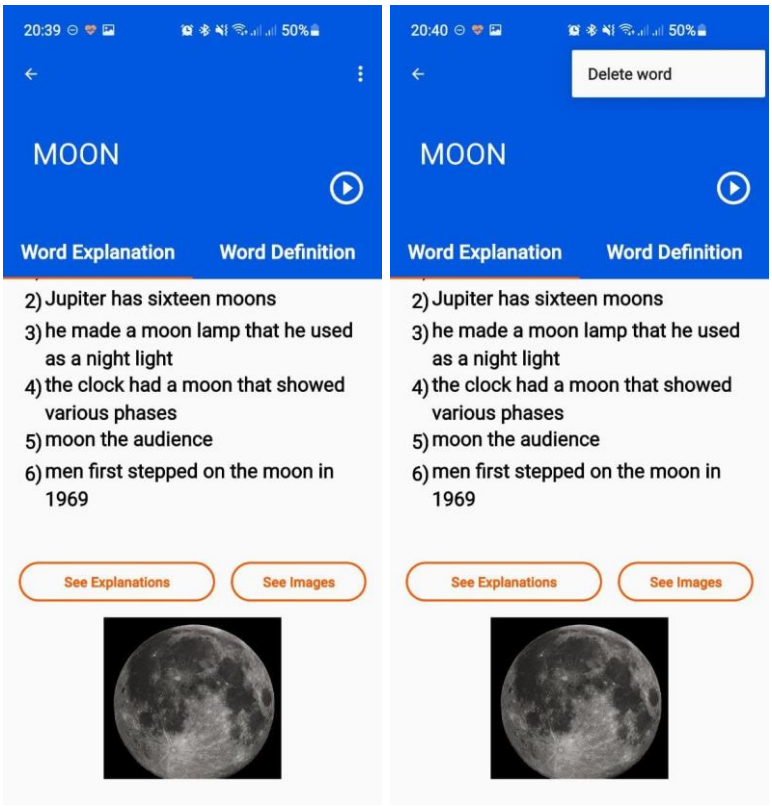

Рисунок 2.44 – Видалення слова з власного словника

Також, існує можливість видалити слово прямо зі списку на формі «My own dictionary». Необхідно лише зробити довге натискання на обране вами слово, і з’явиться ModalDialog, на якому потрібно підтвердити процес видалення слова. Отже, вигляд поданої процедури і результати видалення слів зображені на рисунку 2.45.

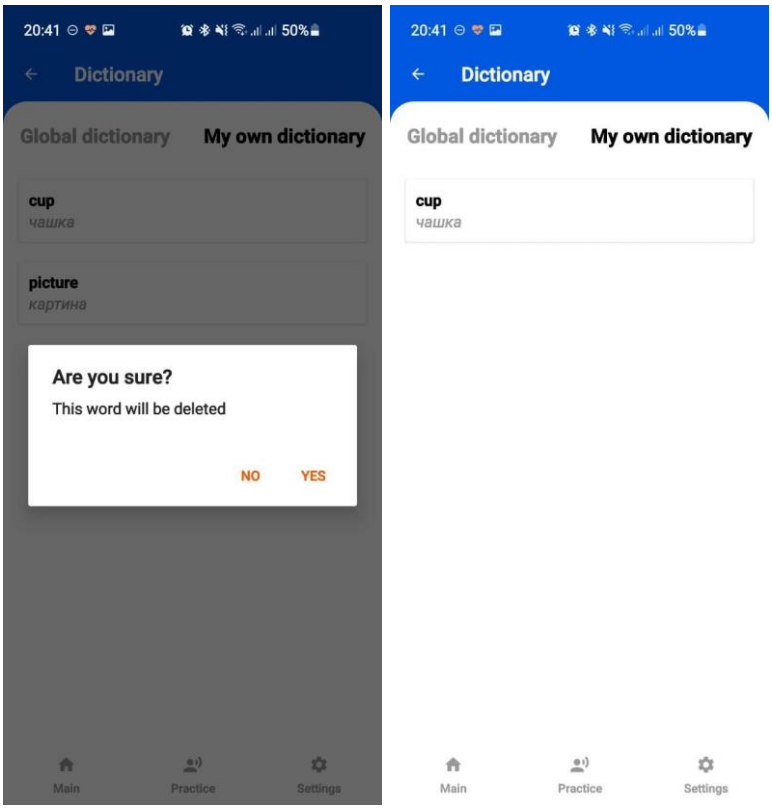

Рисунок 2.45 – Результат видалення слів з власного словника

Таким чином, були наведені скриншоти з прототипу мобільного додатку AnyWord та була надана інструкція щодо використання поданою прикладною програмою.

2.4. Очікуваний ефект від використання мобільного додатку

Мобільний додаток AnyWord має чітко виражений соціальний ефект у вигляді підвищення рівня знань з англійської мови для користувачів. Додатково розроблений прототип має економічний потенціал.Основним

завданням на першому етапі життєвого циклу прикладної програми є завоювання клієнтів. Це необхідно для того, щоб зайняти нішу у сегменті ринку, спрямованого на саморозвиток. У майбутньому планується поширювати поданий мобільний додаток не лише на учасників освітньої діяльності, проте й на кожного бажаючого. Це можна зробити шляхом монетизації додатку.

Монетизація – це процес перетворення предмета, що не приносить доходу, у грошові кошти. У багатьох випадках монетизація розглядає нові методи отримання доходу з нових джерел; наприклад, шляхом включення доходів від реклами у відеокліпи соціальних мереж для оплати творців контенту. Іноді монетизація відбувається за рахунок приватизації, завдяки якій раніше вільний або державний актив перетворюється на центр прибутку - наприклад, дорога загального користування перетворюється на приватну платну дорогу [17, 23].

По-перше, мобільний додаток AnyWord не матиме оплати за завантаження у PlayMarket. Так, як першим і основним завданням є залучення клієнтів, поданий крок може лише відштовхнути їх.

По-друге, у мабутньому існує можливість застосування моделі Freemium. Це означає, що користувачі матимуть можливість безкоштовно завантажити і користуватися мобільним додатком, проте матиме обмежену функціональність. Основною перевагою поданої моделі є те, що охоплення аудиторії не буде зменшуватися, а, навпаки, лише поширюватися і, тим самим, збульшувати кількість завантажень на PlayMarket. Також користувач матиме можливість вибору: чи залишатися на безкоштовній версії, або заплатити і мати всі функції. Наявність поданого вибору є гарним психологічним відтінком, котрий також може привабити потенціальних користувачів [13].

По-третє, існує можливість впровадження реклами до мобільного додатку. Поданий варіант є одним із найпопулярніших способів монетизації на ринку мобільних додатків. У цьому випадку, прикладна програма буде повністю безкоштовною, а можливість відключення реклами буде платною.

По-четверте, у мобільному додатку є можливість здійснювати продажу власних послуг (наприклад, заняття з репетитором) [11].

По-п'яте, існує варіант колоборації з іншими партнерами, тим самим можливо здійснити партнерство з альтернативними мобільними додатками.

По-шосте, у майбутньому планується розробка поданої прикладної програми для її поширення серед користувачів IOS. Поданий крок планується зробити тоді, коли серед Android-користувачів подана програма стане популярною, та відбудуться перші етапи монетизації.

Таким чином, існує багато способів монетизувати мобільний додаток. Головне – на перших кроках існування прикладної програми – захопити власну нішу й привабити клієнтів, що, у свою чергу, дасть можливість провести процес монетизації та отримати джерело пасивного доходу.

ВИСНОВКИ

В ході написання кваліфікаційної роботи бакалавра було досліджено відносини що виникають між студентами та викладачами в процесі надання освітніх послуг в Україні.

Під час дослідження інструментів інтелектуалізації та оптимізації освітньої діяльності, були виявлені наявні альтернативи мобільних додатків для вивчення англійської мови, розглянуті їх переваги та недоліки.

Була досягнута основна мета роботи, а саме був розроблений прототип мобільного додатку для вивчення англійської мови.

Також були виконані всі завдання кваліфікаційної роботи, а саме:

– було охарактеризовано освітню діяльність України;

– було проведено аналіз сучасного стану інтелектуалізації освітньої діяльності;

– були сформовані вимоги до системи та розглянуті основні інструменти їх виконання;

– було змоделювано та спроектовано архітектуру мобільного додатку;

– було реалізоване інформаційне забезпечення прикладної програми;

– був розроблений контрольний приклад та інструкцію щодо використання мобільного додатку;

– було розглянуто можливість створення монетизації мобільного додатку для створення пасивного джерела прибутку.

