Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

Кваліфікаційна робота

за освітнім ступенем «магістр»

на тему:

ІНТЕРАКТИВНИЙ МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ У СФЕРІ ІТ

Виконав:
здобувач 2 курсу
групи М-КН-21
спеціальності 122 «Комп'ютерні науки»
Іван Сергійович Гарбарчук

Науковий керівник:
кандидат технічних наук, доцент
Степанія Михайлівна Бабич

Рівне – 2025

ЗМІСТ

ВСТУП

Актуальність дослідження. У сучасному світі інформаційних технологій фахівці у сфері ІТ повинні володіти не лише технічними знаннями, а й високим рівнем англійської мови, яка є міжнародним стандартом спілкування в цій галузі. Англійська мова відкриває доступ до актуальної спеціалізованої літератури, документації, технічних форумів, онлайн-курсів, а також до передових світових технологічних трендів. Власне без знання англійської складно ефективно працювати з популярними програмними бібліотеками та інструментами, які майже виключно мають англомовну документацію.

Значна частина ІТ-продуктів створюється у міжнародних командах, де комунікація відбувається англійською мовою. Відтак її знання є визначальним фактором для ефективної професійної взаємодії та кар’єрного зростання. Водночас багато ІТ-спеціалістів відчувають труднощі у вивченні англійської через нестачу часу або невисоку ефективність традиційних навчальних інструментів, які не враховують специфіку технічної термінології та професійного сленгу.

У такому контексті розробка мобільного застосунку, орієнтованого на вивчення англійської мови в ІТ-контексті, є надзвичайно актуальною. Мобільні рішення забезпечують зручність, доступність у будь-який час і можливість персоналізованого навчання, що особливо важливо для зайнятих фахівців. Такий застосунок дозволяє поєднувати вивчення термінології, ідіом та технічного сленгу з інтерактивними тестами та вправами, що сприяє закріпленню знань у практичних ситуаціях.

Отже, створення мобільного застосунку для покращення мовних компетенцій ІТ-фахівців є важливим кроком у підвищенні їхньої конкурентоспроможності, професійної мобільності та здатності інтегруватися в міжнародні проєкти.

Метою магістерської роботи є розробка мобільного застосунку, спрямованого на підвищення ефективності вивчення англійської мови ІТ-фахівцями.

Для досягнення мети були вирішені такі **завдання** дослідження:

1. проаналізувати особливості освітніх мобільних застосунків для визначення функціональних та нефункціональних вимог до програмної системи для вивчення англійської мови;
2. проаналізувати особливості функціонування різних мобільних платформ;
3. спроєктувати програмну систему для вивчення англійської мови ІТ-фахівцями;
4. визначити інструменти розробки;
5. провести UI/UX-дизайн для розроблювального мобільного додатку;
6. розробити на основі проведеного дослідження мобільний додаток та протестувати його роботу.

Об’єкт дослідження: процес навчання англійської мови професійного спрямування засобами інформаційних технологій.

Предмет дослідження: програмні засоби та методи розроблення мобільного застосунку для підтримки навчання англійської мови ІТ-фахівцями.

Методи дослідження. У даній роботі теоретичною основою досліджень є: методи аналізу літератури та наукових джерел: огляд публікацій пов’язаних з розробкою мобільних додатків; методи управління вимогами: визначення вимог користувача, а саме функціоналах та не функціоналах вимог; методи алгоритмічного програмування для створення функціоналу мобільного застосунку; методи аналізу для вибору відповідних технологій, платформи для розробки програмної системи для вивчення англійської мови ІТ-фахівцями.

Практичне значення. Розроблений мобільний застосунок може бути використаний студентами ІТ-спеціальностей, викладачами та фахівцями-

початківцями для вдосконалення знань англійської мови у професійному контексті.

Результати дослідження можуть стати основою для подальших розробок у галузі мобільного навчання, зокрема для створення інтелектуальних систем адаптивного навчання англійської мови.

Апробація і впровадження результатів дослідження. Основні теоретичні та практичні результати дослідження доповідалися та обговорювалися на XVIII міжнародній науково-практичній конференції «Інформаційні технології та автоматизація – 2025» (м. Одеса, 30-31 жовтня 2025 р.), звітній науковій конференції викладачів, співробітників і здобувачів вищої освіти Рівненського державного гуманітарного університету за 2024 рік (м. Рівне, 15 травня 2025 року).

Структура та обсяг роботи. Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел із найменувань, додатків і містить 70 сторінок основного тексту, 32 рисунків і 4 таблиці.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ЗАСАДИ СТВОРЕННЯ МОБІЛЬНИХ ЗАСТОСУНКІВ ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ ДЛЯ ІТ-СПЕЦІАЛІСТІВ

1.1. Особливості проєктування освітніх мобільних застосунків

Ринок освітніх технологій переживає бум, очікується, що до 2026 року він досягне 370 мільярдів доларів, оскільки все більше навчальних закладів, спеціалізованих курсів, здобувачів формальної та неформальної освіти звертаються до онлайн-платформ.

У 2024 році вартість освітніх додатків становила 6,2 мільярда доларів, а до 2033 року ця цифра становитиме 41,6 мільярда доларів. Такі додатки вже зараз є однією з провідних категорій в App Store та Google Play, маючи 709 мільйонів користувачів та майже мільярд завантажень лише у 2023 році [1].

Ринок розробки освітніх додатків створює величезні можливості для запуску цифрового рішення. Мільйони студентів, викладачів та тих, хто навчається протягом усього життя, шукають розумніші способи отримання нових навичок. Використання штучного інтелекту призводить до більшої залученості: адаптивний навчальний досвід заохочує користувачів повертатися, збільшуючи утримання та цінність протягом усього життя.

Існує декілька типів навчальних платформ. Розглянемо їх.

1. Багатоцільові навчальні платформи – це рішення, які пропонують курси з багатьох предметів, від STEM до бізнесу та мистецтва. Деякі з них надають сертифікати або ступені, тому вони підходять як для студентів, так і для професіоналів. До таких платформ відносяться Академія Хана (математика, природничі науки, гуманітарні науки), Udemy (бізнес, технології, творче мистецтво) та Coursera (інформатика, бізнес, соціальні науки).

2. Предметні додатки розроблені для глибокого навчання в одній галузі, вони використовують інтерактивні інструменти, мультимедійний контент та самооцінювання, щоб підтримувати зацікавленість користувача. Одним з

найяскравіших прикладів таких додатків є Duolingo (для вивчення мов), Photomath (для вивчення математики), Yousician (для вивчення музики).

3. Платформи оцінювання – це цифрові інструменти, які допомагають вчителям оцінювати прогрес учнів за допомогою вікторин, тестів та інтерактивних оцінювань. Вони пропонують негайний зворотний зв'язок, відстежують прогрес та адаптують навчальний досвід до індивідуальних потреб. Одним з найбільш відомих є Quizizz, що використовується для інтерактивного оцінювання та ProProfs, за допомогою якого проводиться онлайн-тестування та сертифікація.

4. Освітні ігри – це додатки, які містять безліч винагород, рівнів та завдань, які справді залучають учнів. Інтерактивний досвід та багаторазова практика роблять навчання цікавішим та ефективнішим. Наприклад, Prodigy пропонує фентезійну математичну гру, Minecraft навчає програмуванню та математиці під час створення віртуального світу або Wordscapes представляє мовну гру зі словами.

5. Додатки для управління навчанням розроблені, щоб допомогти навчальним закладам та підприємствам керувати курсами, відстежувати прогрес та підтримувати комунікацію між вчителями та учнями. Вони спрощують адміністративні завдання, такі як планування, оцінювання та розподіл контенту. Одним з найбільш поширених є безкоштовний Google Classroom [1,2].

Мобільні програми відіграють величезну роль у сучасному світі, забезпечуючи зручний доступ до різних послуг та інформації. Основні цілі та значення проектування мобільних додатків можна визначити таким чином.

Мобільні технології проникли в сучасну освіту та стали одним із ключових елементів, який не лише прискорює успішність тих, хто навчається, але й робить навчальний процес більш захопливим та змістовним. Мобільні технології заохочують користувачів контролювати своє самостійне навчання. Вони можуть використовувати онлайн-словники, інтерактивні додатки для

навчання, шукати інформацію через Google та читати електронні книги, не обмежуючи себе протягом певних годин.

Освітні мобільні програми останні кілька років користуються великою популярністю. Вони допомагають оптимізувати навчальний процес, спростити завдання навчання для викладачів та учнів. Станом на 3-й квартал 2023 року середній споживач витрачає 5,05 доларів США на мобільні додатки на один смартфон [2]. Мобільні додатки допомагають нам спілкуватися, розважатися та отримувати інформацію. Багато додатків використовуються в освітніх цілях. Університети та школи по всьому світу вже використовують мобільні додатки для полегшення освітнього процесу. Мобільні програми для навчання значно підвищують якість освітнього процесу. Серед основних особливостей слід виділити такі:

1. Доступність, коли користувачі можуть отримати доступ до навчального матеріалу в будь-якому місці та в будь-який час за допомогою смартфонів або планшетів.

2. Індивідуалізація, при якій мобільні програми підлаштовуються під індивідуальні потреби кожного користувача, пропонуючи персональний контент та можливість регулювання темпу вивчення матеріалу.

3. Мотивація, яка включає ігрові елементи, інтерактивний формат та можливість відстежувати свій прогрес, що власне потенційно підвищує прагнення учнів до вивчення нового матеріалу.

За допомогою мобільних додатків користувачів заохочують використовувати свої бажані стратегії навчання поза традиційним навчанням незалежно від часу чи місця, використовуючи портативні мобільні пристрої та мобільні технології [4, 5]. В результаті, користувачі можуть ретельно практикувати та вдосконалювати свої цільові мовні навички, такі як аудіювання, читання та словниковий запас у неформальній навчальній обстановці. Вони здатні самостійно вивчати мову [6].

Студенти мають можливість використовувати мобільні додатки замість формальних навчальних середовищ. У цьому відношенні мобільні технології є

ефективним інструментом для забезпечення навчання всім, хто в іншому випадку міг би залишитися на периферії освіти. Додатки для мобільного вивчення мов поєднують академічні та неформальні навчальні умови. Крім того, після завершення кожної вправи мобільні додатки для вивчення мов надають швидкий зворотний зв’язок, щоб учні могли швидше засвоїти правильні форми [7].

З іншої сторони, у самостійному навчанні через мобільні додатки учні більш схильні завершити курс, якщо він не надто довгий. Це пояснюється тим, що тривалий курс змусить користувачів почуватися нудьгуючими, і вони можуть почуватися обтяженими, оскільки їм доведеться поєднувати роботу та самостійне навчання. Особи, що самостійно навчаються, віддають перевагу більш релевантному та коротшому курсу, який вони можуть вивчати у власному темпі, і результат якого є досяжним.

Отже, на продовження використання мобільних додатків для навчання впливають такі фактори, як простота використання та корисність, надійність, доступність та забезпечення комфортного навчального середовища [8].

Мобільні додатки для вивчення мов допомагають покращити словниковий запас, оскільки навчальні інструменти мають на меті допомогти учням легко отримувати доступ до інформації. Мобільні пристрої можна використовувати в контексті неформального навчання. Більшість користувачів сприймають мобільні пристрої як більш персоналізоване навчання, ніж самостійне, з точки зору автентичності навчання та соціальних зв’язків.

Однією з головних особливостей освітніх додатків є гнучкість. Користувачі можуть вибирати з різних предметів, тем та рівнів складності, що дозволяє їм адаптувати навчальний процес до індивідуальних потреб та інтересів. Деякі освітні додатки мають адаптивні алгоритми, які автоматично визначають рівень знань та підбирають відповідний матеріал.

Освітні додатки – це потужний інструмент сучасної освіти, який може зробити навчання більш доступним, персоналізованим та ефективним. Вони є частиною цифрової трансформації в освіті.

Зазвичай вони містять багато інтерактивного контенту, такого як відео, аудіо, анімація, тексти та вікторини, що робить процес навчання візуальним та привабливим. Процес навчання стає більш мотивуючим та захопливим завдяки гейміфікації, сприяючи довгостроковому навчанню. Також суттєвим аспектом освітніх додатків є можливість відстежувати прогрес та оцінювати результати. Багато додатків пропонують статистику, аналіз помилок та рекомендації щодо покращення, що дозволяє користувачам оцінювати свої досягнення та виявляти слабкі місця для подальшого розвитку.

Мобільні освітні застосунки забезпечують низку переваг, які сприяють ефективнішому засвоєнню навчального матеріалу. Однією з ключових характеристик є їхня доступність у будь-який час та в будь-якому місці, що дозволяє учням навчатися у власному темпі, незалежно від географічного розташування. Це особливо актуально в умовах зростання попиту на дистанційне навчання та гнучкі освітні формати.

Іншою важливою особливістю є інтерактивність, що реалізується через виконання завдань, які потребують активної участі користувача. Такий підхід підвищує рівень залученості та сприяє ґрунтовнішому розумінню матеріалу.

Крім того, сучасні навчальні програми часто базуються на адаптивних технологіях. Зокрема, вони використовують алгоритми машинного навчання для автоматичного налаштування змісту та складності завдань відповідно до індивідуального рівня володіння мовою кожного користувача.

Для забезпечення зручності використання мобільних освітніх програм доцільно передбачити реалізацію низки ключових функціональних елементів, спрямованих на підвищення ефективності навчального процесу. Одним із таких елементів є структуризація навчального матеріалу у вигляді окремих тематичних блоків. Такий підхід сприяє кращій концентрації уваги користувача на засвоєнні нової інформації та дозволяє організувати навчання відповідно до логічної послідовності викладу змісту.

Не менш важливим є впровадження механізмів відстеження навчального прогресу. Фіксація досягнень, виконаних завдань, а також наявність

вбудованої системи мотивації у вигляді нагород або балів позитивно впливають на рівень зацікавленості користувача та стимулюють до регулярного навчання.

Окрему увагу слід приділити функціоналу, який дозволяє відтворювати навчальний контент у фоновому режимі або при заблокованому екрані, а також автоматично трансформувати візуальні матеріали у формат аудіо. Така можливість є особливо актуальною для користувачів, які прагнуть використовувати вільний час під час пересування або виконання інших повсякденних завдань.

У процесі розробки мобільних освітніх застосунків необхідно враховувати індивідуальні потреби та особливості користувачів, що передбачає створення адаптивного, гнучкого та зручного середовища для навчання з урахуванням сучасних технологічних можливостей.

Слід також розробляти освітній додаток з урахуванням потреб користувача; під час розробки додатка слід враховувати вік, стиль навчання та освітній рівень цільової аудиторії.

Найкращі освітні додатки пропонують користувачам адаптивне навчання, тобто коригують навчальний процес залежно від прогресу та здібностей користувача. Таким чином, підсумовуючи, можемо сказати, що освітні додатки стають зручним та ефективним інструментом саме для самонавчання користувачів, тобто для користувачів, які замотивовані у вивченні того чи іншого курсу.

1.2. Використання мобільних застосунків для вивчення англійської мови та їх особливості у сфері ІТ

У сучасних умовах цифрової трансформації освітнього процесу мобільні застосунки посідають важливе місце серед інструментів самостійного вивчення іноземних мов. Вони поєднують у собі доступність,

мультимедійність та інтерактивність, що робить навчання більш персоналізованим та ефективним.

Мобільне навчання мови (MALL) визначається як використання мобільних технологій у підтриманому вивченні мови та характеризується як інструмент для вивчення мови в неформальному навчальному середовищі, доступний за матеріалами та доступний для учнів. Використовуючи мобільні програми для вивчення мов, користувачі матимуть досвід вивчення різноманітних матеріалів поза межами формального навчання, щоб розвивати мовні навички відповідно до своїх навчальних потреб [3].

Серед найпоширеніших типів сучасних освітніх додатків слід виділити такі [5,8]:

1. Додатки для роботи з флеш-картками дозволяють ефективно запам’ятовувати інформацію, використовуючи картки зі словами, термінами чи визначеннями; часто пропонують ігри та вправи для закріплення матеріалу. Також адаптовані для вивчення іноземних мов, історії, математики та інших предметів. Відтак програма Chegg Prep містить багато безкоштовних карток для вивчення різних предметів.

2. Освітні ігри навчають в ігровій формі, роблячи процес цікавим та інтерактивним; підходять для дітей та дорослих. Вони розвивають різні навички: логіку, пам’ять, увагу, швидкість реакції та інше. Наприклад, за допомогою програми Elevate є можливість тренувати увагу, пам’ять, швидкість реакції.

3. Словники надають доступ до визначення слів, перекладів, синонімів та антонімів; містять аудіовимовлення слів, приклади їх використання та граматичні довідники. Як приклад, за допомогою Merriam-Webster щодня можна легко вивчати та запам’ятовувати нові слова.

4. Програми для вивчення мов пропонують курси з вивчення іноземних мов, що включають граматику, лексику, вимову та розмовну практику. За допомогою них можна використовувати різні способи навчання: відеоуроки,

аудіозаписи, інтерактивні вправи, ігри та інші. Існує безліч таких програм, а саме: Chineasy, Memrise, Duolingo та інші.

5. Спеціальні навчальні програми розроблені для навчання конкретним навичкам чи знанням, наприклад, грі на музичному інструменті, малюванні, програмуванні, містять відеоуроки, покрокові інструкції, інтерактивні вправи та тести. Власне програма Otsimo допомагає дітям з розладами аутичного спектра навчитися спілкуватися.

6. Програми для підготовки до іспитів дозволяють тренуватися у вирішенні тестових завдань, схожих на ті, що будуть на іспиті; з них студенти можуть отримати теоретичні матеріали, підказки та стратегії підготовки.

Розглянемо основні мобільні застосунки, які широко використовуються для вивчення англійської мови.

Duolingo є одним із найпопулярніших застосунків, який використовує метод гейміфікації для підтримки мотивації користувачів [9]. Програма пропонує короткі вправи з перекладом, вибором правильної відповіді, написанням слів і вимовою. Система прогресу, балів і нагород стимулює регулярне навчання. Особливістю Duolingo є адаптивна система, яка враховує успішність користувача та відповідно коригує рівень складності завдань (рис.1.1).

Duolingo структурує уроки як дерево навичок, де "бали навичок" нараховуються за завершення уроків. Також є система життів, яка заохочує користувача правильно відповідати на кожне запитання. Потім користувач може використовувати свої знання для перекладу реального контенту, який оцінюється іншими користувачами.

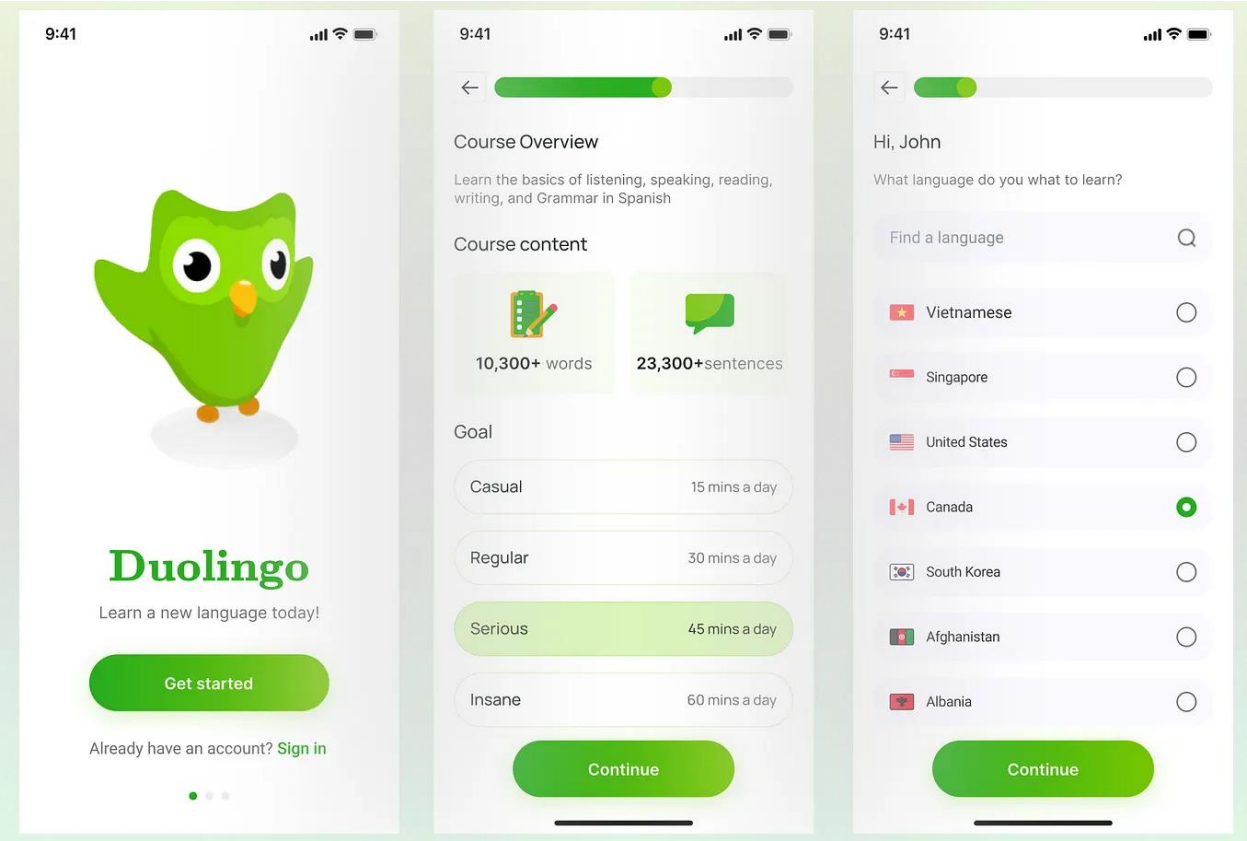

Рисунок 1.1. Сторінки мобільного додатку Duolingo

Memrise – це стосунок орієнтований на запам’ятовування лексики за допомогою мнемонічних технік та повторення [10]. Memrise використовує відеофрагменти з носіями мови, що сприяє кращому засвоєнню реального мовлення та інтонацій. Алгоритм інтервального повторення допомагає зміцнити довготривалу пам’ять. Крім того, застосунок підтримує розділення контенту за тематиками, що дозволяє зосередитися на конкретній сфері (наприклад, бізнес-англійська або технічна термінологія) (рис.1.2).

Додаток інтегрується з основним обліковим записом Memrise.com, але також пропонує офлайн-режим, щоб користувач міг продовжувати навчання навіть без підключення до Інтернету. Memrise також має деякі елементи гейміфікованого навчання, включаючи систему балів.

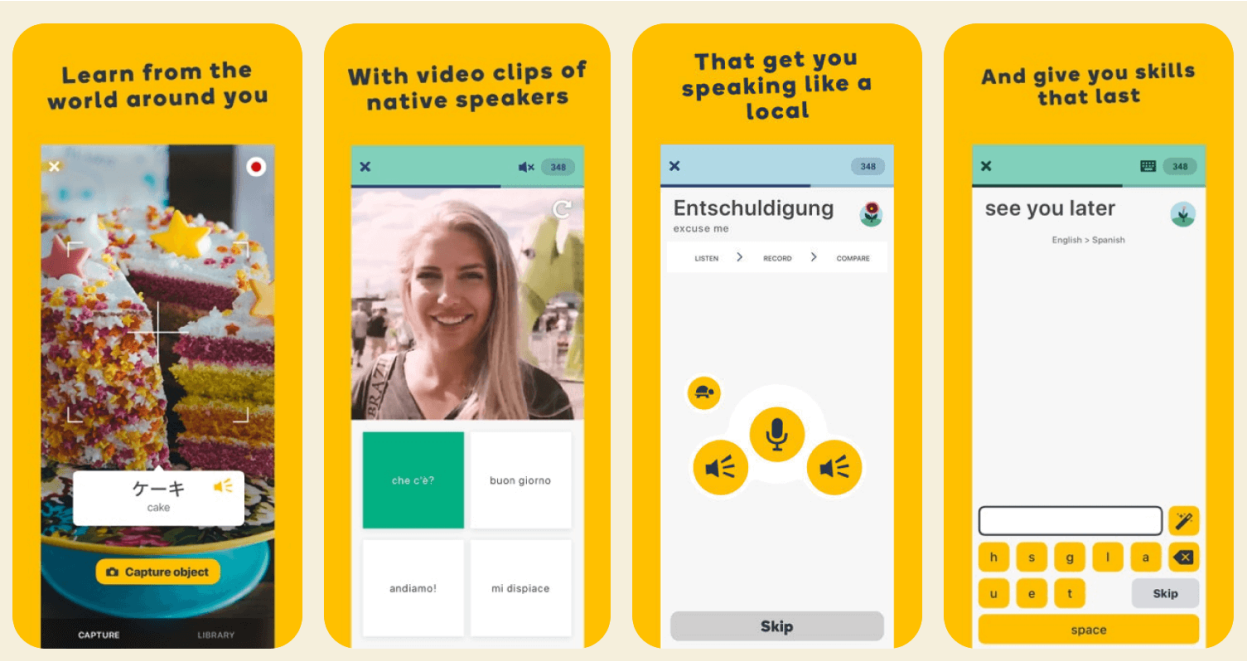

Рисунок 1.2. Сторінки мобільного додатку Memrise

Busuu забезпечує комунікативний підхід до вивчення мови завдяки інтеграції вправ із письма, читання, аудіювання та говоріння [11]. Важливою особливістю є можливість взаємодії з іншими користувачами, зокрема з носіями мови, які можуть перевіряти письмові завдання. Застосунок також пропонує граматичні пояснення, що забезпечує комплексне опанування мови. Busuu включає персоналізований план навчання, що базується на попередньому тестуванні рівня знань (рис.1.3). Цей додаток містить понад 3000 слів і виразів і охоплює широкий спектр тем за допомогою розділів зі словниковим запасом та інтерактивних тестів. Рівні охоплюють від початкового до просунутого, а перші 20 рівнів безкоштовні.

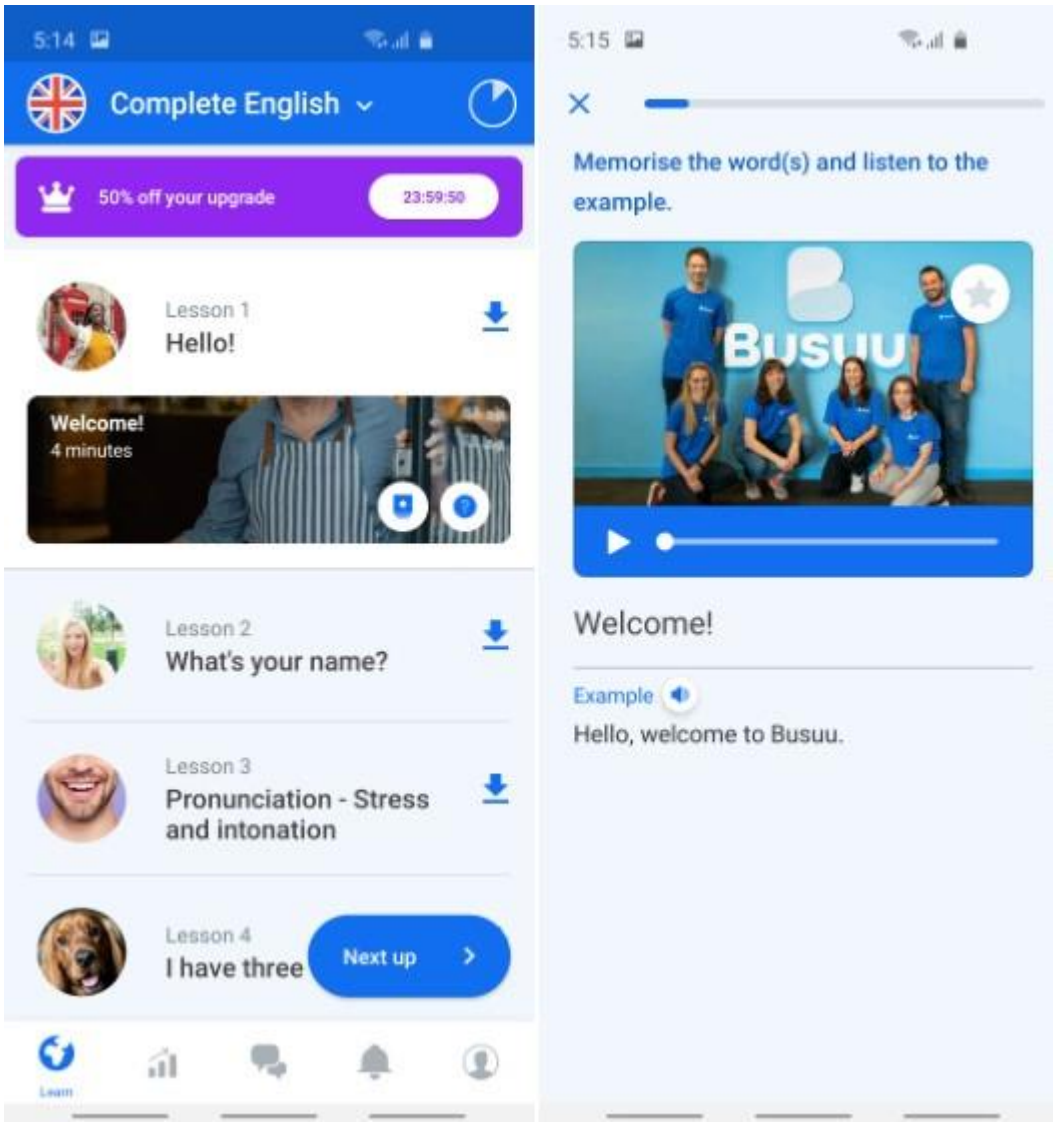

Рисунок 1.3. Сторінки мобільного додатку Busuu

Babbel – це застосунок орієнтований на користувачів, які прагнуть систематичного й ґрунтовного вивчення мови [12]. Уроки Babbel структуровані за тематичними модулями та базуються на комунікативному методі, з акцентом на практичне застосування лексики та граматики у повсякденному спілкуванні. Babbel пропонує контент, адаптований до рівня користувача, та інтегрує функції розпізнавання мовлення для тренування вимови. Додаток підтримує вправи на аудіювання, письмо та говоріння, а також конструктор словникового запасу для розвитку володіння мовою. Додаток також має граматичні вправи, тренажер з вимови та менеджер оглядів

для відстеження прогресу, синхронізацію, а також платну щомісячну та щорічну підписку (рис.1.4).

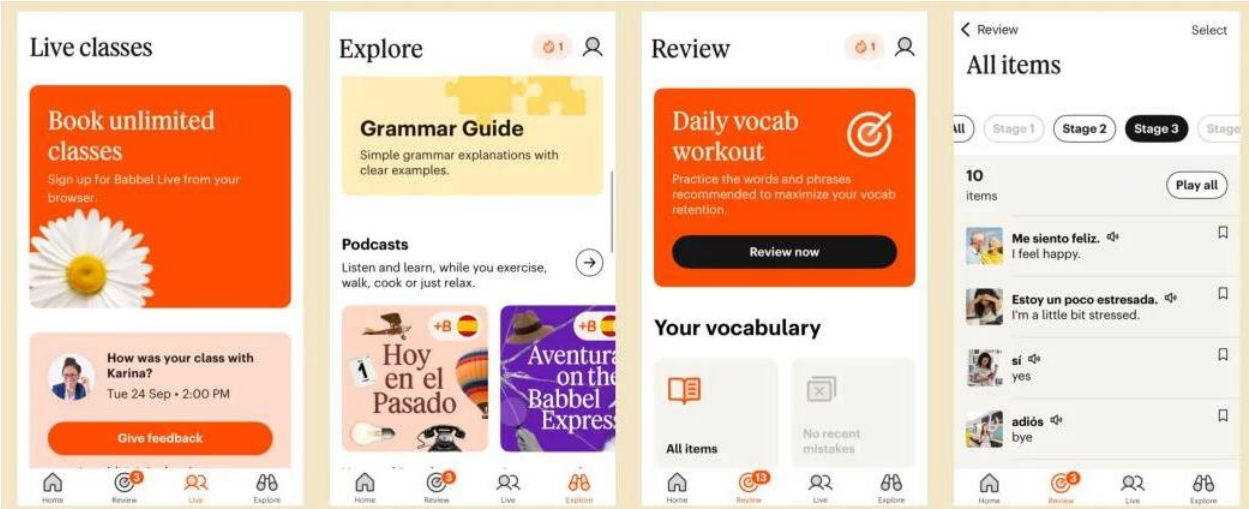

Рисунок 1.4. Сторінки мобільного додатку Babbel

EWA: Learn English є інноваційним застосунком, що поєднує навчання з використанням уривків з фільмів, серіалів і книг [13]. Такий підхід сприяє формуванню навичок сприйняття мови на слух у контексті реального мовного середовища. Застосунок також має функцію вивчення лексики за картками (flashcards) з вбудованим словником та підтримкою повторення (рис.1.5).