Таким чином, за рахунок виконання основної мети кваліфікаційної роботи, можна досягти високого рівня у покращенні рівня володіння англійською мовою серед учасників освітньої діяльності України.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Архітектура та проектування програмного забезпечення. URL: https://learn.ztu.edu.ua/mod/book/view.php?id=278&chapterid=71 (дата звернення: 09.05.2021)

2. Кісь Н. Початкова англійська: 100 базових слів із перекладом. Green Forest. URL: https://greenforest.com.ua/ua/journal/read/nachalnyj-anglijskij-100-bazovyh-slov-s-perevodom (дата звернення: 05.05.2021)

3. Кісь С., Кісь Г. Явище «інтелектуалізації» у діяльності соціально-економічних систем. Галицький економічний вісник. Том 48. № 1. С. 55-62. URL: https://galicianvisnyk.tntu.edu.ua/pdf/48/132.pdf (дата звернення: 05.05.2021)

4. Лопатьєв А.О. Моделювання як методологія пізнання. Львівський державний університет фізичної культури, Центр математичного моделювання ІППММ ім. Я. С. Підстригача НАН України. 2007. № 8. URL: https://core.ac.uk/download/pdf/304295518.pdf (дата звернення: 07.05.2021)

5. Про внесення змін до постанови Кабінету Міністрів України від 30 грудня 2015 р. № 1187: Постанова Кабінету міністрів України від 10 травня 2018 р. № 347. URL: https://zakon.rada.gov.ua/laws/show/347-2018-%D0%BF#n13 (дата звернення: 06.05.2021)

6. Про освіту: Закон України від 05.09.2017 N 2145-VIII. Освітня діяльність. URL: https://ips.ligazakon.net/document/TM057072 (дата звернення: 06.05.2021)

7. Про освіту: Закон України від 05.09.2017 N 2145-VIII. Суб'єкт освітньої діяльності. URL: http://search.ligazakon.ua/l_doc2.nsf/link1/TM057081.html (дата звернення: 06.05.2021)

8. Пройдаков Е. М., Теплицький Л. А. Англо-Український тлумачний словник з обчислювальної техніки, Інтернету і програмування. СофтПрес. С. 552. (дата звернення: 07.05.2021)

9. Рівень Upper-Intermediate – рівень володіння англійською вище середнього B2. EnglishDom. 2019. URL: https://www.englishdom.com/blog/riven-upper-intermediate-riven-volodinnya-anglijskoyu-vishhe-serednogo-b2/ (дата звернення: 07.05.2021)

10. СЛОВНИК – тлумачний словник української мови, орфографічний словник онлайн. URL: https://slovnyk.ua/ (дата звернення: 05.05.2021)

11. Способи монетизації мобільних додатків. Westelecom. 2021. URL: https://westelecom.ua/ua/blog/288_sposoby-monetizacii-mobilnyh-prilozenij.html (дата звернення: 15.05.2021)

12. Туровський Д. Переваги вивчення англійської мови для дорослих. EnglishPrime. 2021. URL: https://englishprime.ua/uk/preimushchestva-izucheniya-anglijskogo-yazyka-dlya-vzroslyh/ (дата звернення: 08.05.2021)

13. Федоричак В. Монетизація мобільних додатків: 8 способів заробити на своєму продукті. Senior. 2021. URL: https://senior.ua/articles/monetizacya-moblnih-dodatkv-8-sposobv-zarobiti-na-svomu-produkt (дата звернення: 15.05.2021)

14. Хохлова Є. Якими бувають рівні володіння англійською? Magazine. URL: https://enguide.ua/ua/magazine/kakie-byvayut-urovni-vladeniya-angliyskim-yazykom (дата звернення: 07.05.2021)

15. Чому рівень володіння англійською мовою в українців нижчий середньосвітового – пояснює дослідниця. Українське радіо. 11.11.2019р. URL: http://www.ukr.radio/news.html?newsID=91542 (дата звернення: 05.05.2021)

16. Що таке алгоритм? МійКлас. URL: https://miyklas.com.ua/p/informatica/5-klas/algoritmi-ta-programi-python-51129/algoritm-ta-iogo-vlastivosti-48217/re-11a12614-aa85-4655-b188-695b028496ad (дата звернення: 10.05.2021)

17. Як просувати й монетизувати мобільні додатки у 2020 році. UAMaster. 2020. URL: https://blog.uamaster.com/how-to-monetirize-apps-2020/ (дата звернення: 14.05.2021)

18. Белокаменцев И. Синхронность и асинхронность процессов. 23 мая 2019. URL: https://habr.com/ru/post/453192/ (дата звернення: 08.05.2021)

19. Что такое Android Studio: Web-Proger. URL: http://web.spt42.ru/index.php/chto-takoe-android-studio (дата звернення: 08.05.2021)

20. Что такое API? Простое объяснение для начинающих. Dev. URL: https://dev.by/news/chto-takoe-api-prostym-yazykom (дата звернення: 10.05.2021)

21. Что такое CEFR и для чего оно нужно? GRADE Education Centre. URL: https://grade.ua/news/chto-takoe-cefr-i-dlya-chego-ono-nuzhno/ (дата звернення: 05.05.2021)

22. A modern programming language that makes developers happier. Kotlin Programming Language. URL: https://kotlinlang.org/ (дата звернення: 07.05.2021)

23. Akhilesh Ganti. Monetize. Investipedia. 2021. URL: https://www.investopedia.com/terms/m/monetize.asp (дата звернення: 15.05.2021)

24. Diogo Ferreira. OkHttp: Android Tutorial. During one of my projects, I looked. Codavel Tech Blog. 2019. URL: https://medium.com/codavel-blog/how-to-integrate-okhttp-on-an-android-project-fe5052cc4fb3 (дата звернення: 09.05.2021)

25. Duolingo – The world's best way to learn a language. Duolingo. URL: https://en.duolingo.com/ (дата звернення: 06.05.2021)

26. Easy ten – any language with ten words a day. Easy ten. URL: http://www.easyten.ru/en (дата звернення: 06.05.2021)

27. EF Education First – Global SIte (English). EF. URL: https://www.ef.com/wwen/ (дата звернення: 05.05.2021)

28. English language levels (CEFR). TrackTest. URL: https://tracktest.eu/english-levels-cefr/ (дата звернення: 07.05.2021)

29. Fielding Roy. Architectural Styles and the Design of Network-based Software Architectures. Каліфорнійський університет в Ірвайні. 2000 (дата звернення: 07.05.2021)

30. Lingualeo – foreign languages online. Lingualeo. URL: https://lingualeo.com/en (дата звернення: 06.05.2021)

31. Madi Connors. What is Xcode and why do I need it? Quora. 2016. URL: https://www.quora.com/What-is-Xcode-and-why-do-I-need-it (дата звернення: 06.05.2021)

32. Martin Heller. What is Kotlin? The Java alternative explained. InfoWorld. 2020. URL: https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html (дата звернення: 07.05.2021)

33. Meet Android Studio. Android Developers. Android Studio. URL: https://developer.android.com/studio/intro#find_sample_code (дата звернення: 07.05.2021)

34. Merriam-Webster. Definition of Optimization. Merriam-Webster. URL: https://www.merriam-webster.com/dictionary/optimization (дата звернення: 06.05.2021)

35. Modeling. Cambridge English Dictionary. URL: https://dictionary.cambridge.org/dictionary/english/modeling (дата звернення: 09.05.2021)

36. OkHttp. GitHub. URL: https://square.github.io/okhttp/ (дата звернення: 10.05.2021)

37. React Native. Learn once, write anywhere. React Native. URL: https://reactnative.dev/ (дата звернення: 06.05.2021)

38. Ricardo Costa. Asynchronous vs. Synchronous Programming: When to Use What. Outsystems. 2021. URL: https://www.outsystems.com/blog/posts/asynchronous-vs-synchronous-programming/ (дата звернення: 10.05.2021)

39. Thinkwik. React Native: What is it? and, Why is it used? Medium. 2018. URL: https://medium.com/@thinkwik/react-native-what-is-it-and-why-is-it-used-b132c3581df (дата звернення: 06.05.2021)

40. What is an API? (Application Programming Interface). MuleSoft. URL: https://www.mulesoft.com/resources/api/what-is-an-api (дата звернення: 11.05.2021)

41. What You Should Know About The App Design Process. Smashine Magazine. 2016. URL: https://www.smashingmagazine.com/2016/11/what-everyone-should-know-about-the-process-behind-app-design/ (дата звернення: 10.05.2021)

42. Xcode 12 – Apple Developer. Apple. URL: https://developer.apple.com/xcode/ (дата звернення: 06.05.2021)

ДОДАТКИ

Додаток А

SUMMURY

Revenko A.V. Intellectualization of channels of optimization of educational activity in Ukraine. Qualifying work of the bachelor. Sumy State University, Sumy, 2021.

The paper examines the process of intellectualization among participants in educational activities in Ukraine. An analysis of the current state of the mobile applications market was conducted. The basic requirements for the system were given and the choice of development technologies was made. The architecture of the mobile application was designed and the prototype of the application program was implemented.