Безкоштовна версія програми надає триденний пробний доступ, після чого користувачам пропонується придбати преміум-підписку, яка відкриває доступ до всіх курсів, книг та ігор.

Курси охоплюють різноманітні лексичні та граматичні теми, що включають різні типи вправ: вибір правильної відповіді, складання речень, прослуховування та повторення фраз, чати з персонажами та рейтинг успішності користувачів.

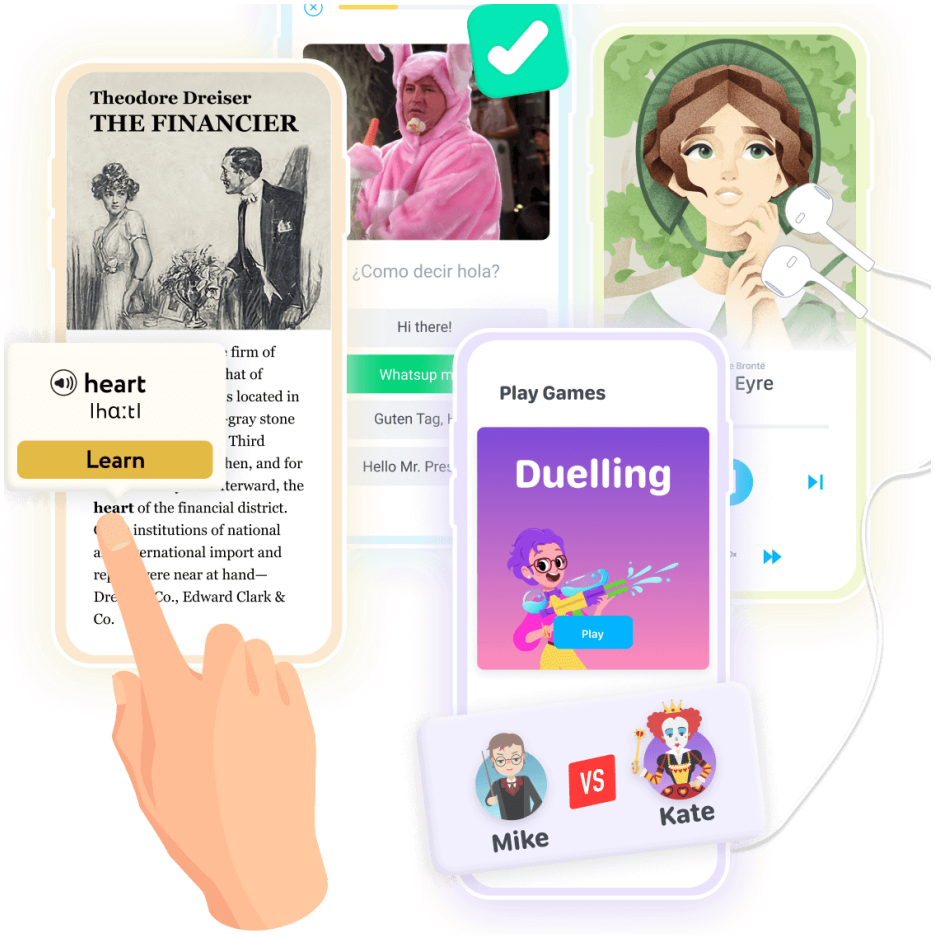

Рисунок 1.5. Сторінки мобільного додатку EWA: Learn English

Результати порівняльного аналізу розглянутих мобільних додатків для вивчення англійської мови з урахуванням потреб IT-фахівців наведені в табл. 1.1.

Таблиця 1.1

Порівняльні характеристики мобільних застосунків для вивчення англійської мови

Застосунок	Основні функції	Переваги	Недоліки з погляду IT-фахівця
Duolingo	Гейміфіковані уроки з базової лексики та граматики	Висока мотивація, простота використання, адаптивне навчання	Відсутність технічної лексики, спрощений контент

Продовження табл. 1.1

Memrise	Вивчення лексики з допомогою мнемонік і відео з носіями мови	Ефективне запам’ятовування, реальна вимова, інтервальне повторення	Лексика загального характеру, немає вузькоспеціалізованих модулів
Busuu	Комплексні курси з граматики, аудіювання, письма, інтерактивні вправи	Соціальна взаємодія з носіями, персоналізовані плани навчання	Обмежена кількість професійно-орієнтованих тем
Babbel	Тематичні модулі для практичного мовлення, граматика	Добре структуровані уроки, інтерація з вимовою	Недостатньо уваги до професійної термінології у сфері ІТ
EWA	Вивчення через уривки з фільмів, книг, серіалів	Живий контекст, велика база мультимедійних матеріалів	Відсутність системності, немає курсів зі спеціалізованою лексикою

Джерело: побудовано автором

Загальний аналіз показує, що мобільні застосунки для вивчення англійської мови пропонують широкі можливості для самостійного навчання завдяки адаптивності, гейміфікації та інтерактивному контенту. Водночас у контексті професійної діяльності, зокрема в сфері інформаційних технологій, виявляється ключовий недолік: відсутність або недостатність вузькоспеціалізованої лексики, що охоплює терміни, скорочення, фразеологізми та ідіоми, типові для ІТ-середовища.

Сучасні ІТ-фахівці потребують не лише базової англійської мови для повсякденного спілкування, а й умінь працювати з технічною документацією,

API-довідниками, коментарями до коду, а також комунікувати у професійних спільнотах. У цьому контексті звичайні навчальні програми, орієнтовані на загальний розвиток мовних навичок, виявляються недостатньо ефективними. Існуючі застосунки зазвичай не передбачають окремих модулів для таких цілей, або їх додавання є складним і малопрактичним для більшості користувачів.

Таким чином, виникає необхідність у створенні спеціалізованих мобільних застосунків, які б орієнтувалися на термінологію IT-галузі та включали сценарії комунікації в IT-командах (наприклад, обговорення задач, рев'ю коду, презентація рішень). Такі застосунки повинні мати модулі з технічною лексикою, англомовною документацією та цитатами з типових технічних текстів, зокрема, інтерфейси GitHub, Stack Overflow, документацію Python, React тощо.

Розробка подібного роду застосунків сприятиме покращенню якості підготовки IT-фахівців і дозволить ефективніше реалізувати вимоги міжнародної співпраці, де англійська мова відіграє роль основного засобу професійного спілкування.

1.3. Формулювання вимог для мобільного застосунку з вивчення англійської мови

Розробка освітнього програмного забезпечення повинна враховувати особливості такого типу додатків, щоб забезпечити ефективний та зручний процес навчання. Під час розробки необхідно врахувати такі характеристики [14]:

1. Застосунок повинен дозволяти користувачеві вибирати свій рівень знань, інтереси, що дозволить адаптувати навчальний матеріал до індивідуальних потреб та уподобань, роблячи навчання більш ефективним та мотивуючим.

2. Застосунок має містити різноманітний контент та інтерактивні елементи; такий підхід робить процес навчання візуальним, захопливим та цікавим.

3. Впровадження гейміфікованих елементів, таких як досягнення, бали, рейтинги, винагороди та рівні, сприяє залученню та утриманню користувачів, роблячи навчання більш захопливим та стимулюючим.

4. У застосунку слід забезпечувати можливість відстежувати прогрес навчання, записувати результати тестів, аналізувати помилки та надавати рекомендації, що допоможе користувачам контролювати свій прогрес та визначати області, які потребують додаткової роботи.

5. Навчальні застосунки повинні забезпечувати доступ до матеріалів навіть за відсутності підключення до Інтернету, дозволяючи користувачам продовжувати навчання в будь-який час і в будь-якому місці.

6. Навчальний застосунок повинен бути доступним на різних платформах.

7. Застосунок повинні мати інтуїтивно зрозумілий та простий інтерфейс, що забезпечує швидкий доступ до контенту та функцій без зайвих кроків, що є важливим для комфортного та продуктивного навчання.

Перейдемо до визначення функціональних та нефункціональні вимог до розроблюваного освітнього додатку для вивчення англійської мови ІТ-фахівцями.

Функціональні вимоги є однією з ключових складових технічної специфікації програмного забезпечення, що визначає, які саме функції має виконувати система для задоволення потреб користувача або виконання заданих бізнес-процесів. Згідно з міжнародними стандартами в галузі інженерії програмного забезпечення, функціональні вимоги описують зовнішню поведінку системи, тобто реакцію програмного продукту на певні вхідні дані, події або запити [15].

Ці вимоги формулюються на підставі аналізу предметної області, очікувань замовника та кінцевих користувачів і зазвичай охоплюють набір

операцій, які система повинна підтримувати. Функціональні вимоги концентруються виключно на що повинна робити система, а не на як вона має це реалізувати. Саме на основі функціональних вимог проєктуються основні модулі системи, визначаються сценарії взаємодії користувача з інтерфейсом, формуються технічні завдання для команди розробників та здійснюється валідація програмного продукту під час тестування.

У таблиці 1.2 наведені функціональні вимоги для освітнього застосунку з вивчення англійської мови ІТ-фахівцями.

Таблиця 1.2

Функціональні вимоги до мобільного застосунку з вивчення англійської мови

№	Категорія	Опис вимоги
1	Реєстрація та автентифікація	Система має дозволяти реєстрацію, підтримувати автентифікацію користувача.
2	Вивчення матеріалу	Система повинна надавати користувачу доступ до навчальних матеріалів, можливість переглядати контент, знайомитись з матеріалом уроку та вивчати нові терміни.
3	Тестування	Система має забезпечувати проходження тестів за вивченим матеріалом.
4	Редагування контенту	Система повинна забезпечувати можливість оновлення контенту, як розробниками, так і користувачами.
5	Аналітика та звітність	Система повинна зберігати прогрес користувача у вивченні англійської мови, зберігати результати тестування з зазначенням зроблених помилок.

Джерело: побудовано автором

Нефункціональні вимоги є невід’ємною складовою процесу розробки програмного забезпечення та визначають якісні характеристики інформаційної системи, що не стосуються її безпосередньої функціональної поведінки. На

відміну від функціональних вимог, які описують конкретні дії або послуги, що має забезпечувати система, нефункціональні вимоги визначають обмеження та умови, за яких ці функції повинні реалізовуватись.

Нефункціональні вимоги формулюють очікування щодо таких властивостей програмного продукту, як надійність, продуктивність, масштабованість, безпека, зручність інтерфейсу, доступність, відповідність стандартам тощо. Ці вимоги суттєво впливають на архітектуру системи, вибір технологій, методів тестування та обслуговування, а також на загальну якість програмного забезпечення з погляду кінцевого користувача [15].

У таблиці 1.3 наведені основні нефункціональні вимоги для освітнього застосунку з вивчення англійської мови.

Таблиця 1.3.

Нефункціональні вимоги до мобільного застосунку з вивчення англійської мови

№	Категорія	Опис вимоги
1	Продуктивність та масштабованість	Час запуску Застосунку не повинен перевищувати 3 секунд на пристроях з середнім рівнем продуктивності. Застосунок повинен підтримувати масштабування контенту, не втрачаючи продуктивності. Початковий обсяг встановленого додатку не повинен перевищувати 100 МБ.
2	Безпека даних	Дані користувачів повинні зберігатися в зашифрованому вигляді. Аутентифікація користувачів має здійснюватися з використанням сучасних методів безпечного входу.
3	Кросплатформеність	Застосунок має коректно працювати на платформах Android (версії 8.0 і вище) та iOS (версії 13.0 і вище) та підтримувати різні розміри екранів (телефони, планшети) та орієнтацію (портретна, альбомна).

Продовження таблиці 1.3

4	Юзабіліті	Інтерфейс повинен бути інтуїтивно зрозумілим, доступним для користувачів без спеціальних технічних знань. Час навчання нового користувача (навігація, виконання завдання) має бути меншим за 5 хвилин.
5	Надійність роботи	Система повинна зберігати прогрес користувача навіть у разі втрати мережевого з'єднання, з подальшою синхронізацією при відновленні доступу. Основні функції (перегляд вивченого контенту, виконання вправ, збереження результатів) мають бути доступні без підключення до Інтернету.
6	Вимоги до підтримки	Архітектура застосунку повинна бути модульною та розширюваною для можливості додавання нових функцій без зміни існуючих.

Джерело: побудовано автором

Сформульовані функціональні та нефункціональні вимоги забезпечують якість, безпеку та зручність використання мобільного застосунку для вивчення англійської мови ІТ-фахівцями, а також формують чітке уявлення розробника щодо архітектурних рішень і технічної реалізації.

Висновки до розділу 1

В сучасному світі мобільні застосунку використовуються в багатьох областях, зокрема і для навчання користувачів. Такі освітні застосунки отримали широке розповсюдження.

Розробка мобільних освітніх додатків повинна враховувати їх особливості. Розробка повинна бути спрямована на задоволення індивідуальних потреб користувачів, підтримку їхньої мотивації та забезпечення ефективного навчання. Завдяки цим функціям сучасні освітні застосунки можуть значно спростити та збагатити процес навчання, зробивши його доступним, захопливим та ефективним для користувачів різного віку та рівня знань.

Для ІТ-фахівців для вивчення іноземної мови потрібно створювати спеціалізовані програмні продукти, оскільки ця область має свою особливу лексику та сталі вирази. Розширення існуючих застосунків не завжди є можливим, оскільки це може бути платною послугою, а поява нових термінів в області інформаційних технологій відбувається постійно. Тому потрібно віддавати перевагу створенню спеціалізованих застосунків.

Сформульовані вимоги щодо функцій застосунку, його продуктивності, надійності, масштабованості є основою для подальшого проєктування застосунку.

РОЗДІЛ 2
ПРОЄКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ "АНГЛІЙСЬКА ДЛЯ ПРОГРАМІСТІВ"

2.1. Архітектура мобільного застосунку

Архітектура і розповсюдження мобільних платформ значною мірою визначають вибір технологій у розробці мобільних додатків. Зараз основними платформами для мобільних застосунків є Android та iOS. Вони забезпечують найбільшу частку ринку та мають критичне значення для проєктувальників мобільних додатків.

За даними за період травня – червня 2025 року, глобальна частка Android становить 72 %, тоді як iOS – близько 27 % [16]. Хоча конкретне розповсюдження тих чи інших ОС залежить від регіону, однак для України тенденція зберігається (рис.2.1).

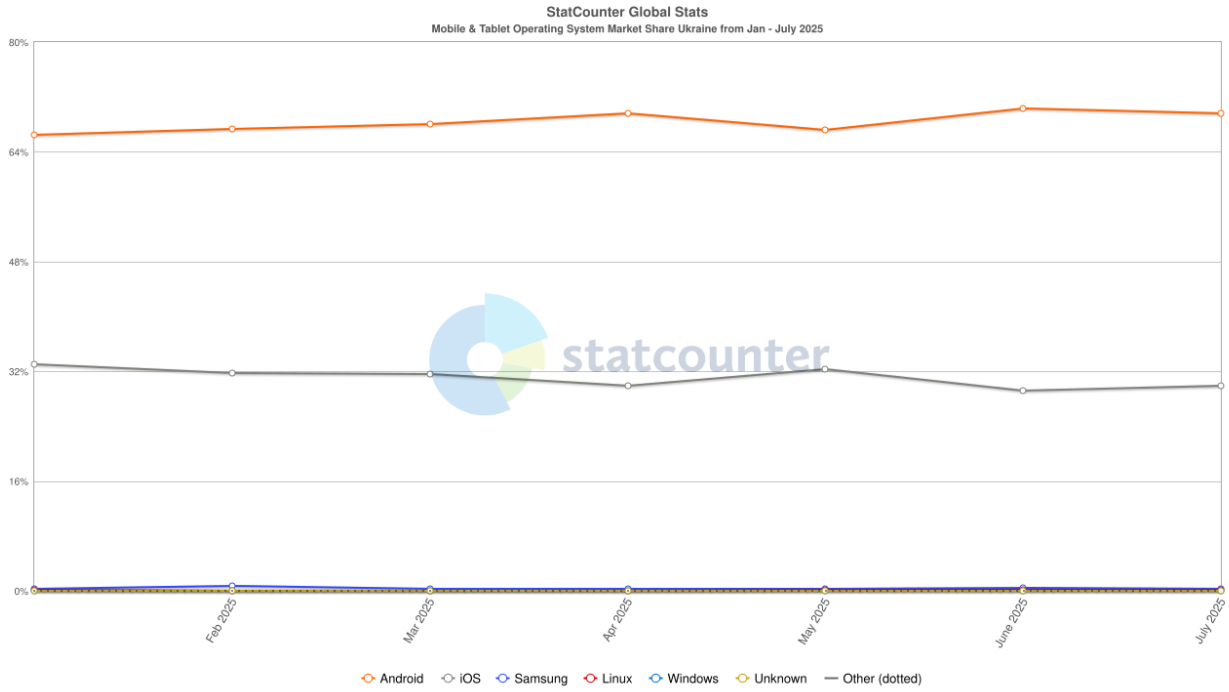

Рисунок 2.1. Розповсюдження ОС для мобільних пристроїв в Україні у 2025 році за даними Global Stats

Таким чином, при розробці мобільного застосунку слід зосередитися на платформах Android та iOS, які є найбільш розповсюдженими.

Домінування операційної системи Android для мобільних пристроїв обумовлено її відкритістю, широкою апаратною підтримкою та гнучкістю інтеграції з різними видами пристроїв. Android є операційною системою з відкритим вихідним кодом, яка базується на модифікованому ядрі Linux та надає багаторівневу архітектуру для розробки, розгортання та виконання мобільних застосунків.

Система Android побудована за модульним принципом і має чотиришарову архітектуру [17,18]:

1. Ядро Linux (Linux Kernel). В основі Android лежить ядро Linux, яке виконує функції керування пам’яттю, процесами, мережею, системою безпеки та файловими системами. Ядро також забезпечує апаратну абстракцію для взаємодії з апаратними компонентами (камерою, дисплеєм, GPS тощо).

2. Бібліотеки та середовище виконання (Native Libraries & Android Runtime). Даний рівень містить набір бібліотек C/C++ (наприклад, libc, WebKit), які використовуються додатками. Також тут розміщено середовище виконання Android Runtime (ART). ART здійснює попередню компіляцію байт-коду (AOT-компіляція) у машинний код для покращення продуктивності.

3. Фреймворк (Application Framework). Android-фреймворк надає API високого рівня, що дозволяє розробникам отримувати доступ до системних сервісів, таких як управління активностями, службами, контент-провайдерами, ресурсами та сповіщеннями. Фреймворк підтримує концепції інкапсуляції та повторного використання коду.

4. Рівень додатків (Applications). На найвищому рівні розташовані застосунки – як системні (наприклад, "Налаштування", "Телефон", "Файли"), так і сторонні, розроблені користувачами або компаніями. Усі додатки працюють у відокремлених середовищах, що забезпечує ізоляцію процесів і безпечне виконання.

Android-додатки розробляються за допомогою мови Java або Kotlin, а також можуть використовувати інші мови через JNI (Java Native Interface). Кожен застосунок компілюється у формат .apk (Android Package), який містить байт-код, ресурси, маніфест і необхідні бібліотеки [19].

Основною структурною одиницею додатку є компоненти:

1. Activity – екран інтерфейсу користувача.
2. Service – фоновий процес, що може виконуватись без взаємодії з UI.
3. BroadcastReceiver – обробка системних або користувацьких широкомовних повідомлень.
4. ContentProvider – механізм доступу до структурованих даних, які можуть бути спільно використані іншими додатками.

Кожен застосунок запускається у власному віртуальному середовищі ("пісочниці") з унікальним ID користувача (UID) та ізольованим доступом до ресурсів, що запобігає несанкціонованому втручанню інших додатків.

Усі компоненти зв’язуються через інтенти (intents) – механізми передачі повідомлень, які дозволяють ініціювати дії, запускати інші компоненти або передавати дані, що забезпечує модульність і гнучкість архітектури.

Життєвий цикл застосунку на платформі Android являє собою послідовність станів, через які проходить мобільний застосунок від моменту його запуску до завершення роботи. Розуміння особливостей цього циклу має важливе значення для ефективного управління ресурсами системи, збереження даних користувача, забезпечення стабільності та зручності у використанні застосунку. Він зображений на рис.2.2.

Основною структурною одиницею Android-застосунку є Activity – компонент, що відповідає за один екран інтерфейсу користувача. Життєвий цикл кожної Activity визначається низкою методів зворотного виклику, які викликаються системою Android автоматично відповідно до взаємодії користувача або змін у стані системи.

До основних методів життєвого циклу належать [17,19]:

– onCreate(), який викликається при створенні Activity. Тут ініціалізується інтерфейс користувача, виконується завантаження ресурсів, налаштовуються обробники подій.

– onStart(), який викликається після onCreate() або коли Activity стає видимою після приховання.

– onResume(), який викликається, коли Activity отримує фокус і стає інтерактивною. У цьому стані користувач безпосередньо взаємодіє із застосунком.

– onPause(): викликається перед втратою фокусу, наприклад, при переході до іншої Activity або при виведенні діалогового вікна. У цьому методі доцільно зберігати критичні дані або зупиняти анімації та інші ресурсоємні процеси.

– onStop(), який викликається, коли Activity більше не видима. Можна зберігати стан застосунку, зупиняти фонові процеси тощо.

– onRestart(), який викликається перед повторним запуском Activity, яка була зупинена.

– onDestroy(), який викликається перед остаточним знищенням Activity. Тут звільняються всі зайняті ресурси та відбувається фіналізація.

Важливою особливістю Android-платформи є те, що система самостійно управляє життєвим циклом Activity, зважаючи на обмеженість ресурсів пристрою. Наприклад, у разі нестачі оперативної пам’яті система може знищити неактивні Activity без виклику onDestroy().

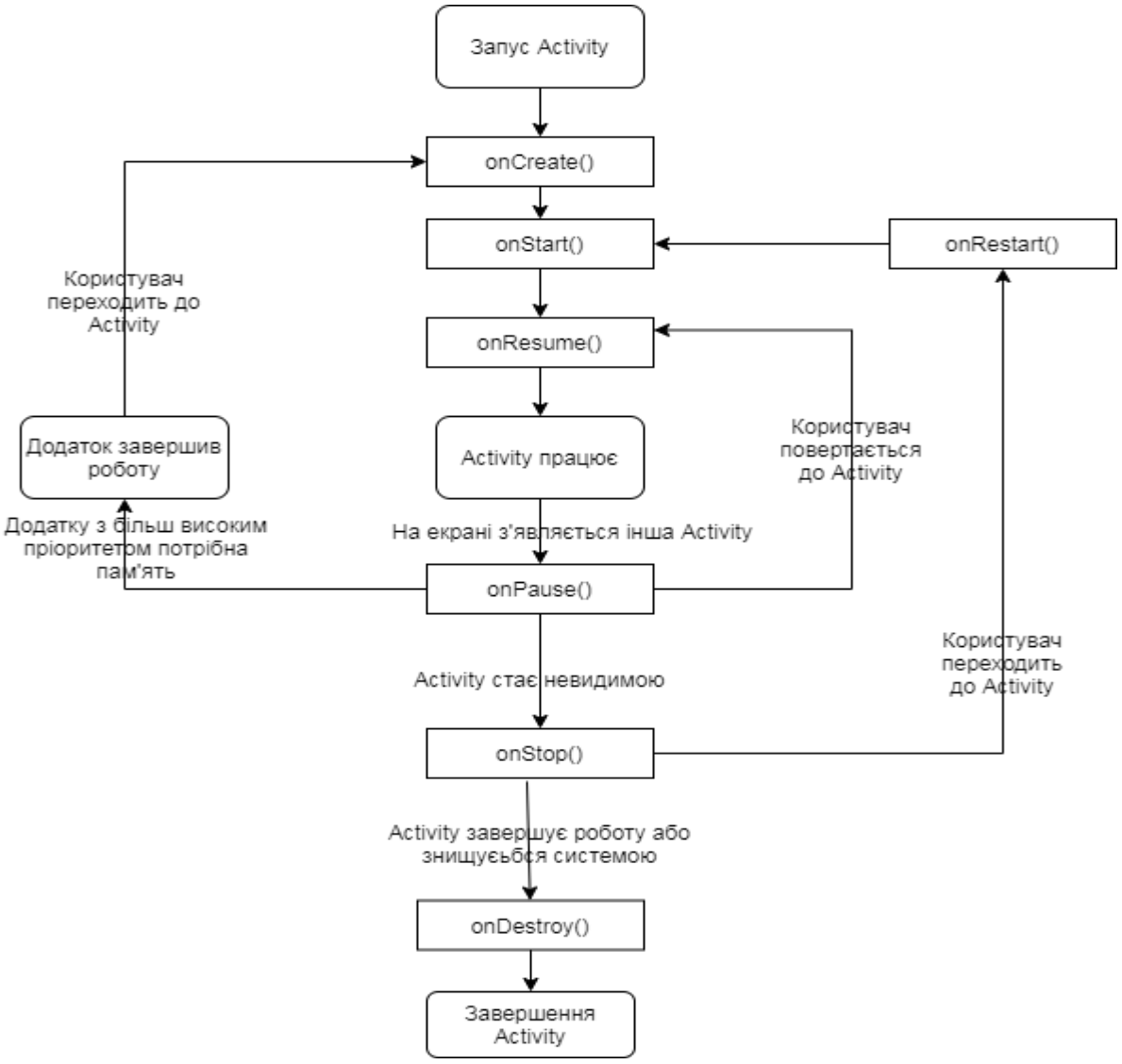

Рисунок 2.2 – Життєвий цикл застосунку на Android [23]

Платформа Android реалізує багаторівневу, модульну архітектуру. Така структура сприяє ефективному розмежуванню функцій, масштабованості та безпеці. Розробка та функціонування мобільних застосунків у цій екосистемі забезпечуються компонентно-орієнтованою моделлю, гнучкою системою повідомлень та захищеним середовищем виконання.

Операційна система iOS є власницькою мобільною платформою, розробленою компанією Apple Inc., яка використовується на пристроях iPhone, iPad та iPod Touch. Вона вирізняється високим рівнем оптимізації, безпеки та тісної інтеграції апаратного й програмного забезпечення. Завдяки жорсткому контролю з боку Apple та обмеженому набору підтримуваних пристроїв, iOS

забезпечує стабільну продуктивність і передбачувану поведінку додатків [17,18].

Архітектура iOS побудована за принципом ієрархії програмних рівнів. Вона має чотиришарову структуру, яка забезпечує ізоляцію, повторне використання компонентів та контроль доступу до системних ресурсів [17,18].

1. Core OS (ядро операційної системи) – це найнижчий рівень, який містить ядро системи, що базується на XNU (гібрид ядра, що поєднує мікроядро Mach із компонентами BSD UNIX). Тут також розміщено драйвери, засоби безпеки (наприклад, Secure Enclave), енергозбереження, управління пам’яттю та мережеві інтерфейси.

2. Core Services (базові служби) представляє системний рівень, який надає фундаментальні системні служби, включаючи Core Foundation, Foundation Framework, Core Data (для роботи з БД), iCloud Storage, а також API для роботи з потоками, файлами, геолокацією, мережами тощо. У цьому шарі закладено логіку обробки подій та управління даними.

3. Media (мультимедійний рівень) відповідає за обробку графіки, відео, аудіо та анімацій. Сюди входять фреймворки Core Animation, AVFoundation, Core Graphics, Metal (для роботи з GPU), а також API для камери та мікрофона. Завдяки цим компонентам реалізуються високопродуктивні мультимедійні інтерфейси.

4. Cocoa Touch (рівень інтерфейсу) є найвищим рівнем архітектури. Він забезпечує роботу з елементами користувацького інтерфейсу та взаємодією з користувачем. Включає SwiftUI, що відповідає за обробку жестів, екранів, контролерів перегляду, кнопок, а також систему повідомлень і управління життєвим циклом додатка.

Застосунки для iOS створюються за допомогою мов Swift або Objective-C із використанням середовища розробки Xcode. Всі застосунки функціонують у так званому "пісочному середовищі" (sandbox), яке ізолює їх один від одного і від системних ресурсів, забезпечуючи високий рівень безпеки.

Основними складовими iOS-додатків є [17,18]:

– View Controllers – контролери, що керують логікою та інтерфейсом певного екрану;

– Views – графічні компоненти інтерфейсу (кнопки, поля вводу, таблиці тощо);

– Model – структура даних і логіка обробки (реалізується через патерни MVC або MVVM);

– Delegates та Notifications – механізми взаємодії між компонентами;

– AppDelegate / SceneDelegate керують життєвим циклом додатку.

Функціонування додатку в iOS базується на подійно-орієнтованій моделі. Події (жести, повідомлення системи, сигнали сенсорів) обробляються через головний цикл подій (event loop), який координує взаємодію користувача з графічним інтерфейсом (рис.2.3).

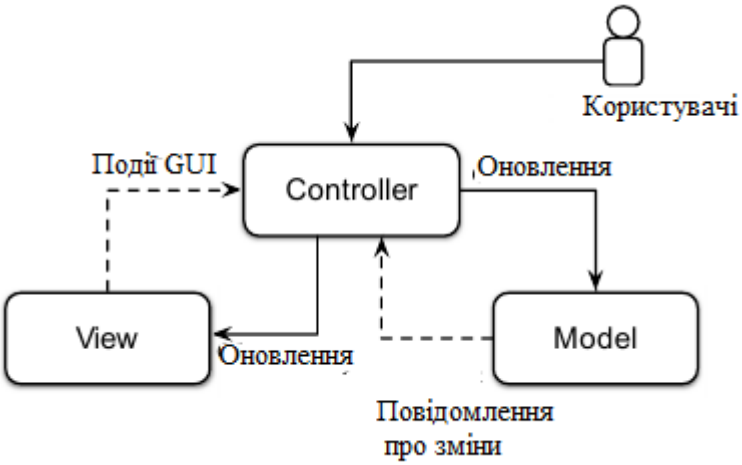

Рисунок 2.3 – Cocoa MVC модель [15]

Всі додатки для iOS повинні проходити модерацію Apple перед публікацією в App Store. Це забезпечує дотримання стандартів якості, безпеки, продуктивності та конфіденційності. Система сертифікатів, цифрових підписів і контроль дозволів запобігає виконанню шкідливого коду та несанкціонованому доступу до даних користувача.

Операційна система iOS побудована за модульною, багаторівневою архітектурою. Усі застосунки функціонують у захищеному середовищі та

створюються із суворим дотриманням інженерних стандартів Apple. Завдяки централізованій екосистемі, передбачуваній поведінці API та суворим вимогам до якості, iOS забезпечує стабільну платформу для розробки, розгортання та використання мобільних застосунків.

У сучасній практиці розробки мобільних застосунків виділяють кілька основних типів, серед яких найбільш поширеними є нативні (native) та кросплатформені (cross-platform) застосунки. Вибір між цими підходами залежить від низки технічних, економічних та функціональних чинників, включно з цілями проєкту, аудиторією, ресурсами розробки та вимогами до продуктивності [19, 20].

Нативні мобільні застосунки створюються окремо для кожної цільової операційної системи за допомогою спеціалізованих мов програмування та інструментів. Зокрема, для платформи Android зазвичай використовують Java або Kotlin у середовищі Android Studio, тоді як для iOS застосовуються Swift або Objective-C у середовищі Xcode. Нативний підхід дозволяє максимально використовувати можливості конкретної операційної системи, забезпечує високу продуктивність, кращу інтеграцію з апаратним забезпеченням пристрою, повноцінний доступ до API та дотримання специфічних стандартів дизайну. Основними недоліками нативного підходу є висока вартість розробки, необхідність підтримки окремих кодових баз для кожної платформи та більші витрати часу при масштабуванні [20].

Кросплатформені застосунки, навпаки, дозволяють створювати єдину кодову базу, яка може бути скомпільована для декількох платформ (переважно Android та iOS). Це досягається за рахунок використання спеціалізованих фреймворків, таких як .NET MAUI, Flutter, React Native або Xamarin.

Основні переваги кросплатформеного підходу включають зменшення витрат на розробку, швидше створення і зручність супроводу. При цьому, хоча кросплатформені застосунки можуть поступатися нативним у продуктивності чи специфічності реалізації UI/UX, сучасні фреймворки дозволяють досягати високого рівня оптимізації, що задовольняє потреби більшості проєктів.