Keywords: intellectualization, automation, mobile application, application program, educational activity, prototype, English language.

АНОТАЦІЯ

Ревенко А. В. Інтелектуалізація каналів оптимізації освітньої діяльності в Україні. Кваліфікаційна робота бакалавра. Сумський державний університет, Суми, 2021 р.

У роботі досліджено процес інтелектуалізації серед учасників освітньої діяльності в Україні. Було проведено аналіз сучасного стану ринку мобільних додатків. Були наведені основні вимоги до системи та був здійснений вибір технологій розробки. Було спроектовано архітектуру мобільного додатку та реалізовано прототип прикладної програми.

Ключові слова: інтелектуалізація, автоматизація, мобільний додаток, прикладна програма, освітня діяльність, прототип, англійська мова.

Додаток Б

Лістинг Б.1 – Код форми LoadingActivity

```
var databaseHelper: DatabaseHelper? = null
var databaseOpened = false
class LoadingActivity : AppCompatActivity() {
    var loadingLogosText: IntArray = intArrayOf(R.mipmap.loading_text_all,
R.mipmap.loading_text_words, R.mipmap.loading_text_that, R.mipmap.loading_text_you,
R.mipmap.loading_text_need, R.mipmap.loading_text_in, R.mipmap.loading_text_any_word)
    var loadingProgressIndicators: IntArray =
intArrayOf(R.mipmap.loading_progress_indicator_1,R.mipmap.loading_progress_indicator_
2,R.mipmap.loading_progress_indicator_3,R.mipmap.loading_progress_indicator_4,R.mipma
p.loading_progress_indicator_5,R.mipmap.loading_progress_indicator_6,R.mipmap.loading
_progress_indicator_7)
    var loadingLogoText: ImageView? = null
    var loadingProgressIndicator: ImageView? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_loading)
        loadingLogoText = findViewById(R.id.loadingLogoText)
        loadingProgressIndicator = findViewById(R.id.loadingProgressIndicator)
        databaseHelper = DatabaseHelper(this)
        if (databaseHelper?.checkDatabase() == true) {
            openDatabase()
        } else {
            val loadDatabaseAsync = LoadDatabaseAsync(this)
            loadDatabaseAsync.execute() }
        startTimer()
    }
    @SuppressLint("LongLogTag")
    private fun openDatabase() {
        try {
            databaseHelper?.openDatabase()
            databaseOpened = true
        } catch (e: SQLException) {
            Log.w("Form-LoadingActivity-openDatabase-error", "$e") }}
    private fun startTimer() {
        var counter = 0
        val timer = Timer()
        timer.schedule(object : TimerTask() {
            override fun run() {
                runOnUiThread {
                    loadingLogoText?.setImageResource(loadingLogosText[counter])
loadingProgressIndicator?.setImageResource(loadingProgressIndicators[counter])
                    counter++
                    if (counter == loadingLogosText.size) { timer.cancel()
                        Handler().postDelayed({goToMainActivity()}, 1000)
}}}}, 0, 1000)}
    fun goToMainActivity() {startActivity(Intent(this, MainActivity::class.java))}
```

Лістинг Б.2 – Код форми MainActivity

```
var wordMeaning = String()
var wordExamples = String()
var wordEnglish = String()
var mainManager: FragmentManager? = null
@SuppressLint(«StaticFieldLeak»)
```

```
var mainContext: Context? = null
class MainActivity : AppCompatActivity() {
    var mainSearchView: SearchView? = null
    var simpleCursorAdapter: SimpleCursorAdapter? = null
    var mainRecyclerViewHistory: RecyclerView? = null
    var mainRelativeLayout: RelativeLayout? = null
    var cursorHistory: Cursor? = null
    var intentMeaning = Intent()
    @SuppressLint(«LongLogTag»)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        mainManager = supportFragmentManager
        mainContext = applicationContext
        setSupportActionBar(findViewById(R.id.mainToolbar))
        whatDictionary = «Global»
        if (checkSelfPermission(Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED ||
checkSelfPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE) !=
PackageManager.PERMISSION_GRANTED) {
            requestPermissions(
                arrayOf(Manifest.permission.CAMERA,
Manifest.permission.WRITE_EXTERNAL_STORAGE ), 1)}
        mainSearchView = findViewById(R.id.mainSearchView)
        mainSearchView?.setOnClickListener { mainSearchView?.isIconified = false }
        val from = arrayOf(«en_word»)
        val to = intArrayOf(R.id.suggestionRowText)
        simpleCursorAdapter = object : SimpleCursorAdapter( this@MainActivity,
R.layout.suggestion_row, null, from, to, 0) {
         override fun changeCursor(cursor: Cursor) { super.changeCursor(cursor)} }
        mainSearchView?.suggestionsAdapter = simpleCursorAdapter
        mainSearchView?.setOnSuggestionListener(object :
SearchView.OnSuggestionListener {
            override fun onSuggestionClick(position: Int): Boolean {
                val suggestionsAdapter = mainSearchView?.suggestionsAdapter
                val cursor = suggestionsAdapter?.cursor
                cursor?.moveToPosition(position)
                wordEnglish =
cursor?.getString(cursor.getColumnIndex(«en_word»)).toString()
                mainSearchView?.setQuery(wordEnglish, false)
                mainSearchView?.clearFocus()
                mainSearchView?.isFocusable = false
                intentMeaning = Intent(this@MainActivity,
WordMeaningActivity::class.java)
                ApiForMainActivity(intentMeaning,
mainManager).getWordMeaning(«definitions»)
                val bundle = Bundle()
                bundle.putString(«en_word», wordEnglish)
                intentMeaning.putExtras(bundle)
                return true }
            override fun onSuggestionSelect(position: Int): Boolean {
                return true }
        })
        mainSearchView?.setOnQueryTextListener(object :
SearchView.OnQueryTextListener {
            override fun onQueryTextSubmit(query: String): Boolean {
                wordEnglish = mainSearchView?.query.toString()
                val cursor: Cursor? =
databaseHelper?.getMeaningFromGlobal(wordEnglish)
                if (cursor?.count == 0) {
                    mainSearchView?.setQuery(«», false)
                    val builder = AlertDialog.Builder(this@MainActivity,
```

```
R.style.MyDialogTheme)
                    builder.setTitle(«Word Not Found»)
                    builder.setMessage(«Please search again»)
                    val positiveText = getString(android.R.string.ok)
                    builder.setPositiveButton(
                        positiveText
                    ) { dialog, which -> }
                    val negativeText = getString(android.R.string.cancel)
                    builder.setNegativeButton(
                        negativeText
                    ) { dialog, which -> mainSearchView?.clearFocus() }
                    val dialog = builder.create()
                    dialog.show()
                } else {
                    mainSearchView?.clearFocus()
                    mainSearchView?.isFocusable = false
                    intentMeaning = Intent(this@MainActivity,
WordMeaningActivity::class.java)
                    val bundle = Bundle()
                    bundle.putString(«en_word», wordEnglish)
                    intentMeaning.putExtras(bundle)
                    ApiForMainActivity(intentMeaning,
mainManager).getWordMeaning(«definitions») }
                return false }
            override fun onQueryTextChange(s: String): Boolean {
                mainSearchView?.setIconifiedByDefault(false)
                val cursorSuggestion: Cursor? =
databaseHelper?.getSuggestionsFromGlobal(s)
                simpleCursorAdapter?.changeCursor(cursorSuggestion)
                return false }
        })
        mainRelativeLayout = findViewById<RelativeLayout>(R.id.mainRelativeLayout)
        mainRecyclerViewHistory =
findViewById<RecyclerView>(R.id.mainRecyclerViewHistory)
        val layoutManager = LinearLayoutManager(this)
        mainRecyclerViewHistory?.layoutManager = layoutManager
        fetchHistory() }
    private fun fetchHistory() {
        val historyList = ArrayList<History>()
        val historyAdapter = HistoryAdapter(this, historyList, intentMeaning)
        mainRecyclerViewHistory?.adapter = historyAdapter
        var history: History
        if (databaseOpened) {
            cursorHistory = databaseHelper?.getHistory
            if (cursorHistory?.moveToFirst() == true) {
                do {
                    history = History(
cursorHistory!!.getString(cursorHistory!!.getColumnIndex(«word»)),
cursorHistory!!.getString(cursorHistory!!.getColumnIndex(«word_translate»))
                    )
                    historyList.add(history)
                } while (cursorHistory!!.moveToNext())
            } else {
                Log.w(«Form-MainActivity-fetchHistory», «cursorHistory == false»)
            }
            historyAdapter.notifyDataSetChanged()
        } else {
            Log.w(«Form-MainActivity-fetchHistory», «databaseOpened == false»)
        }
        if (historyAdapter.itemCount == 0) {
            mainRelativeLayout?.visibility = View.VISIBLE
        } else {
```

```
            mainRelativeLayout?.visibility = View.GONE }
    }
    fun goToPracticeActivity(view: View) {
        startActivity(Intent(this, PracticeActivity::class.java))
    }
    fun goToSettingsActivity(view: View) {
        startActivity(Intent(this, SettingsActivity::class.java))
    }
    override fun onBackPressed() {}
    @SuppressLint(«LongLogTag»)
    override fun onResume() { super.onResume()}
    @SuppressLint(«LongLogTag»)
    override fun onPause() { super.onPause() }
    @SuppressLint(«LongLogTag»)
    override fun onStop() { super.onStop() }
    @SuppressLint(«LongLogTag»)
    override fun onRestart() {
        super.onRestart()
        fetchHistory()
    }
    @SuppressLint(«LongLogTag»)
    override fun onDestroy() { super.onDestroy() }
}
```