Зважаючи на мету розробки, а саме створення мобільного застосунку для вивчення англійської мови, обґрунтованим є вибір кросплатформеного підходу. По-перше, цільова аудиторія охоплює користувачів як Android-, так і iOS-пристроїв, що вимагає мультиплатформної доступності. По-друге, розроблювальний проєкт немає високих вимог щодо ресурсів та оптимізації. Таким чином, кросплатформене рішення дозволяє досягти балансу між якістю, продуктивністю, масштабованістю та економічною доцільністю.

Архітектура кросплатформеного мобільного застосунку являє собою структурну модель, яка забезпечує її функціонування на різних мобільних платформах з використанням єдиного програмного коду. Основною метою побудови такої архітектури є забезпечення повторного використання коду, оптимізація витрат на розробку та супровід, а також підтримка узгодженого інтерфейсу користувача на різних пристроях [20].

На концептуальному рівні архітектура кросплатформеного застосунку зазвичай базується на багаторівневій моделі, яка включає такі основні шари:

1. Шар презентації (Presentation Layer) відповідає за взаємодію з користувачем, відображення графічного інтерфейсу та обробку подій. У контексті кросплатформеного підходу цей шар часто реалізується за допомогою таких фреймворків, як Xamarin.Forms або .NET MAUI, що дозволяють створювати інтерфейси з єдиним кодом для кількох платформ.

2. Шар бізнес-логіки (Business Logic Layer) містить основні алгоритми, правила та обробку даних, пов’язані з логікою функціонування застосунку. Саме на цьому рівні реалізуються функціональні вимоги, незалежно від платформи. Для забезпечення модульності і тестованості логіка структурується у вигляді окремих сервісів, менеджерів або компонентів з використанням патернів, таких як MVVM (Model-View-ViewModel).

3. Шар доступу до даних (Data Access Layer) забезпечує взаємодію з джерелами даних: локальними базами (наприклад, SQLite), віддаленими API (через HTTP/REST), хмарними сервісами або файловими сховищами.

Комунікація на цьому рівні відбувається з використанням абстракцій, що дозволяє змінювати джерела даних без впливу на інші шари.

4. Шар платформи (Platform Abstraction Layer) реалізує інтерфейси для доступу до функцій, специфічних для конкретної платформи (наприклад, камера, GPS, Bluetooth). Через використання залежностей і механізмів інжекції (Dependency Injection) забезпечується можливість виклику нативних API без порушення цілісності кросплатформенного коду.

Ключовим принципом проєктування архітектури такого типу є відокремлення логіки додатку від інтерфейсних і платформозалежних компонентів, що дозволяє значно знизити витрати на підтримку та адаптацію до нових платформ. Крім того, широке застосування шаблонів проєктування (MVVM, Repository, Dependency Injection) сприяє зростанню гнучкості, тестованості та масштабованості застосунку.

Таким чином, архітектура кросплатформеного мобільного застосунку є результатом поєднання програмних практик, що забезпечують незалежність від конкретної операційної системи, підтримку високого рівня повторного використання коду та ефективне управління складністю проєкту.

2.2. Проєктування поведінки додатку та бази даних

Перехід від етапу визначення вимог до об’єктно-орієнтованого проєктування мобільного застосунку передбачає деталізацію функціональних і нефункціональних потреб користувача у вигляді структурованих моделей. Отримані вимоги будуть трансформовані в систему взаємопов’язаних діаграм, які репрезентують поведінку програмної системи. Об’єктно-орієнтоване проєктування створює основу для подальшої реалізації мобільного застосунку з урахуванням принципів модульності та повторного використання коду [21,22].

Діаграма варіантів використання (use case diagram) для мобільного застосунку з вивчення англійської мови відображає основні сценарії взаємодії

користувача з системою та дозволяє структурувати функціональні вимоги до програмного забезпечення. Основним актором є Користувач, який взаємодіє із застосунком (рис.2.4).

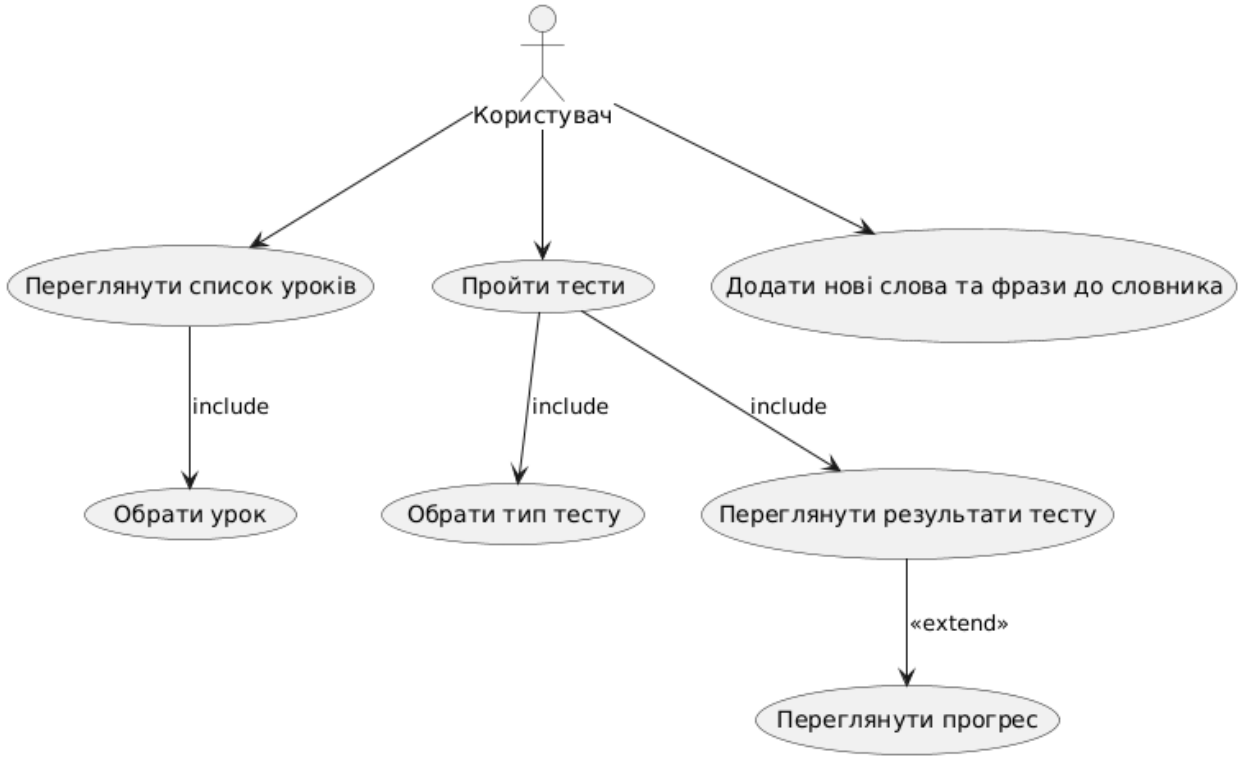

Рисунок 2.4. Діаграма варіантів використання для мобільного застосунку

Опис основних варіантів використання наведений у табл. 2.1-2.5.

Таблиця 2.1

Варіант використання "Переглянути список уроків"

Короткий опис	**Варіант використання дозволяє переглянути список уроків**
Актори	Користувач
Передумова	Користувач зайшов у мобільний додаток
Основний потік	Користувач натискає кнопку "Lessons". На екрані з’являється список уроків
Включення	Варіант використання "Обрати урок"

Таблиця 2.2

Варіант використання "Обрати урок"

Короткий опис	**Варіант використання дозволяє обрати урок з вивчення англійської мови**
Актори	Користувач
Передумова	Користувач перейшов у вікно "Уроки"
Основний потік	Користувач переглядає список уроків та натискає кнопку з ім’ям відповідного уроку. На екрані з’являється матеріал обраного уроку (слова, що потрібно вивчити)
Зв’язок	Включений у варіант використання " Переглянути список уроків"

Таблиця 2.3

Варіант використання "Пройти тести"

Короткий опис	**Варіант використання забезпечує роботу з тестами**
Актори	Користувач
Передумова	Користувач зайшов у мобільний додаток "
Основний потік	Користувач натискає кнопку "Тест". Користувач обирає тип тесту
Включення	Варіант використання "Обрати тип тесту"
Включення	Варіант використання "Проглянути результати тесту"

Таблиця 2.4

Варіант використання "Обрати тип тесту"

Короткий опис	**Варіант використання дозволяє пройти тестування**
Актори	Користувач

Продовження таблиці 2.4

Передумова	Користувач перейшов у вікно "Робота з тестами"
Основний потік	Користувач переглядає список типів тестів та обирає відповідний. На екрані з’являється тест, який можна пройти
Зв’язок	Включений у варіант використання "Пройти тест"

Таблиця 2.5

Варіант використання "Переглянути результати тесту"

Короткий опис	**Варіант використання дозволяє переглянути результати тестування**
Актори	Користувач
Передумова	Користувач перейшов у вікно "Робота з тестами"
Основний потік	Користувач натискає кнопку "Перегляд результатів тестування". На екрані з’являються результати пройдених тестів
Зв’язок	Включений у варіант використання "Пройти тест"
Розширення	Варіант використання "Перегляд прогресу"

Таблиця 2.6

Варіант використання "Додати нові слова та фрази до словника"

Короткий опис	**Варіант використання дозволяє керувати словником**
Актори	Користувач
Передумова	Користувач натиснув кнопку "Словник"
Основний потік	Користувач вводить слово, переклад, приклад, малюнок та натискає кнопку "Додати"

Перейдемо до деталізацію основних варіантів використання для мобільного застосунку.

Варіант використання "Переглянути список уроків" опишемо за допомогою діаграми активності (рис.2.5).

Користувач мобільного застосунку вибирає перехід до списку уроків. На екрані з’являється список доступних для даного користувача уроків. Користувач може обрати наступний урок або переглянути вже пройдені. Після вибору уроку на екрані користувача з’являється вміст уроку.

Рисунок 2.5. Діаграма активності для варіанту використання "Переглянути список уроків"

Далі опишемо варіант використання "Пройти тест". Для цього буде використовуватись діаграма послідовностей, показати взаємодію між об’єктами (користувачем і системними компонентами), які беруть участь у тестуванні.

Загалом, послідовність взаємодії користувача з системами для проходження тесту можна представити таким чином:

1. Користувач запускає тест.
2. Система надсилає питання.

3. Користувач дає відповідь.

4. Система перевіряє її; даний процес повторюється до тих пір, поки в тесті є питання.

5. Наприкінці система показує результат.

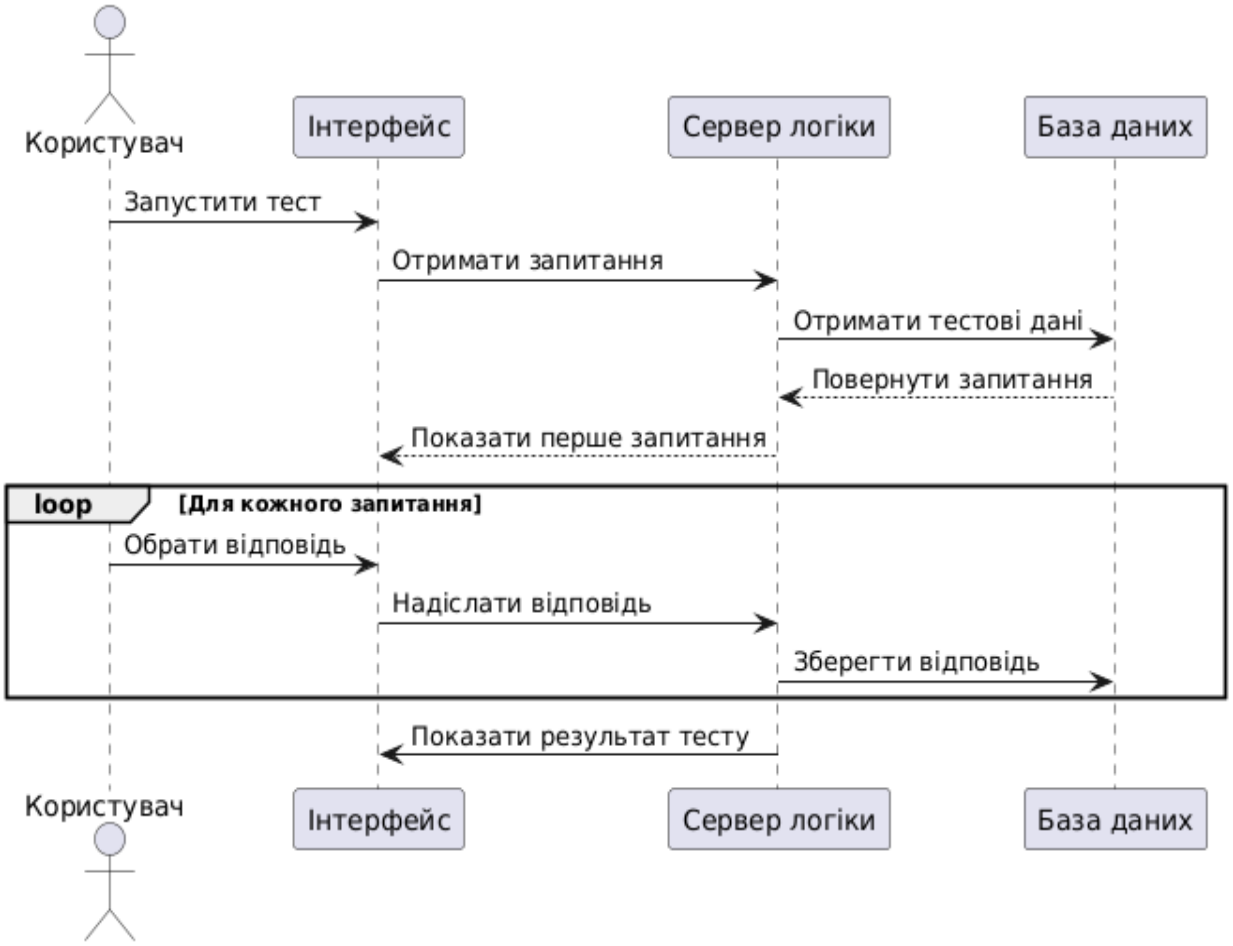

Рисунок 2.6. Діаграма послідовностей для варіанту використання "Пройти тест"

Також деталізуємо варіант використання "Додавання слів та фраз до словника". Оскільки його виконання передбачає просту послідовність дій користувача без складної взаємодії об’єктів, то можна використати діаграму активності (рис.2.7).

1. Користувача заходить у розділ "Словник".

2. Користувач додає слово або фразу в словник та її переклад.

3. Якщо у користувача є картинка, то він може також її завантажити.

4. Користувач натискає кнопку "Додати" і завершує роботу зі словником

Рисунок 2.7. Діаграма активності для варіанту використання "Додавання слів та фраз до словника"

Розроблений проєкт мобільного застосунку буде використовуватись для реалізації програмної системи.

Перейдемо до проєктування сховища даних для мобільного застосунку. Розроблена структура повинна забезпечувати як ефективне управління навчальним контентом, так і можливість персоналізації навчального процесу. Пропонується здійснювати розподілене зберігання даних: в БД буде зберігатися інформація про діяльність користувача (прогрес в проходженні уроків, результати тестування), а самі тести будуть зберігатися в файлі JSON формату.

Такий підхід є доцільним у контексті мобільного середовища, оскільки дозволяє зберігати структури змінної складності без необхідності змінювати

схему бази даних при кожному оновленні тестів. Крім того, зчитування тестів у форматі JSON спрощує їх обробку на стороні клієнтського застосунку та забезпечує можливість швидкої адаптації інтерфейсу до різних типів завдань (наприклад, вибір однієї відповіді, множинний вибір, відповідність тощо).

У запропонованій архітектурі бази даних передбачено окремі сутності для користувачів, слів та фраз, уроків, тестів та результатів проходження, що відповідає принципам нормалізації та розділення обов'язків між функціональними модулями.

Інформація про слова та фрази включає текст, переклад та шлях до зображення. Зберігання зображень не в самій базі, а як посилань на локальні або хмарні файли є обґрунтованим рішенням, оскільки дозволяє значно зменшити обсяг бази даних і покращити продуктивність мобільного застосунку. Це також полегшує оновлення мультимедійного контенту без потреби внесення змін до структури БД.

ER-схема БД представлена на рис.2.8.