Лістинг Б.3 – Код форми PracticeActivity

```
var levelTest = «A1»
class PracticeActivity : AppCompatActivity() {
    @SuppressLint(«LongLogTag»)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_practice)
        setSupportActionBar(findViewById<Toolbar>(R.id.practiceToolbar))
        findViewById<Button>(R.id.testForA1).setOnClickListener {
            levelTest = «A1»
            startActivity(Intent(this, QuizQuestionsActivity::class.java))
        }
        findViewById<Button>(R.id.testForA2).setOnClickListener {
            levelTest = «A2»
            startActivity(Intent(this, QuizQuestionsActivity::class.java))
        }
        findViewById<Button>(R.id.testForB1).setOnClickListener {
            levelTest = «B1»
            startActivity(Intent(this, QuizQuestionsActivity::class.java))
        }
        findViewById<Button>(R.id.testForB2).setOnClickListener {
            levelTest = «B2»
            startActivity(Intent(this, QuizQuestionsActivity::class.java))
        }
    }
    fun goToSettingsActivity(view: View) {
        startActivity(Intent(this, SettingsActivity::class.java))
    }
    fun goToMainActivity(view: View) {
        startActivity(Intent(this, MainActivity::class.java))
    }
}
```

Лістинг Б.4 – Код форми SettingsActivity

```
class SettingsActivity : AppCompatActivity() {
    var myDbHelper: DatabaseHelper? = null
    var typeOfClean = String()
    @SuppressLint("SetTextI18n", "LongLogTag")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_settings)
        if (databaseOpened) {
            val cursorDictionary = databaseHelper?.getDictionaryOwn
            if (cursorDictionary?.moveToFirst() == true) {
                do {
                    Log.w("Form-cursorDictionary",
cursorDictionary.getString(cursorDictionary.getColumnIndex("en_word")).toString())
                } while (cursorDictionary.moveToNext())
}}
        setSupportActionBar(findViewById(R.id.settingsToolbar))
        val appVersion = packageManager.getPackageInfo(packageName, 0).versionName
        val settingsVersionText = findViewById<TextView>(R.id.settingsVersionText)
        settingsVersionText.text = "Version $appVersion"
        val clearHistory = findViewById<Button>(R.id.settingsClearHistory)
        clearHistory.setOnClickListener {
            myDbHelper = DatabaseHelper(this@SettingsActivity)
            try {
                myDbHelper?.openDatabase()
            } catch (e: SQLException) {
                e.printStackTrace()
            }
            typeOfClean = "history"
            showAlertDialog()
        }
        val clearOwnDictionary =
findViewById<Button>(R.id.settingsClearOwnDictionary)
        clearOwnDictionary.setOnClickListener {
            myDbHelper = DatabaseHelper(this@SettingsActivity)
            try { myDbHelper?.openDatabase()
            } catch (e: SQLException) { e.printStackTrace() }
            typeOfClean = "dictionary"
            showAlertDialog()
}}
    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        if (item.itemId == android.R.id.home) onBackPressed()
        return super.onOptionsItemSelected(item)
    }
    private fun showAlertDialog() {
        val builder = AlertDialog.Builder(this@SettingsActivity,
R.style.MyDialogTheme)
        builder.setTitle("Are you sure?")
        if (typeOfClean=="history") {
            builder.setMessage("All the history will be deleted")
        } else {
            builder.setMessage("Your own dictionary will be deleted")
        }
        val positiveText = "Yes"
        builder.setPositiveButton(
            positiveText
        ) { dialog, which ->
            if (typeOfClean=="history") {
                myDbHelper?.deleteHistory()
                startActivity(Intent(this, MainActivity::class.java))
            } else {
```

```
                myDbHelper?.deleteNewWords()
}}
        val negativeText = "No"
        builder.setNegativeButton(
            negativeText
        ) { dialog, which -> }
        val dialog = builder.create()
        dialog.show()
    }
    fun goToPracticeActivity(view: View) {
        startActivity(Intent(this, PracticeActivity::class.java))
    }
    fun goToMainActivity(view: View) {
        startActivity(Intent(this, MainActivity::class.java))
    }
    fun goToAboutAppActivity(view: View) {
        startActivity(Intent(this, AboutAppActivity::class.java))
    }
    fun goToDictionaryActivity(view: View) {
        startActivity(Intent(this, DictionaryActivity::class.java))
    }
    fun goToAddWordActivity(view: View) {
        startActivity(Intent(this, AddWordActivity::class.java))
    }
}
```

Додаток В

Лістинг В.1 – Код форми ApiForMainActivity

```
class ApiForMainActivity(var intent: Intent, var fragmentManager: FragmentManager?) {
    private val client = OkHttpClient()
    val dialogProgressIndicator = DialogProgressIndicator()
    fun getWordMeaning(end: String) {
        var apiEnd = end
        if (apiEnd != "examples") dialogProgressIndicator.show(fragmentManager!!,
"customDialog")
        val request = Request.Builder()
            .url("https://wordsapiv1.p.rapidapi.com/words/$wordEnglish/$apiEnd")
            .get()
            .addHeader("x-rapidapi-key",
"ffd451c28fmsh27f2da64dbe1a0fp1b069bjsnb8f2bcbd8ec3")
            .addHeader("x-rapidapi-host", "wordsapiv1.p.rapidapi.com")
            .build()
        client.newCall(request).enqueue(object : Callback {
            @SuppressLint("CommitPrefEdits", "LongLogTag")
            override fun onResponse(call: Call, response: Response) {
                response.use {
                    if (!response.isSuccessful) {
                        try {
                            val json = JSONObject(response.body()!!.string())
                            dialogProgressIndicator.dismiss()
                        } catch (e: JSONException) {
                         Log.w("Form-ApiForMainActivity-catch ", e.toString()) }
                    } else {
                        try {
                            val json = JSONObject(response.body()!!.string())
                            if (apiEnd != "examples") {
                                val responseArray = json.getJSONArray("definitions")
                                wordMeaning = responseArray.toString()
                                apiEnd = "examples"
                                getWordMeaning(apiEnd)
                            } else {
                                val responseArray = json.getJSONArray("examples")
                                wordExamples = responseArray.toString()
                                dialogProgressIndicator.dismiss()
                                intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK
                                startActivity(mainContext!!, intent, intent.extras) }
                        } catch (e: JSONException) {
                            Log.w("Form-ApiForMainActivity-catch", e.toString())
}}}}
            @SuppressLint("LongLogTag")
            override fun onFailure(call: Call, e: IOException) {
                Log.w("Form-ApiForMainActivity-onFailure", e.toString())
}})}}
```

Лістинг В.2 – Код форми WordMeaningActivity

```
var wordForSearch = String()
var wordMeaningContext: Context? = null
class WordMeaningActivity : AppCompatActivity() {
    var viewPager: ViewPager? = null
    var enWord = ""
    var wordTranslate = ""
    var wordLevel = ""
```

```
    var wordImage = ""
    private var textToSpeech: TextToSpeech? = null
    private val myDbHelper = DatabaseHelper(this)
    @SuppressLint("LongLogTag")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_word_meaning)
        wordMeaningContext = applicationContext
        val bundle = intent.extras
        enWord = bundle?.getString("en_word").toString()
        wordForSearch = enWord.toLowerCase()
        try { myDbHelper.openDatabase() } catch (sqle: SQLException) { throw sqle }
        val wordMeaningToolbar = findViewById<Toolbar>(R.id.wordMeaningToolbar)
        setSupportActionBar(wordMeaningToolbar)
        wordMeaningToolbar.setNavigationIcon(R.drawable.ic_action_bar_back)
        supportActionBar?.title = enWord
        val cursor: Cursor? = if (whatDictionary != "Global") {
            myDbHelper.getMeaningFromOwn(enWord)
        } else {
            myDbHelper.getMeaningFromGlobal(enWord) }
        if (cursor?.moveToFirst() == true) {
            wordTranslate =
cursor.getString(cursor.getColumnIndex("word_translate")).toString()
            if (whatDictionary == "Global") {
                wordLevel =
cursor.getString(cursor.getColumnIndex("word_level")).toString()
            } else {
              wordImage =
cursor.getString(cursor.getColumnIndex("word_image")).toString() }}
        if (whatDictionary == "Global") myDbHelper.insertIntoHistory(enWord)
        val wordMeaningBtnSpeak = findViewById<ImageButton>(R.id.wordMeaningBtnSpeak)
        wordMeaningBtnSpeak.setOnClickListener {
            textToSpeech = TextToSpeech(
                this@WordMeaningActivity
            ) { status -> // TODO Auto-generated method stub
                if (status == TextToSpeech.SUCCESS) {
                    val result: Int? = textToSpeech?.setLanguage(Locale.getDefault())
                    if (result == TextToSpeech.LANG_MISSING_DATA || result ==
TextToSpeech.LANG_NOT_SUPPORTED) {
                    } else {
                        textToSpeech?.speak(enWord, TextToSpeech.QUEUE_FLUSH, null) }
                } else {  Log.w("Form-WordMeaningActivity", "Initialization Failed!")
}}}
        viewPager = findViewById(R.id.wordMeaningViewPager)
        viewPager?.let { setupViewPager(it) }
        val wordMeaningTabLayout = findViewById<TabLayout>(R.id.wordMeaningTabLayout)
        wordMeaningTabLayout.setupWithViewPager(viewPager)
        wordMeaningTabLayout.addOnTabSelectedListener(object : OnTabSelectedListener
{
            override fun onTabSelected(tab: TabLayout.Tab) { viewPager?.currentItem =
tab.position}
            override fun onTabUnselected(tab: TabLayout.Tab) {}
            override fun onTabReselected(tab: TabLayout.Tab) {} })}
    class ViewPagerAdapter internal constructor(fragmentManager: FragmentManager?) :
        FragmentPagerAdapter(fragmentManager!!) {
        private val mFragmentList: MutableList<Fragment> = ArrayList()
        private val mFragmentTitleList: MutableList<String> = ArrayList()
        fun addFragment(fragment: Fragment, title: String) {
            mFragmentList.add(fragment)
            mFragmentTitleList.add(title) }
        override fun getItem(position: Int): Fragment {return
mFragmentList[position]}
```

```
        override fun getCount(): Int {return mFragmentList.size}
        override fun getPageTitle(position: Int): CharSequence? { return
mFragmentTitleList[position]} }
    private fun setupViewPager(viewPager: ViewPager) {
        val viewPagerAdapter = ViewPagerAdapter(supportFragmentManager)
        viewPagerAdapter.addFragment(FragmentsWordExplanation(), "Word Explanation")
        viewPagerAdapter.addFragment(FragmentsWordDefinition(), "Word Definition")
        viewPager.adapter = viewPagerAdapter }
    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        if (whatDictionary != "Global") menuInflater.inflate(R.menu.main_menu, menu)
        return true }
    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        val id = item.itemId
        if (id == R.id.actionDelete) {
            myDbHelper.deleteFromNewWords(enWord)
            startActivity(Intent(this, SettingsActivity::class.java))
            return true }
        if (item.itemId == android.R.id.home) onBackPressed()
        return super.onOptionsItemSelected(item)
}}
```

Додаток Г

Лістинг Г.1 – Код класу LoadDatabaseAsync

```
class LoadDatabaseAsync(val context: Context) : AsyncTask<Void?, Void?, Boolean?>() {
    override fun doInBackground(vararg params: Void?): Boolean? {
        databaseHelper = DatabaseHelper(context)
        try {
            databaseHelper?.createDatabase()
        } catch (e: IOException) {
            throw Error("Database was not Created")
        }
        databaseHelper?.close()
        return null
    }
    override fun onProgressUpdate(vararg values: Void?) {
        super.onProgressUpdate(*values)
    }
    override fun onPostExecute(aBoolean: Boolean?) {
        super.onPostExecute(aBoolean)
        openDatabase()
    }
    @SuppressLint("LongLogTag")
    private fun openDatabase() {
        try {
            databaseHelper?.openDatabase()
            databaseOpened = true
            Log.w("Form-LoadDatabaseAsync-openDatabase", "$databaseHelper,
$databaseOpened")
        } catch (e: SQLException) {
            e.printStackTrace()
            Log.w("Form-LoadDatabaseAsync-openDatabase-error", "$e")
        }
    }
}
```

Лістинг Г.2 – Код класу DatabaseHelper

```
@SuppressLint("SdCardPath")
class DatabaseHelper(private val context: Context) : SQLiteOpenHelper(context,
DB_NAME, null, 1) {
    private var DB_PATH: String? = null
    private var myDatabase: SQLiteDatabase? = null
    @SuppressLint("LongLogTag")
    @Throws(IOException::class)
    fun createDatabase() {
        val dbExist = checkDatabase()
        if (!dbExist) {
            this.readableDatabase
            try {
                copyDatabase()
            } catch (e: IOException) {
                throw Error("Error Copying Database")
            }
        }
    }
    @SuppressLint("LongLogTag")
    fun checkDatabase(): Boolean {
```

```
        var checkDB: SQLiteDatabase? = null
        try {
            val myPath = DB_PATH + DB_NAME
            checkDB = SQLiteDatabase.openDatabase(myPath, null,
SQLiteDatabase.OPEN_READONLY)
        } catch (e: SQLiteException) {
            Log.w("Form-DatabaseHelper-checkDatabase-error", "$e")
        }
        checkDB?.close()
        return checkDB != null
    }
    @Throws(IOException::class)
    private fun copyDatabase() {
        val myInput = context.assets.open(DB_NAME)
        val outFileName = DB_PATH + DB_NAME
        val myOutput: OutputStream = FileOutputStream(outFileName)
        val buffer = ByteArray(64)
        var length: Int
        while (myInput.read(buffer).also { length = it } > 0) {
            myOutput.write(buffer, 0, length)
        }
        myOutput.flush()
        myOutput.close()
        myInput.close()
    }
    @Throws(SQLException::class)
    fun openDatabase() {
        val path = DB_PATH + DB_NAME
        myDatabase = SQLiteDatabase.openDatabase(path, null,
SQLiteDatabase.OPEN_READWRITE)
    }
    @Synchronized
    override fun close() {
        if (myDatabase != null) {
            myDatabase?.close()
        }
        super.close()
    }
    override fun onCreate(sqLiteDatabase: SQLiteDatabase) {}
    override fun onUpgrade(sqLiteDatabase: SQLiteDatabase, i: Int, i1: Int) {
        try {
            this.readableDatabase
            context.deleteDatabase(DB_NAME)
            copyDatabase()
        } catch (e: IOException) {
            e.printStackTrace()
}}
    fun getMeaningFromGlobal(text: String): Cursor? {
        return myDatabase?.rawQuery(
            "SELECT word_translate, word_level FROM words WHERE
en_word==UPPER('$text')", null)}
    fun getMeaningFromOwn(text: String): Cursor? {
        return myDatabase?.rawQuery(
            "SELECT word_translate, word_image FROM words_new WHERE
en_word==UPPER('$text')", null
        )
    }
    fun getSuggestionsFromGlobal(text: String): Cursor? {
        return myDatabase?.rawQuery(
            "SELECT _id, en_word FROM words WHERE en_word LIKE '$text%' LIMIT 40",
            null
)}
```

```
    fun insertIntoHistory(text: String) {
        myDatabase?.execSQL("INSERT INTO history(word) VALUES (UPPER ('$text'))")
    }
    fun insertDataIntoOwn(en_word: String, word_translate: String, word_image:
String) {
        myDatabase?.execSQL("INSERT INTO words_new(en_word, word_translate,
word_image) VALUES (UPPER ('$en_word'), UPPER('$word_translate'),
UPPER('$word_image'))")
    }
    val getInsertedImageFromOwn: Cursor?
        get() = myDatabase?.rawQuery(
            "SELECT  word_image FROM words_new", null
        )
    fun deleteFromNewWords(en_word: String) {
        myDatabase?.execSQL("DELETE FROM words_new WHERE en_word==UPPER('$en_word')")
    }
    fun deleteNewWords() {
        myDatabase?.execSQL("DELETE FROM words_new")
    }
    val getDictionaryOwn: Cursor?
        get() = myDatabase?.rawQuery(
            "select distinct en_word, word_translate from words_new order by _id
desc", null
        )
    val getHistory: Cursor?
        get() = myDatabase?.rawQuery(
            "select distinct word, word_translate from history h join words w on
h.word==w.en_word order by h._id desc", null
        )
    val getDictionaryGlobal: Cursor?
        get() = myDatabase?.rawQuery(
            "SELECT  en_word, word_translate FROM words GROUP BY en_word", null
        )
    fun deleteHistory() {
        myDatabase?.execSQL("DELETE FROM history")
    }
    companion object {
        private const val DB_NAME = "dictionary.db"
    }
    init {
        DB_PATH = "/data/data/com.example.anyword/databases/"
}}
```