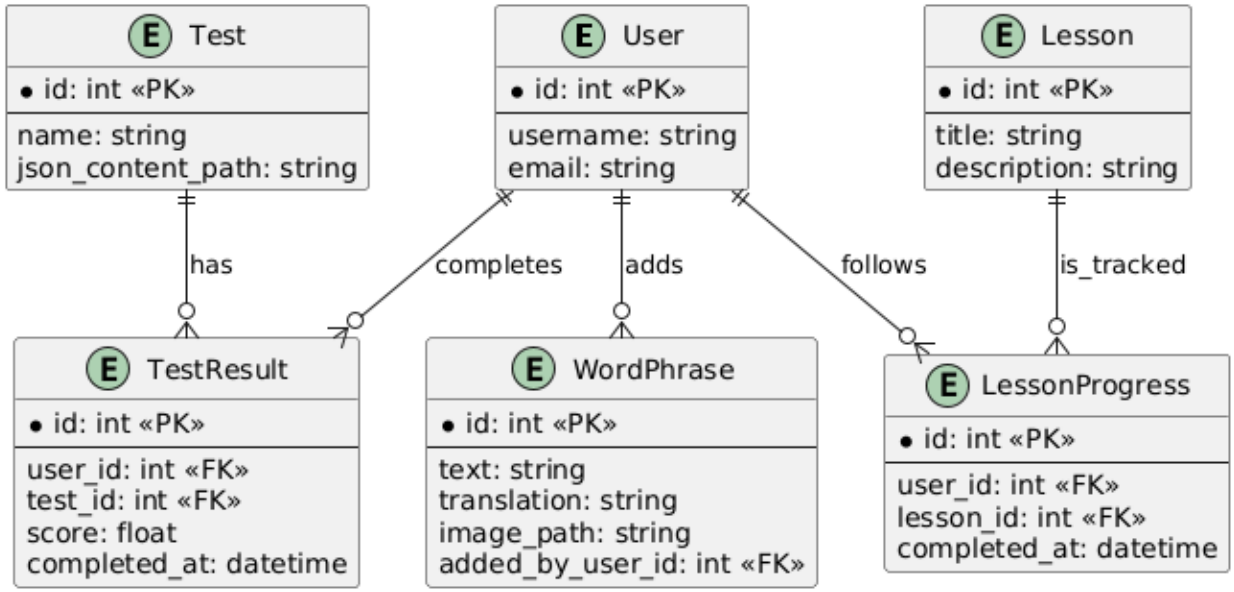

Рисунок 2.8. ER-модель БД

Таким чином, запропонована модель бази даних є структурно гнучкою, масштабованою та оптимізованою для мобільного використання. Вона

забезпечує збереження та доступ до ключової інформації про користувача, навчальний матеріал і результати, а також дозволяє легко розширювати функціональність застосунку в майбутньому.

2.3. Вибір технологій для реалізації

Оскільки передбачалось створення кросплатформеного мобільного застосунку, то було прийнято рішення використовувати .NET MAUI (Multi-platform App UI) як основну платформу для розробки інтерфейсу та бізнес-логіки застосунку, а також SQLite як вбудовану систему керування базами даних (СКБД) для локального збереження даних.

.NET MAUI – це кросплатформена платформа для створення мобільних і класичних додатків за допомогою C# і XAML. За допомогою .NET MAUI можна розробляти програми, які можуть працювати в Android, iOS, macOS та Windows з однієї загальної бази коду [24].

.NET MAUI має відкритий вихідний код і є еволюцією Xamarin.Forms, розроблена для підвищення продуктивності і розширюваності. За допомогою .NET MAUI можна створювати багатоплатформні програми за допомогою одного проекту, але при необхідності можна додати вихідний код та ресурси для конкретної платформи. Однією з ключових цілей .NET MAUI є реалізація максимальної частини логіки програми та макету інтерфейсу користувача в одному коді.

За допомогою .NET MAUI можна створювати кросплатформені додатки у XAML і C# з однієї загальної бази коду Visual Studio. Розробники можуть спільну використовувати макет інтерфейсу користувача на різних платформах. Таким чином забезпечується спільне використання коду, тестів та бізнес-логіки на різних платформах.

.NET MAUI об’єднує API Android, iOS, macOS та Windows в єдиний API, який працює за принципом "написав один раз, працює скрізь", а також забезпечує глибокий доступ до всіх аспектів кожної нативної платформи [25].

Платформи для створення програм .NET для Android, .NET для iOS, .NET для Mac Catalyst і бібліотеки інтерфейсу користувача Windows 3 (WinUI 3) мають доступ до однієї бібліотеки базових класів .NET (BCL). Ця бібліотека абстрагує інформацію про базову платформу від коду. BCL залежить від середовища виконання .NET, щоб надати середовище виконання коду. Для Android, iOS та macOS середовище реалізується за допомогою Mono, реалізація .NET-рантайму. На платформі Windows виконання забезпечується загальномовним середовищем виконання .NET Core (CLR) [24].

Хоча BCL дозволяє додаткам, що працюють на різних платформах, спільно використовувати одну бізнес-логіку, різні платформи мають різні способи визначення інтерфейсу користувача для програми, а також надають різні моделі для визначення того, як елементи інтерфейсу користувача взаємодіють між собою. Звісно, можна створити інтерфейс для кожної платформи окремо за допомогою відповідної платформи (.NET для Android, .NET для iOS, .NET для Mac Catalyst або WinUI 3), але цей підхід вимагає підтримки бази коду для кожного окремого сімейства пристроїв [24,25].

.NET MAUI надає єдину платформу для створення інтерфейсів для мобільних і класичних додатків. Високорівневе представлення архітектури програми .NET MAUI представлено на рис.2.9. У програмі .NET MAUI розробник створює код, який в основному взаємодіє з елементами керування .NET MAUI та рівнем API (1). Потім цей шар безпосередньо використовує власні платформи API (3). Крім того, код програми може безпосередньо виконувати API платформи (2), якщо це необхідно.

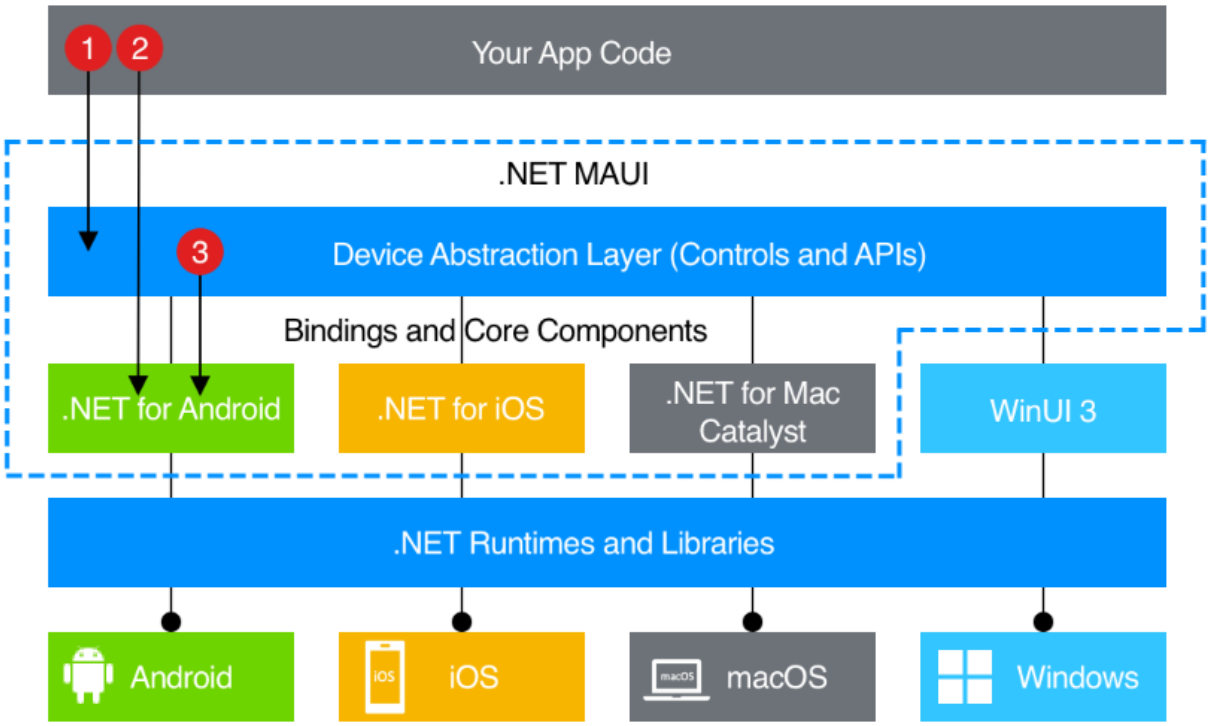

Рисунок 2.9. Схема архітектури .NET MAUI [24]

Програми .NET MAUI можна записувати на ПК або Mac і компілювати у власні пакети програм [24].

Програми Android, створені за допомогою .NET MAUI, компілюються з C# в проміжну мову (IL), яка потім компілюється методом JIT в нативне зборку при запуску програми.

Програми iOS, створені за допомогою .NET MAUI, повністю заздалегідь компілюються із C# у власний код збірки ARM.

Програми macOS, створені за допомогою .NET MAUI, використовують Mac Catalyst, рішення від Apple, яке переводить програму iOS, створену за допомогою UIKit на робочий стіл, і розширює її за допомогою додаткових API AppKit і платформи в міру необхідності.

Програми Windows, створені за допомогою .NET MAUI, використовують бібліотеку інтерфейсу користувача Windows 3 (WinUI 3) для створення власних програм, призначених для робочого столу Windows.

Для реалізації механізмів збереження локальних даних у мобільному застосунку було обрано SQLite — легку реляційну СКБД, яка ідеально

підходить для мобільних платформ завдяки своїй компактності, високій продуктивності та відсутності потреби у серверній інфраструктурі. SQLite дозволяє організувати структуру даних у відповідності до нормалізованої схеми, підтримує транзакції, індекси, запити SQL та забезпечує надійне зберігання навіть у режимі офлайн. Власне це особливо важливо для мобільних застосунків освітнього спрямування, оскільки користувачі повинні мати змогу працювати з контентом без постійного доступу до мережі Інтернет.

Поєднання .NET MAUI та SQLite дозволяє створити продуктивний, гнучкий та масштабований мобільний застосунок з розширеними можливостями для зберігання, обробки та візуалізації навчального матеріалу. Такий технологічний стек повністю відповідає вимогам до сучасного кросплатформного мобільного програмного забезпечення в освітній сфері.

Висновки до розділу 2

На основі проведеного аналізу мобільних платформ, особливостей проєктування освітнього застосунку та вибору відповідних технологій можна сформулювати такі висновки.

По-перше, сучасні мобільні платформи Android та iOS займають провідні позиції на ринку, охоплюючи разом понад 99% мобільних пристроїв. У зв’язку з цим розробка мобільного застосунку має передбачати підтримку обох платформ, що обумовлює доцільність використання кросплатформених засобів розробки. У цьому контексті .NET MAUI виступає як ефективне рішення, що забезпечує уніфікований підхід до створення інтерфейсу, бізнес-логіки та доступу до локальних ресурсів на основі спільної кодової бази.

По-друге, особливості проєктування мобільного освітнього застосунку вимагають дотримання принципів адаптивності, інтуїтивної взаємодії, локального збереження прогресу навчання, підтримки мультимедійного контенту та персоналізації. Було запропоновано логічну модель застосунку, яка включає в себе основні варіанти використання: вивчення уроків, проходження тестів, додавання слів та зображень до словника. На основі цієї моделі побудовано діаграми варіантів використання та активності, що сприяють формалізації вимог та переходу до об’єктно-орієнтованого проєктування.

По-третє, вибір SQLite як системи керування базами даних дозволяє ефективно реалізувати локальне збереження навчального контенту, даних користувача та структури словника без потреби у зовнішніх підключеннях. Такий підхід є виправданим у контексті мобільного навчального застосунку, оскільки гарантує автономну роботу, швидкий доступ до ресурсів, низьке навантаження на пристрій і простоту розгортання.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ "АНГЛІЙСЬКА ДЛЯ ПРОГРАМІСТІВ"

3.1. UX/UI-дизайн застосунку "Англійська для програмістів"

UX/UI-дизайн мобільного застосунку є одним із ключових етапів розробки програмного забезпечення, що безпосередньо впливає на якість взаємодії користувача з продуктом. Поняття UX (User Experience) охоплює сукупність вражень, емоцій, зручностей та загального досвіду користувача при роботі із застосунком. У свою чергу, UI (User Interface) стосується безпосередньо візуального оформлення, структури інтерфейсу, елементів керування та логіки їхнього розміщення [26].

Основна мета UX/UI-дизайну полягає у забезпеченні інтуїтивної, послідовної та ефективної взаємодії користувача із системою, що знижує когнітивне навантаження, мінімізує кількість дій для досягнення цілі та сприяє підвищенню рівня задоволеності. У випадку мобільних освітніх застосунків UX/UI-дизайн має особливу вагу, оскільки напряму впливає на мотивацію до навчання, збереження уваги користувача та ефективність засвоєння матеріалу. Зокрема, дизайн повинен враховувати обмеження розміру екрана, можливість використання застосунку в русі, забезпечення доступності для різних вікових і мовних груп.

Досить суттэвими принципами сучасного UX/UI-дизайну є послідовність, адаптивність, мінімалізм, візуальна ієрархія, швидкий зворотний зв’язок, а також відповідність очікуванням цільової аудиторії.

І Play Store, і Apple App Store вимагають від розробників дотримання певних вимог щодо дизайну освітніх програмних застосунків. Серед них можна виділити [26,27]:

1. Інтерфейс має бути побудований таким чином, щоб користувач без додаткового навчання міг зрозуміти його структуру та логіку. Це передбачає

логічне розміщення елементів керування, використання загальноприйнятих символів і термінів, а також відповідність очікуванням користувачів. Інтуїтивність значно знижує когнітивне навантаження та підвищує швидкість адаптації до нових функцій, що особливо актуально для освітніх продуктів із широкою цільовою аудиторією.

2. Застосунок повинен забезпечувати швидке та зручне отримання користувачем необхідного контенту – уроків, тестів, словникових записів тощо. Для цього слід реалізувати зрозумілу навігацію, а також структурувати матеріал за категоріями та рівнями складності. Забезпечення інформаційної доступності є критичним чинником у підтриманні мотивації користувача та ефективності навчального процесу.

3. Мінімалістичний підхід передбачає усунення зайвих елементів інтерфейсу, зосередженість лише на функціонально необхідних об’єктах та гармонійне поєднання кольорової палітри, типографіки та графічних елементів. Такий підхід сприяє зменшенню візуального шуму, покращує концентрацію уваги користувача та полегшує навігацію, що особливо важливо при використанні застосунку в динамічних умовах (наприклад, під час пересування).

4. Дотримання "правила великого пальця" – це правило, яке стосується ергономіки взаємодії з мобільним інтерфейсом: усі основні елементи управління повинні бути розташовані в зоні досяжності великого пальця користувача, що тримає пристрій однією рукою. Дотримання цієї вимоги сприяє комфортному використанню застосунку, зменшує ймовірність помилкових натискань і робить взаємодію з інтерфейсом безпечною та зручною навіть під час руху.

5. Візуальна ієрархія. Правильне розташування елементів за принципом візуальної ієрархії дозволяє користувачам швидко орієнтуватися в інтерфейсі, виділяючи основні дії та блоки інформації. Це досягається за допомогою розмірів шрифтів, кольорів, контрасту, відступів та розміщення на сторінці.

Ефективна візуальна ієрархія сприяє підвищенню юзабіліті застосунку та формує позитивний досвід користування.

Ґрунтуючись на цих принципах, проведемо проєктування інтерфейсу застосунку.

Ієрархія системи екранів мобільного застосунку наведена на рис.3.1.