Додаток Д

Лістинг Д.1 – Код об’єкту ConstantsForA1

```
object ConstantsForA1 {
    fun getQuestions(): ArrayList<QuestionClass> {
        val questionsList = ArrayList<QuestionClass>()
        val question1 = QuestionClass(1, "Як перекладається слово \"тварина\"?",
"animal", "annual", "annoy", "ankle", 1)
        questionsList.add(question1)
        val question2 = QuestionClass(1, "Оберіть правильний перевод для дієслова
\"можу\".", "cannot", "can", "scan", "could",2)
        questionsList.add(question2)
        val question3 = QuestionClass(1, "Як перекладається слово \"вчений\"?",
"science", "scientist", "scientific", "doctor", 2)
        questionsList.add(question3)
        val question4 = QuestionClass( 1, "Оберіть правильний переклад до слова
\"питання\".", "request", "queen", "question", "unique", 3)
        questionsList.add(question4)
        val question5 = QuestionClass( 1, "Як перекладається слово \"кожен\"?",
"clever", "several", "severe", "every", 4)
        questionsList.add(question5)
        val question6 = QuestionClass( 1, "Як перекладається слово
\"подія\"?","event","even","evening", "prevent",1)
        questionsList.add(question6)
        val question7 = QuestionClass(1, "Оберіть правильний переклад до слова
\"тиждень\".","week","weekend","sweep","sweet",)
        questionsList.add(question7)
        val question8 = QuestionClass(1,"Як перекладається слово \"урок\"?",
"unless","lesson","less","careless",2)
        questionsList.add(question8)
        val question9 = QuestionClass(1,"Оберіть правильний переклад до слова
\"старий\".","hold","cold","old","gold",3)
        questionsList.add(question9)
        val question10 = QuestionClass(1,"Оберіть правильний переклад до слова
\"голодний\".", "hungry", "hundred", "hunt", "hurry",1)
        questionsList.add(question10)
        return questionsList
}}
```

Лістинг Д.2 – Код об’єкту ConstantsForA2

```
object ConstantsForA2 {
    fun getQuestions(): ArrayList<QuestionClass> {
        val questionsList = ArrayList<QuestionClass>()
        val question1 = QuestionClass(1,"Як перекладається слово
\"закон\"?","lawsuit","lawyer","law","lawn",3)
        questionsList.add(question1)
        val question2 = QuestionClass(1,"Оберіть правильний перевод для дієслова
\"ділитися\".","shake","shall","share","sharp",3)
        questionsList.add(question2)
        val question3 = QuestionClass(1,"Як перекладається слово \"мати\"?","have
to", "have","behave","purchase",2)
        questionsList.add(question3)
        val question4 = QuestionClass(1,"Оберіть правильний переклад для дієслова
\"реагувати\".","reaction","reach","rich","react",4)
        questionsList.add(question4)
        val question5 = QuestionClass(1,"Як перекладається слово
```

```
\"носок\"?","sock","soccer","south", "son", 1 )
        questionsList.add(question5)
        val question6 = QuestionClass(1,"Як перекладається слово \"позначка\"?",
"mark", "march","marry", "remark", 1)
        questionsList.add(question6)
        val question7 = QuestionClass(1,"Оберіть правильний переклад до слова
\"мозок\".", "library","brand","brave","brain", 4)
        questionsList.add(question7)
        val question8 = QuestionClass( 1,"Як перекладається слово
\"пошук\"?","search","season","research","disease",1)
        questionsList.add(question8)
        val question9 = QuestionClass(1,"Оберіть правильний переклад до слова
\"завод\".","facility","factory","market","surface",2)
        questionsList.add(question9)
        val question10 = QuestionClass(1,"Оберіть правильний переклад до слова
\"позичати\".","silent","lend","length","plenty",2)
        questionsList.add(question10)
        return questionsList
}}
```

Лістинг Д.3 – Код об’єкту ConstantsForB1

```
object ConstantsForB1 {
    fun getQuestions(): ArrayList<QuestionClass> {
        val questionsList = ArrayList<QuestionClass>()
        val question1 = QuestionClass(1,"Як перекладається слово
\"чесний\"?","honest","honour","honor","hone",1)
        questionsList.add(question1)
        val question2 = QuestionClass(1,"Оберіть правильний перевод для слова
\"отрута\".","point","pool","poison","poor",3)
        questionsList.add(question2)
        val question3 = QuestionClass(1,"Як перекладається слово
\"філія\"?","branch","brand","brave","brain",1)
        questionsList.add(question3)
        val question4 = QuestionClass(1,"Оберіть правильний перевод для слова
\"мирний\".","peaceful","peace","place","appeal",1)
        questionsList.add(question4)
        val question5 = QuestionClass(1,"Як перекладається слово
\"відсутність\"?","black", "lack","place", "lake", 2)
        questionsList.add(question5)
        val question6 = QuestionClass(1,"Як перекладається слово \"досліджувати\"?",
"search","explore","explain","explode", 2)
        questionsList.add(question6)
        val question7 = QuestionClass(1,"Оберіть правильний переклад до слова
\"сильно\".","heavy", "hear","heavily", "heaven", 3)
        questionsList.add(question7)
        val question8 = QuestionClass( 1, "Як перекладається слово
\"казка\"?","tale","tall","total","vital", 1)
        questionsList.add(question8)
        val question9 = QuestionClass(1,"Оберіть правильний переклад до слова
\"застосовувати\".","approach","apply", "application","appeal",2)
        questionsList.add(question9)
        val question10 = QuestionClass( 1, "Оберіть правильний переклад до слова
\"пара\".", "court","count", "could", "couple", 4)
        questionsList.add(question10)
        return questionsList
}}
```

Лістинг Д.4 – Код об’єкту ConstantsForB2

```
object ConstantsForB2 {
    fun getQuestions(): ArrayList<QuestionClass> {
        val questionsList = ArrayList<QuestionClass>()
        val question1 = QuestionClass( 1, "Як перекладається дієслово
\"переконувати\"?", "convert", "convince", "concept", "contest", 1)
        questionsList.add(question1)
        val question2 = QuestionClass( 1, "Оберіть правильний перевод для слова
\"порядок денний\".",  "agenda", "agent","average", "stare", 1)
        questionsList.add(question2)
        val question3 = QuestionClass( 1,"Як перекладається слово
\"озброєнний\"?","alarm","warm", "farm","armed",4)
        questionsList.add(question3)
        val question4 = QuestionClass( 1, "Оберіть правильний перевод для слова
\"місце проведення\".","heaven","venue","oven","event", 2)
        questionsList.add(question4)
        val question5 = QuestionClass( 1,"Як перекладається слово \"постійно\"?",
"conscious","constantly", "construction","consider", 2)
        questionsList.add(question5)
        val question6 = QuestionClass(1, "Як перекладається слово \"вексель\"?",
"bill", "fill","hill", "silly", 1)
        questionsList.add(question6)
        val question7 = QuestionClass(1, "Оберіть правильний переклад до слова
\"мудрий\".", "wine","wise",  "wish","wired", 2)
        questionsList.add(question7)
        val question8 = QuestionClass(1,"Як перекладається слово \"тюрма\"?",
"pristine","prison", "poison", "prism", 2)
        questionsList.add(question8)
        val question9 = QuestionClass( 1, "Оберіть правильний переклад до слова
\"судовий розгляд\".", "trial", "court", "law","strict", 1)
        questionsList.add(question9)
        val question10 = QuestionClass(1,"Оберіть правильний переклад до слова
\"каблук\".", "heel", "sheet", "wheel","sheep", 1)
        questionsList.add(question10)
        return questionsList
    }
}
```

Лістинг Д.5 – Код форми QuizQuestionsActivity

```
class QuizQuestionsActivity : AppCompatActivity(), View.OnClickListener {
    private var mCurrentPosition: Int = 1
    private var mQuestionList: ArrayList<QuestionClass>? = null
    private var mSelectedOptionPosition: Int = 0
    private var optionFirst: TextView? = null
    private var optionSecond: TextView? = null
    private var optionThird: TextView? = null
    private var optionFourth: TextView? = null
    private var submitButton: Button? = null
    private var progressCount: TextView? = null
    private var mainQuestion: TextView? = null
    private var progressBar: ProgressBar? = null
    private var numberOfCorrectAnswers = 0
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_quiz_questions)
        val quizToolbar = findViewById<Toolbar>(R.id.quizToolbar)
        quizToolbar.title = "Test $levelTest"
        setSupportActionBar(quizToolbar)
        if (levelTest == "B2") {
            mQuestionList = ConstantsForB2.getQuestions()
```

```
        } else if (levelTest == "B1") {
            mQuestionList = ConstantsForB1.getQuestions()
        } else if (levelTest == "A2") {
            mQuestionList = ConstantsForA2.getQuestions()
        } else {
            mQuestionList = ConstantsForA1.getQuestions()
        }
        optionFirst = findViewById(R.id.optionFirst)
        optionSecond = findViewById(R.id.optionSecond)
        optionThird = findViewById(R.id.optionThird)
        optionFourth = findViewById(R.id.optionFourth)
        submitButton = findViewById(R.id.submitButton)
        progressBar = findViewById(R.id.progressBar)
        progressCount = findViewById(R.id.progressCount)
        mainQuestion = findViewById(R.id.mainQuestion)
        optionFirst?.setOnClickListener(this)
        optionSecond?.setOnClickListener(this)
        optionThird?.setOnClickListener(this)
        optionFourth?.setOnClickListener(this)
        submitButton?.setOnClickListener(this)
        submitButton?.isEnabled = false
        startTimer()
    }
    private fun startTimer() {
        var counter = 0
        val timer = Timer()
        timer.schedule(object : TimerTask() {
            override fun run() {
                runOnUiThread {
                    counter++
                    if (counter == 1) {
                        timer.cancel()
                        setQuestion()
}}}}, 0, 1000)}
    @SuppressLint("SetTextI18n")
    private fun setQuestion() {
        val question = mQuestionList?.get(mCurrentPosition - 1)
        defaultOptionsView()
        if (mCurrentPosition == mQuestionList?.size) {
            submitButton?.text = "Finish"
        } else {
            submitButton?.text = "Submit"
        }
        progressBar?.progress = mCurrentPosition
        progressCount?.text = "$mCurrentPosition" + "/" + progressBar?.max
        mainQuestion?.text = question?.question
        optionFirst?.text = question?.optionOne
        optionSecond?.text = question?.optionTwo
        optionThird?.text = question?.optionThree
        optionFourth?.text = question?.optionFour
    }
    private fun defaultOptionsView() {
        val options = ArrayList<TextView>()
        optionFirst?.let { options.add(0, it) }
        optionSecond?.let { options.add(1, it) }
        optionThird?.let { options.add(2, it) }
        optionFourth?.let { options.add(3, it) }
        for (option in options) {
            option.setTextColor(Color.parseColor("#7A8089"))
            option.typeface = Typeface.DEFAULT
            option.background = ContextCompat.getDrawable(
                this,
                R.drawable.default_option_border_bg
)}}
    @SuppressLint("SetTextI18n")
```

```
    override fun onClick(view: View?) {
        when (view?.id) {
            R.id.optionFirst -> { optionFirst?.let { selectedOptionView(it, 1) } }
            R.id.optionSecond -> { optionSecond?.let { selectedOptionView(it, 2) }}
            R.id.optionThird -> {optionThird?.let { selectedOptionView(it, 3) }}
            R.id.optionFourth -> {optionFourth?.let { selectedOptionView(it, 4) }}
            R.id.submitButton -> {
                if (mSelectedOptionPosition != 0) {
                    val question = mQuestionList?.get(mCurrentPosition - 1)
                    if (question?.correctOption != mSelectedOptionPosition) {
                        answerView(mSelectedOptionPosition,
R.drawable.wrong_option_border_bg)
                    } else {
                        numberOfCorrectAnswers++ }
                    if (question != null) {
                        answerView(question.correctOption,
R.drawable.correct_option_border_bg)
                    }
                    if (mCurrentPosition == mQuestionList?.size) {
                        submitButton?.text = "Finish"
                    } else {
                        submitButton?.text = "Go to next question" }
                    mSelectedOptionPosition = 0
                } else {
                    mCurrentPosition++
                    submitButton?.isEnabled = false
                    when {
                        mCurrentPosition <= mQuestionList?.size!! -> {setQuestion()}
                        else -> {
                            Toast.makeText(
                                this,
                                "You have successfully completed the Test. Your score
on the test $levelTest is $numberOfCorrectAnswers.",
                                Toast.LENGTH_LONG).show()
                            startActivity(Intent(this, PracticeActivity::class.java))
}}}}}}
    private fun selectedOptionView(tv: TextView, selectedOptionNum: Int) {
        defaultOptionsView()
        mSelectedOptionPosition = selectedOptionNum
        tv.setTextColor(Color.parseColor("#363A43"))
        tv.setTypeface(tv.typeface, Typeface.BOLD)
        tv.background = ContextCompat.getDrawable(
            this,
            R.drawable.selected_option_border_bg
        )
        submitButton?.isEnabled = true
    }
    private fun answerView(answer: Int, drawableView: Int) {
        when (answer) {
            1 -> {
                optionFirst?.background = ContextCompat.getDrawable(
                    this, drawableView )}
            2 -> {
                optionSecond?.background = ContextCompat.getDrawable(
                    this, drawableView)}
            3 -> {
                optionThird?.background = ContextCompat.getDrawable(
                    this, drawableView)}
            4 -> {
                optionFourth?.background = ContextCompat.getDrawable(
                    this, drawableView
)}}}}
```

Додаток Е

Лістинг Е.1 – Код форми DictionaryActivity

```
var dictionaryManager: FragmentManager? = null
var whatDictionary = "Global"
class DictionaryActivity : AppCompatActivity() {
    private var dictionaryGlobalRecycler: RecyclerView? = null
    private var dictionaryOwnRecycler: RecyclerView? = null
    private var databaseFunction: Cursor? = null
    private var dictionaryOwnText: TextView? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_dictionary)
        dictionaryManager = supportFragmentManager
        val dictionaryToolbar = findViewById<Toolbar>(R.id.dictionaryToolbar)
        setSupportActionBar(dictionaryToolbar)
        dictionaryToolbar.setNavigationOnClickListener {
            startActivity(Intent(this, SettingsActivity::class.java))
        }
        dictionaryGlobalRecycler = findViewById(R.id.dictionaryGlobalRecyclerView)
        dictionaryOwnRecycler = findViewById(R.id.dictionaryOwnRecyclerView)
        dictionaryOwnText = findViewById(R.id.dictionaryOwnText)
        dictionaryOwnText?.isVisible = false
        whatDictionary = "Global"
        dictionaryOwnRecycler?.isVisible = false
        val layoutManager = LinearLayoutManager(this)
        val layoutManager1 = LinearLayoutManager(this)
        dictionaryGlobalRecycler?.layoutManager = layoutManager
        dictionaryOwnRecycler?.layoutManager = layoutManager1
        databaseFunction = databaseHelper?.getDictionaryGlobal
        val dictionaryGlobal: TextView = findViewById(R.id.dictionaryGlobal)
        val dictionaryOwn: TextView = findViewById(R.id.dictionaryOwn)
        dictionaryGlobal.setOnClickListener {
            dictionaryGlobalRecycler?.layoutManager = layoutManager
            databaseFunction = databaseHelper?.getDictionaryGlobal
            dictionaryOwnRecycler?.isVisible = false
            dictionaryGlobalRecycler?.isVisible = true
            whatDictionary = "Global"
            fetchDictionary()
            dictionaryOwn.setTextColor(applicationContext.getColor(R.color.hintGray))
            dictionaryGlobal.setTextColor(applicationContext.getColor(R.color.black))
            dictionaryOwnText?.isVisible = false
        }
        dictionaryOwn.setOnClickListener {
            dictionaryOwnRecycler?.layoutManager = layoutManager1
            databaseFunction = databaseHelper?.getDictionaryOwn
            dictionaryGlobalRecycler?.isVisible = false
            dictionaryOwnRecycler?.isVisible = true
            whatDictionary = "Own"
            fetchDictionary()
            dictionaryOwn.setTextColor(applicationContext.getColor(R.color.black))

dictionaryGlobal.setTextColor(applicationContext.getColor(R.color.hintGray))
        }
        fetchDictionary()
    }
    @SuppressLint("LongLogTag")
    private fun fetchDictionary() {
        val dictionaryList = ArrayList<Dictionary>()
```

```
        val dictionaryAdapter = DictionaryAdapter(this, dictionaryList, Intent(this,
WordMeaningActivity::class.java))
        if (whatDictionary == "Global") {
            dictionaryGlobalRecycler?.adapter = dictionaryAdapter
        } else {
            dictionaryOwnRecycler?.adapter = dictionaryAdapter
        }
        if (databaseOpened) {
            val cursorDictionary = databaseFunction
            if (cursorDictionary?.moveToFirst() == true) {
                do {
                    val dictionary = Dictionary(
cursorDictionary.getString(cursorDictionary.getColumnIndex("en_word")),

cursorDictionary.getString(cursorDictionary.getColumnIndex("word_translate"))
                    )
                    dictionaryList.add(dictionary)
                } while (cursorDictionary.moveToNext())
            } else {
                Log.w("Form-DictionaryActivity-fetchHistory",
"cursorDictionary?.moveToFirst() == false")
                dictionaryOwnText?.isVisible = true
            }
            dictionaryAdapter.notifyDataSetChanged()
        } else {
            Log.w("Form-DictionaryActivity-fetchHistory", "databaseOpened == false")
        }
    }
    fun goToMainActivity(view: View) {
        startActivity(Intent(this, MainActivity::class.java))
    }
    fun goToPracticeActivity(view: View) {
        startActivity(Intent(this, PracticeActivity::class.java))
    }
    fun goToSettingsActivity(view: View) {
        startActivity(Intent(this, SettingsActivity::class.java))
    }
}
```