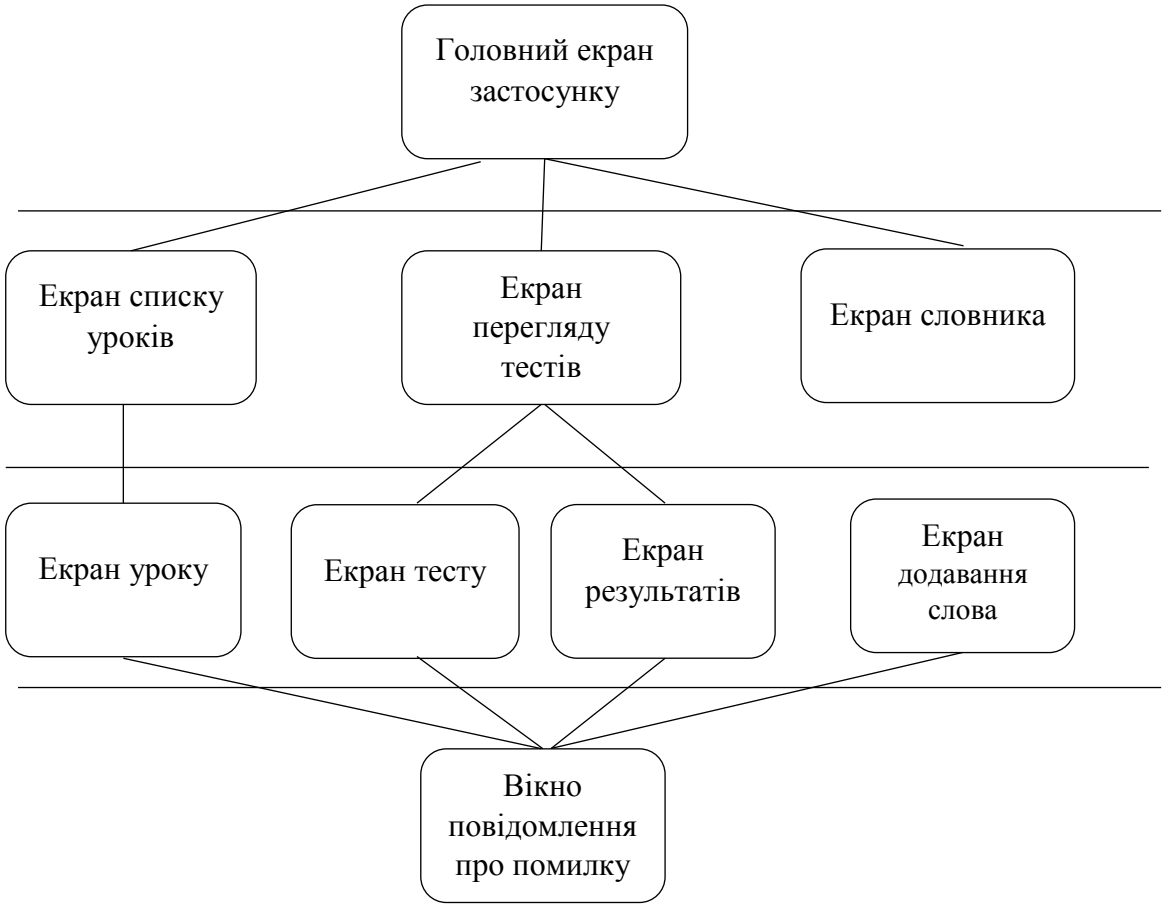

Рисунок 3.1. Ієрархія екранних форм мобільного застосунку з вивчення англійської мови

У процесі проєктування мобільного застосунку важливим аспектом є вибір кольорової гами інтерфейсу користувача, оскільки саме колірне оформлення істотно впливає на візуальне сприйняття, емоційний комфорт та загальну зручність взаємодії з програмним продуктом.

У проєкті було реалізовано кольорову гаму шляхом створення ресурсного словника у форматі XAML (ResourceDictionary), що забезпечує

централізоване та багаторазове використання кольорів у всіх компонентах інтерфейсу.

Основними кольорами інтерфейсу є фіолетово-синя гама:

– Primary (#512BD4) – базовий фірмовий колір, що використовується для основних елементів навігації, заголовків та кнопок дій.
– PrimaryDark (#ac99ea) – полегшений варіант основного кольору, який застосовується як фоновий або допоміжний для створення ієрархії елементів.
– Tertiary (#2B0B98) – насичений допоміжний колір для акцентування ключових дій.

Допоміжні кольори використовуються для розділення змістових блоків, фону, а також елементів з нижчим рівнем важливості. Secondary (#DFD8F7) використовується для підрозділів і другорядних елементів, а PrimaryDarkText (#242424) і SecondaryDarkText (#9880e5) для забезпечення читабельності.

Додаткові відтінки, зокрема Magenta, MidnightBlue, OffBlack, а також відтінки сірого від Gray100 до Gray950, використовуються для побудови візуальної структури, контурів, меж та неактивних елементів.

Кольори були реалізовані як ресурси типу SolidColorBrush, що забезпечує гнучкість у компонуванні стилів і дозволяє оперативно змінювати палітру без потреби втручання у структуру XAML-компонентів.

Обрана палітра кольорів відповідає сучасним принципам UX/UI-дизайну, оскільки забезпечує високий контраст і читабельність, підтримує візуальну ієрархію елементів. Така схема є емоційно привабливою та забезпечує зручність забезпечує зручність використання як у світлому, так і в темному режимі.

Загальний принцип побудови екранів полягає в наступному: основну частину екрану займають елементи уроку або питання тесту, а елементи управління знаходяться внизу. В горі останніх екранів знаходиться назва поточної екранної форми та стрілка для повернення на попередню екранну форму. Якщо на екрані передбачається вибір якихось дій, то елементи

управління розташовані зверху вниз. Такий підхід відображає звичне для людини сприйняття інформації.

Ескізи екранних форм мобільного додатку наведені на рис.3.2-3.7.

На головній екранній формі передбачена наявність кнопок для вибору режиму роботи застосунку (рис.3.2).

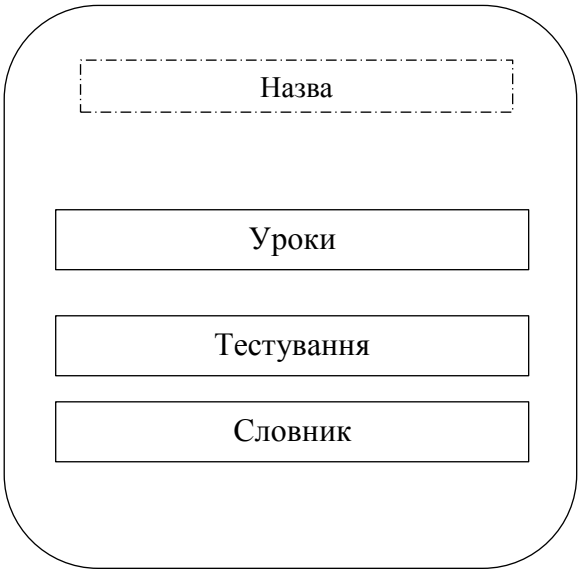

Рисунок 3.2. Ескіз головного вікна застосунку

Вибір уроків вирішено реалізувати у вигляді колекції кнопок, кожна з яких надає доступ до змісту конкретного уроку. Таким чином користувач розуміє на інтуїтивному рівні, як обрати потрібний урок. Ескіз вікна наведений на рис.3.3.

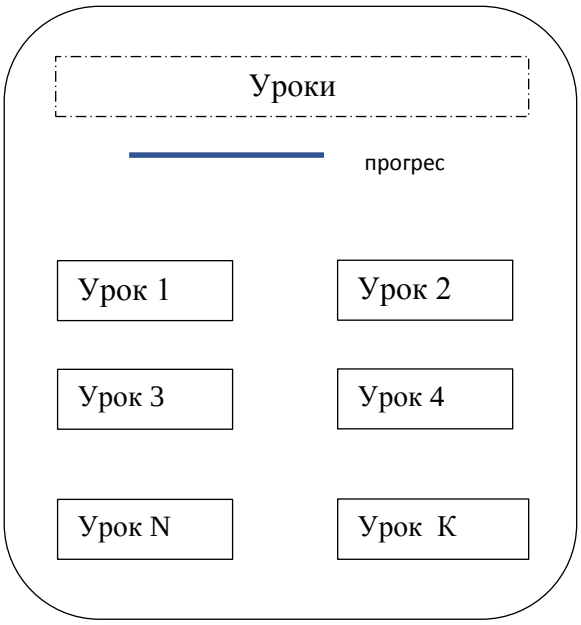

Рисунок 3.3. Ескіз вікна зі списком уроків

Ескіз екрану з тестовим питанням наведений на рис.3.4.

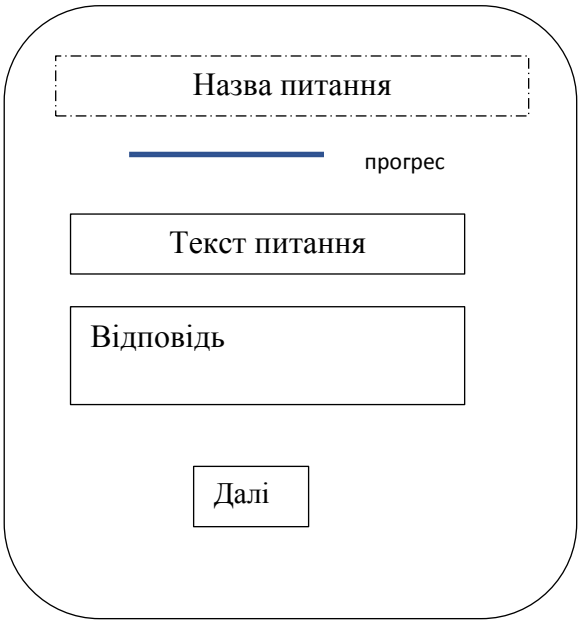

Рисунок 3.4. Ескіз екрану з тестовим питанням

Оскільки користувач може проходити один і той же тест декілька разів, то розроблений екран перегляду тестів повинен містити всі пройдені користувачем тести. Ескіз екрану наведений на рис.3.5.

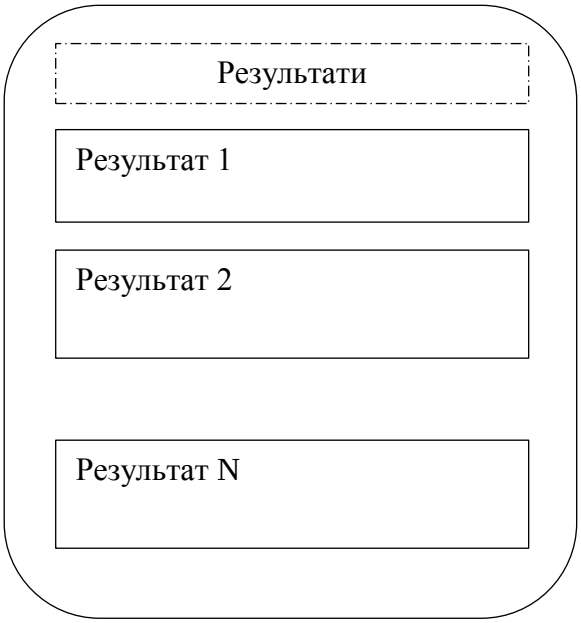

Рисунок 3.5. Ескіз вікна застосунку при перегляді результатів тестів

Для зручності користувача при вході у режим роботи зі словником користувачу пропонується можливість переглянути список вже доданих слів. Таким чином користувач може легко контролювати, чи потрібно йому додати нове слово. Ескіз екрану наведений на рис.3.6.

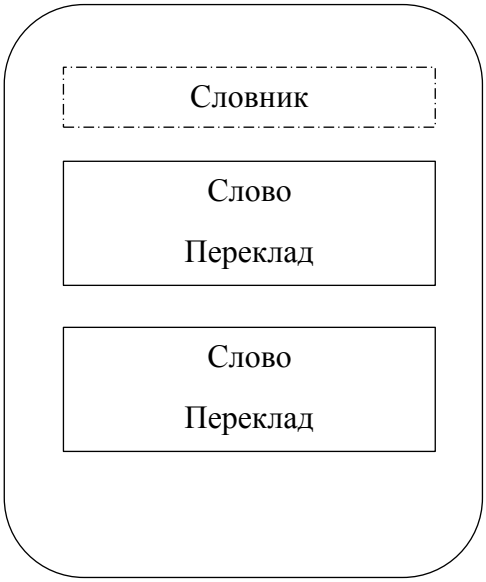

Рисунок 3.6. Ескіз вікна з переглядом словника

Для додавання слова був розроблений окремий екран, його ескіз наведений на рис. 3.7.

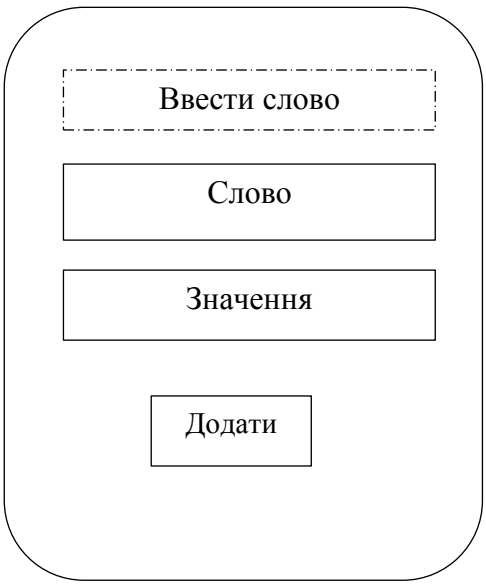

Рисунок 3.7. Ескіз вікна з переглядом словника

Проведене UI/UX проєктування дозволило створити застосунок зі зручним мінімалістичним інтерфейсом, який відповідає сучасним дизайнерським принципам. Його екрани необтяжені зайвою інформацією, а користувач при використанні може спиратися на свій користувацький досвід.

3.2. Реалізація основних функцій застосунку "Англійська для програмістів"

Для реалізації застосунку використовувалось середовище розробки Visual Studio Code.

Опишемо загальний алгоритм роботи мобільного застосунку.

Користувач заходить у головне вікно програми. Він має змогу обрати один з режимів роботи – переглянути урок, пройти тестування, працювати зі словником.

При виборі перегляду уроків він потрапляє до списку уроків, вибирає один з них, переглядає, і може потім повернутися назад до списку уроків.

При виборі тестування користувач може вибрати тип тесту або переглянути результати тестування. При виборі тесту користувач проходить

тест, відповідаючи на питання. Також він може повернутися до списку тестів. При виборі перегляду результатів тестування користувач може побачити результати своїх тестувань.

При виборі роботи з словником користувач може переглянути список доданих слів та додати нове слово до словника.

Схема алгоритму роботи користувач з застосунком наведена на рис.3.8.

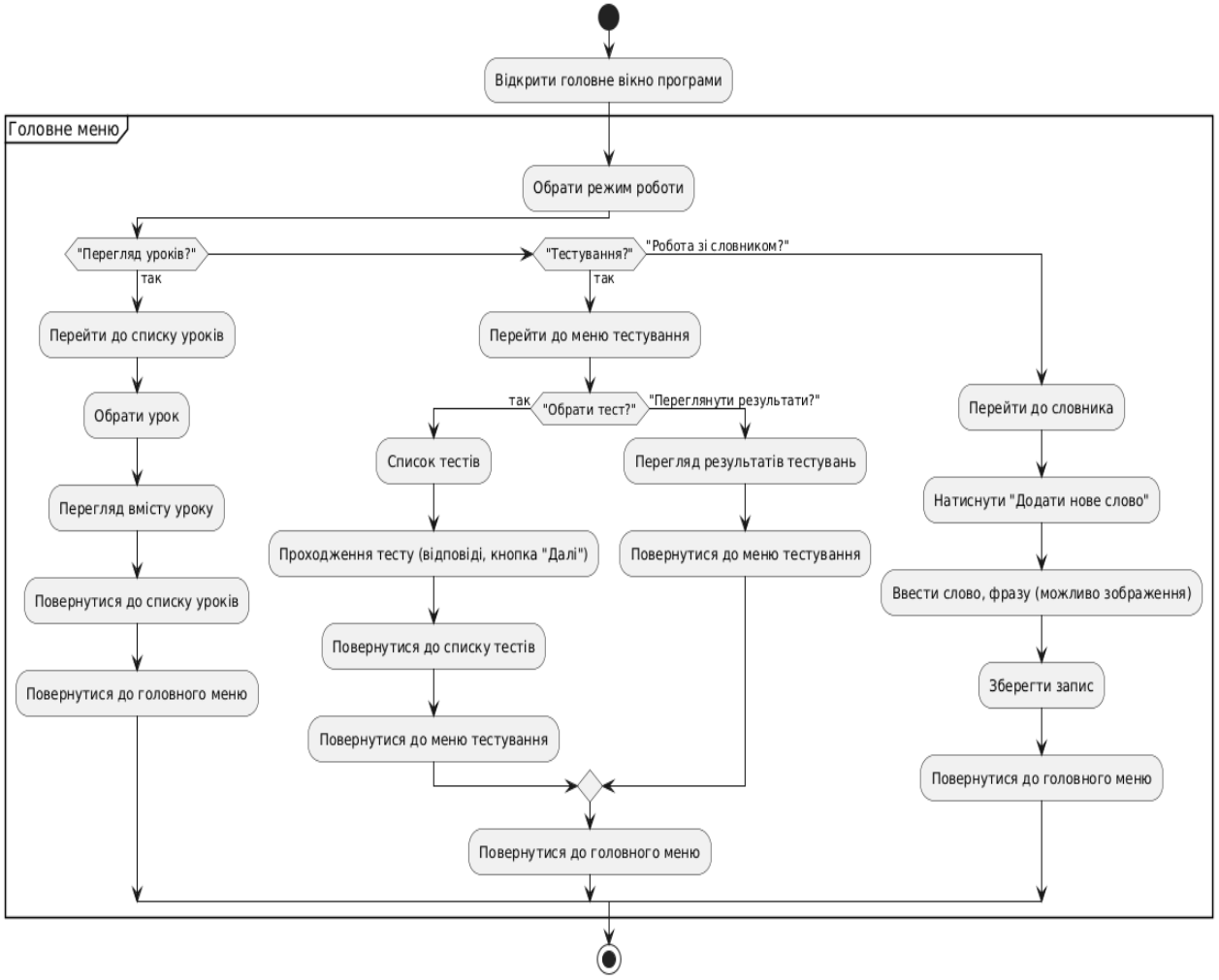

Рисунок 3.8. Схема алгоритму роботи користувача з мобільним застосунком для вивчення англійської мови

Головне вікно програми виконує роль центрального навігаційного вузла для користувача. Основна функціональність MainPage полягає у забезпеченні доступу до ключових розділів системи: перегляду навчальних матеріалів,

проходження тестування, роботи з електронним словником та налаштувань застосунку.