Лістинг Е.2 – Код форми AddWordActivity

```
var addWordResolver: ContentResolver? = null
class AddWordActivity : AppCompatActivity() {
    private var pickImage = 1
    private var imageUri: Uri? = null
    private var imageName: String? = null
    private val imageValues = ContentValues()
    private var imageType = String()
    private var dialogAddImage: Dialog? = null
    private var addWordImage: ImageView? = null
    private var addWordFrame: ImageView? = null
    private var addWordVIew: View? = null
    private var myDbHelper: DatabaseHelper? = null
    private var realPathFromURI = String()
    private fun String.toEditable(): Editable =
Editable.Factory.getInstance().newEditable(this)
    private var timerCheckFields = Timer()
    private var addWordInputWord: TextInputLayout? = null
    private var addWordInputTranslate: TextInputLayout? = null
    private var addWordButton: Button? = null
    @SuppressLint("LongLogTag", "UseCompatLoadingForDrawables")
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_add_word)
        addWordResolver = contentResolver
        if (checkSelfPermission(Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED
            ||
            checkSelfPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE) !=
PackageManager.PERMISSION_GRANTED
        ) {
            requestPermissions(
                arrayOf(
                    Manifest.permission.CAMERA,
                    Manifest.permission.WRITE_EXTERNAL_STORAGE
), 1)}
        runTimerCheckFields()
        myDbHelper = DatabaseHelper(this)
        try {
            myDbHelper?.openDatabase()
        } catch (sqle: SQLException) {
            throw sqle
        }
        val addWordToolbar = findViewById<Toolbar>(R.id.addWordToolbar)
        setSupportActionBar(addWordToolbar)
        addWordToolbar.setNavigationOnClickListener {
            startActivity(Intent(this, SettingsActivity::class.java))
        }
        addWordImage = findViewById(R.id.addWordImage)
        addWordFrame = findViewById(R.id.addWordFrame)
        addWordVIew = findViewById(R.id.addWordVIew)
        addWordInputWord = findViewById(R.id.addWordInputWord)
        addWordInputTranslate = findViewById(R.id.addWordInputTranslate)
        val addWordAddImage = findViewById<Button>(R.id.addWordAddImage)
        addWordAddImage.setOnClickListener {
            dialogAddImage = Dialog(this, R.style.DialogSheetTheme)
            val layoutDialogAddImage =
LayoutInflater.from(applicationContext).inflate(
                R.layout.layout_dialog_add_image,
                findViewById<ConstraintLayout>(R.id.layoutDialogAddImage) )
            dialogAddImage?.setContentView(layoutDialogAddImage)
            dialogAddImage?.setCancelable(false)
            dialogAddImage?.setCanceledOnTouchOutside(false)
            dialogAddImage?.show()
            val addImageTakePicture =
                layoutDialogAddImage.findViewById<Button>(
                    R.id.addImageTakePicture
                )
            addImageTakePicture.setOnClickListener {
                imageValues.put(MediaStore.Images.Media.TITLE, "New picture")
                imageValues.put(MediaStore.Images.Media.DESCRIPTION, "From the
camera")
                imageUri = contentResolver.insert(
                    MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
                    imageValues
                )
                imageType = "camera"
                val cameraIntent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
                cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, imageUri)
                intent.action = MediaStore.ACTION_IMAGE_CAPTURE
                startActivityForResult(cameraIntent, pickImage)
            }
            val addImageChooseImage =
```

```
                layoutDialogAddImage.findViewById<Button>(R.id.addImageChooseImage)
            addImageChooseImage.setOnClickListener {
                val photoPickerIntent = Intent(Intent.ACTION_PICK)
                imageType = "gallery"
                photoPickerIntent.type = "image/*"
                startActivityForResult(photoPickerIntent, pickImage)
            }
            layoutDialogAddImage.findViewById<Button>(R.id.addImageCancel)
                .setOnClickListener {dialogAddImage?.dismiss()}
        }
        addWordButton = findViewById<Button>(R.id.addWordButton)
        addWordButton?.setOnClickListener {
            myDbHelper?.insertDataIntoOwn(
                addWordInputWord?.editText?.text.toString(),//.toUpperCase(),
                addWordInputTranslate?.editText?.text.toString(),
                realPathFromURI )
            val wordAddedSheet = BottomSheetDialog(this, R.style.BottomSheetTheme)
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O_MR1)
setTransparentNavigationBar(wordAddedSheet)
            val bottomWordAddedSheet =
LayoutInflater.from(applicationContext).inflate(
                R.layout.layout_word_added,
                findViewById<ConstraintLayout>(R.id.bottomWordAddedSheet) )
            bottomWordAddedSheet.findViewById<ImageButton>(R.id.wordAddedClose)
                .setOnClickListener {
                    wordAddedSheet.dismiss()
                    startActivity(Intent(this, SettingsActivity::class.java)) }
            wordAddedSheet.setContentView(bottomWordAddedSheet)
            wordAddedSheet.setCancelable(false)
            wordAddedSheet.setCanceledOnTouchOutside(false)
            wordAddedSheet.show()
}}
    private fun runTimerCheckFields() {
        timerCheckFields.schedule(object : TimerTask() {
            override fun run() {
                runOnUiThread {
                    addWordButton?.isEnabled = false
                    allFieldsAreCompleted()
}}}, 0, 500)}
    fun allFieldsAreCompleted() {
        val addWordField =
            addWordInputWord?.editText?.text.toString()
        val addWordTranslateField =
            addWordInputTranslate?.editText?.text.toString()
        if (addWordField.isNotEmpty() && addWordTranslateField.isNotEmpty()) {
            addWordButton?.isEnabled = true
            timerCheckFields.cancel()
}}
    @SuppressLint("LongLogTag", "UseCompatLoadingForDrawables")
    @RequiresApi(Build.VERSION_CODES.M)
    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        when (requestCode) {
            pickImage -> if (resultCode == Activity.RESULT_OK) {
                try {
                    if (imageType == "camera") {
                        imageUri
                    } else if (imageType == "gallery") {
                        imageUri = data?.data }
                    realPathFromURI = getRealPathFromURI(imageUri, addWordResolver!!)
                    Glide.with(this)
                        .load(realPathFromURI)
```

```
                        .into(addWordImage!!)
                    imageName = getFileName(imageUri, addWordResolver!!)
                    if (imageName != null) {
                        dialogAddImage?.dismiss()
                        val dialogConfirmImage = Dialog(this,
R.style.DialogSheetTheme)
                        val layoutDialogConfirmImage =
                            LayoutInflater.from(applicationContext).inflate(
                                R.layout.layout_dialog_confirm_image,
findViewById<ConstraintLayout>(R.id.layoutDialogConfirmImage)
                            )
                        dialogConfirmImage.setContentView(layoutDialogConfirmImage)
                        dialogConfirmImage.setCancelable(false)
                        dialogConfirmImage.setCanceledOnTouchOutside(false)
                        dialogConfirmImage.show()
layoutDialogConfirmImage.findViewById<Button>(R.id.confirmImageAgree)
                            .setOnClickListener {
                                dialogConfirmImage.dismiss()
addWordFrame?.setImageResource(R.drawable.ic_square_active)
}
layoutDialogConfirmImage.findViewById<Button>(R.id.confirmImageDismiss)
                            .setOnClickListener {

Glide.with(this).load(getDrawable(R.mipmap.ic_photo_bg)).into(addWordImage!!)
                                dialogConfirmImage.dismiss()
addWordFrame?.setImageResource(R.drawable.ic_square_inactive)
}}
                } catch (e: FileNotFoundException) {
                    e.printStackTrace()
}}}}
    fun goToMainActivity(view: View) {
        startActivity(Intent(this, MainActivity::class.java)) }
    fun goToPracticeActivity(view: View) {
        startActivity(Intent(this, PracticeActivity::class.java)) }
    fun goToSettingsActivity(view: View) {
        startActivity(Intent(this, SettingsActivity::class.java)) }
}
```