Клас MainPage, який успадковується від базового класу ContentPage, ініціалізується методом InitializeComponent(), що забезпечує завантаження графічного інтерфейсу, визначеного у відповідному XAML-файлі. Усі логічні обробники взаємодії з елементами інтерфейсу реалізовано як асинхронні методи, що забезпечує неконфліктну роботу з користувацьким інтерфейсом у режимі реального часу (рис.3.9).

Зокрема, при натисканні на відповідні кнопки головного меню реалізовано перехід до визначених сторінок за допомогою механізму Shell.Current.GoToAsync(), що є складовою частиною навігаційної моделі Shell, рекомендованої у сучасних MAUI-застосунках.

Винятком є перехід до сторінки налаштувань (SettingsPage), який реалізовано з використанням стекової навігації (Navigation.PushAsync(...)). Такий підхід дає змогу користувачеві повертатися назад, зберігаючи контекст попередньої взаємодії.

```
private async void OnStartLessonClicked(object sender, EventArgs e)
{
    // Перехід на сторінку уроків за абсолютним маршрутом
    await Shell.Current.GoToAsync("//LessonsPage");
}

private async void OnOpenDictionaryClicked(object sender, EventArgs e)
{
    // Передаємо базу даних при переході на DictionaryPage

    await Shell.Current.GoToAsync("//DictionaryPage");
}

private async void OnStartTestClicked(object sender, EventArgs e)
{
    // Перехід на сторінку тестування за абсолютним маршрутом
    await Shell.Current.GoToAsync("//QuizPage");
}
private async void OnSettingsClicked(object sender, EventArgs e)
{
    await Navigation.PushAsync(new SettingsPage()); // якщо вже є SettingsPage
}
```

Рисунок 3.9. Фрагмент коду реалізації головної сторінки застосунку

Таким чином, головна сторінка застосунку виконує роль інтерфейсного посередника між користувачем і функціональними модулями системи, забезпечуючи логічно структуровану навігацію та інтуїтивну взаємодію.

Для реалізації структури даних для застосунку, було розроблено модель, яка описує логіку подання навчального матеріалу. Модель представлена у вигляді трьох взаємопов’язаних класів: Lesson, Content та Section, що відображають ієрархічну організацію навчального контенту.

Клас Lesson описує загальну структуру навчального уроку. Він містить такі основні властивості: Number – порядковий номер уроку, що забезпечує впорядкування у послідовності навчального процесу, Title – заголовок уроку, який використовується в інтерфейсі користувача для навігації, Description – короткий опис змісту уроку, Content – об’єкт типу Content, який містить деталізовані розділи (секції) уроку, NextLesson – номер наступного уроку, що дозволяє реалізувати послідовну логіку переходу між етапами навчання. Завдяки цій структурі, клас Lesson виконує роль контейнера для всіх навчальних компонентів, що входять до певного уроку.

Клас Content є допоміжною структурою, що містить список об’єктів Section. Такий підхід дозволяє створювати гнучку і розширювану структуру навчального матеріалу, де кожна секція представляє окрему одиницю інформації (наприклад, текстовий блок, зображення або відео). Цей рівень абстракції є зручним для реалізації механізмів відображення в інтерфейсі, адаптивної візуалізації та подальшої підтримки мультимедійного контенту.

Клас Section описує окремий елемент контенту в уроці. Його властивості дають змогу відобразити різні типи матеріалів з відповідним форматуванням: Type визначає тип секції (наприклад, "text", "image", "video"), що забезпечує адаптивну обробку під час рендерингу інтерфейсом, Value – текстове значення або шлях до медіа-файлу, Style – текстовий опис стилю (наприклад, "Bold", "Italic", "Center"), що впливає на візуальне оформлення.

Клас також містить дві похідні властивості. FontStyle реалізує логіку конвертації стилю в значення типу FontAttributes, яке використовується для форматування тексту в UI MAUI. HorizontalAlignment реалізує інтерпретацію стилю для горизонтального вирівнювання тексту, повертаючи значення типу TextAlignment.

Схема класів представлена на рис.3.10.

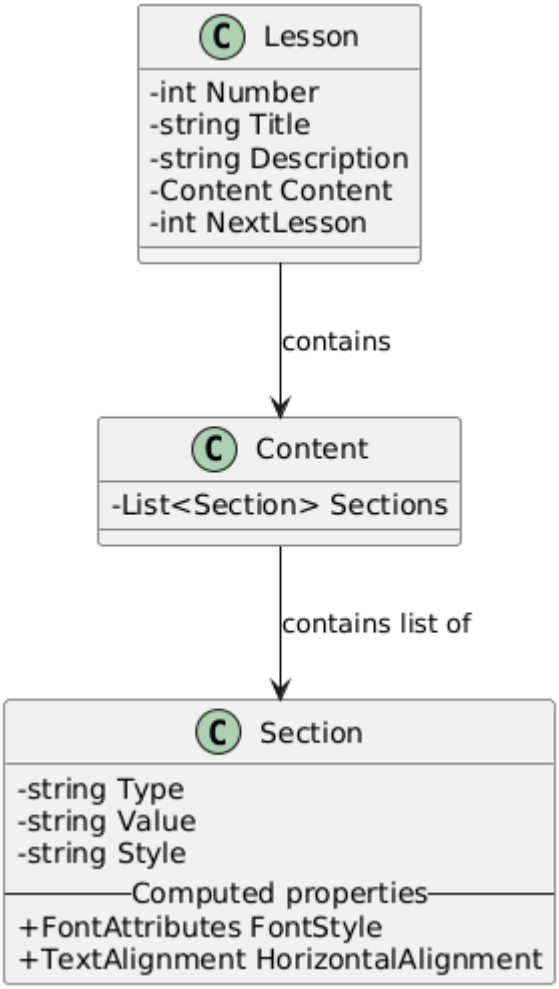

Рисунок 3.10. Схема класів моделі Lessons

В запропонованій моделі реалізований принцип інкапсуляції, що полегшує масштабування функціональності, повторне використання компонентів та інтеграцію з візуальними елементами .NET MAUI. Гнучкість реалізації секцій дозволяє динамічно адаптувати уроки до різних форматів (наприклад, лише текстові, комбіновані з медіа, тощо), що відповідає сучасним вимогам до мобільного освітнього програмного забезпечення.

В програмній системі реалізовано декілька варіантів тестів, що дає змогу перевірити як знання перекладу слів, так і вміння грамотно їх писати та використовувати.

На сторінці тестів реалізована обробка різних кнопок за допомогою асинхронної стекової навігації, що забезпечує плавність переходу користувача між екранами різних тестів та дозволяє гнучко управляти історією переходів.

Одним з основних тестів є тест на перевірку лексики. Він реалізований за допомогою компоненти ITWordTest.xaml.cs. Користувачу послідовно представляються значення слів, на які потрібно надати відповідник мовою вивчення. Слова завантажуються зі словника і випадковим чином відбирається 24 слова з дотриманням унікальності відбору. Для перевірки відповіді можуть використовуватися декілька стратегій, що і реалізовано за допомогою патерна "Стратегія". Такий підхід сприяє гнучкості у визначенні правил перевірки правильності.

Якщо відповідь правильна, збільшується лічильник балів та виводиться позитивне повідомлення. У випадку помилки виводиться повідомлення про неправильну відповідь разом із правильною відповіддю, стилізоване за допомогою об’єкта FormattedString.

Фрагмент коду перевірки має вигляд:

```
if (_answerCheckStrategy.CheckAnswer(userAnswer, correctAnswer))
   {
     ResultLabel.Text = Strings.Right;
     ResultLabel.TextColor = Colors.Green;
     score++;
    }
  else
    {
      // Форматуємо текст для лейбла
     ResultLabel.FormattedText = new FormattedString
       {
         Spans =
          {
     // Текст "Wrong! Correct answer:" буде червоним і по центру
           new Span { Text = Strings.WrongAnswerMessage, TextColor = Colors.Red, FontSize
= 18, TextDecorations = TextDecorations.None, },

    // Текст правильної відповіді буде жирним, червоним і по центру, з нижчим відступом
           new Span { Text = correctAnswer, TextColor = Colors.Red, FontSize = 20,
FontAttributes = FontAttributes.Bold, }
            }
```

Логіка компоненти забезпечує блокування кнопки перевірки на час очікування між питаннями, що покращує UX та запобігає повторному натисканню.

Після проходження всіх питань відображається підсумковий результат, який зберігається у сховищі за допомогою сервісу TestResultService. Збереження результату включає кількість правильних відповідей, загальну кількість питань, дату проходження та тип тесту.

Другий варіант тестів, реалізований в системі, – це тести з множинним вибором. Для цього був розроблений компонент MASTest.xaml.cs. Даний компонент реалізує логіку проведення тестування за допомогою вибірки випадкових запитань. У процесі ініціалізації компонента відбувається завантаження питань із JSON-файлу з використанням сервісу FileService.

Фрагмент коду завантаження питань

```
public NewPage1()
   {
      InitializeComponent();
      _fileService = new FileService();
      LoadQuestions();
   }

private async void LoadQuestions()
   {
     // Завантажуємо питання з JSON файлу
     allQuestions = await _fileService.LoadDataAsync<List<QuestionModel>>
                              ("questions.json");
     // Вибір 4 випадкових питань без повторень
    selectedQuestions = allQuestions.OrderBy(x => Guid.NewGuid()).Take(10).ToList();
    }
```

З повного набору питань випадковим чином обираються 10 унікальних екземплярів, які формують поточний сеанс тестування.

Кожне питання відображається з випадково перемішаними варіантами відповідей. При взаємодії користувача з відповідним елементом інтерфейсу (кнопкою) відбувається перевірка вибраного варіанту на відповідність правильному. У разі правильної відповіді лічильник correctAnswers збільшується. Користувацький інтерфейс надає візуальний зворотний зв’язок: правильна відповідь підсвічується зеленим кольором, неправильна – червоним. Додатково відзначається правильна відповідь навіть при помилковому виборі.

Також відбувається збереження результатів тесту. Фрагмент коду, що відповідає за збереження результатів.

```
private async void ShowResult()
    {
     // Сховуємо питання та кнопки
       questionLabel.IsVisible = false;
       optionsLayout.IsVisible = false;

     // Показуємо результат
       resultLabel.IsVisible = true;
       resultLabel.Text = $"{Strings.Res} {correctAnswers}/{selectedQuestions.Count}";
       resultLabel.TextColor = Color.FromHex("#A9B8D3");

       // Зберігаємо результат тесту
       var testResult = new TestResult
         {
            Score = correctAnswers,
            TotalQuestions = selectedQuestions.Count,
            TestDate = DateTime.Now,
            TestType = "Choose 1 correct answer"
          };

        await TestResultService.Instance.SaveResultAsync(testResult);

        // Кнопка повернення на головну сторінку
        backButton.IsVisible = true;
}
```

Компонет Test1Page реалізує функціональність проходження тесту, заснованого на візуальних зображеннях. Користувачеві по черзі демонструються зображення, до яких необхідно ввести відповідь. Застосунок порівнює введену відповідь із прийнятними варіантами та формує зворотний зв’язок. Тест завершується після досягнення ліміту у 10 питань або вичерпання доступних зображень.

При реалізації усі дії, пов’язані з навігацією або затримкою, реалізовано за допомогою асинхронних методів (async/await) для уникнення блокування UI-потоку. Було введено механізм для приховання клавіатури, очищення попередніх результатів, динамічну зміну інтерфейсу після завершення тесту. Використовується внутрішній облік вже використаних питань для уникнення дублювань.

Фрагмент коду, для оптимізації UI/UX-дизайну наведений нижче.

```
// Додаємо жест для приховання клавіатури на весь контейнер сторінки
var tapGestureRecognizer = new TapGestureRecognizer();
tapGestureRecognizer.Tapped += OnScreenTapped; // Використовуємо один метод

// Додаємо GestureRecognizer до кореневого елемента (VerticalStackLayout)
MainLayout.GestureRecognizers.Add(tapGestureRecognizer);

private void OnScreenTapped(object sender, EventArgs e)
 {
```

```
    // При натисканні на екран знімаємо фокус з поля вводу, що призведе до приховання
клавіатури
     AnswerEntry?.Unfocus();
    }
```

Компонент AddWordPage призначений для додавання нових слів до лексичної бази даних. Для цього в класі AddWordPage передбачена ініціалізацію об’єкта типу SQLiteDatabase, який відповідає за взаємодію з локальною реляційною базою даних.

```
    private SQLiteDatabase _database;
    public AddWordPage(SQLiteDatabase database)
       {
         InitializeComponent();
         _database = database;
       }
```

Основна функціональність реалізована у методі-обробнику OnAddWordClicked, який активується у відповідь на натискання кнопки додавання слова. У межах цього методу здійснюється створення об’єкта Word, властивості якого WordText та Meaning заповнюються на основі введених користувачем значень.

Перед збереженням даних до бази здійснюється валідація введених даних: перевіряється, чи не є поля порожніми або такими, що містять лише пробіли. У разі успішного проходження перевірки новий об’єкт додається до бази даних за допомогою асинхронного методу AddWordAsync, після чого користувачу виводиться повідомлення про успішне завершення операції. У протилежному випадку відображається повідомлення про помилку введення.

Фрагмент коду реалізації

```
private async void OnAddWordClicked(object sender, EventArgs e)
  {
    var newWord = new Word
     {
      WordText = WordEntry.Text,
      Meaning = MeaningEntry.Text
     };

    if (!string.IsNullOrWhiteSpace(newWord.WordText) &&
!string.IsNullOrWhiteSpace(newWord.Meaning))
     {
       await _database.AddWordAsync(newWord);  // Додаємо слово до бази даних
       await DisplayAlert(Strings.AddWordSuccessTitle, Strings.AddWordSuccessMessage,
"OK");

 // Повертаємось на попередню сторінку
       await Navigation.PopAsync();
```

```
    }
    else
     {
       await DisplayAlert(Strings.AddWordErrorTitle, Strings.AddWordErrorMessage, "OK");
     }
}
```

Таким чином, даний програмний компонент забезпечує зручну та захищену форму введення лексичних одиниць, сприяючи поступовому збагаченню персоналізованої мовної бази користувача.

Використання асинхронного підходу до обробки даних дозволяє підтримувати належний рівень чутливості інтерфейсу, що відповідає сучасним вимогам до мобільних застосунків. Всі компоненти успішно реалізують мінімалістичний інтуїтивний інтерфейста реалізують зручний мінімалістичний інтерфейс.

3.3. Тестування застосунку "Англійська для програмістів"

Проведемо тестування застосунку, розглянувши його роботу.

При запуску застосунку на екрані з’являється навігаційне вікно програми (рис.3.11).

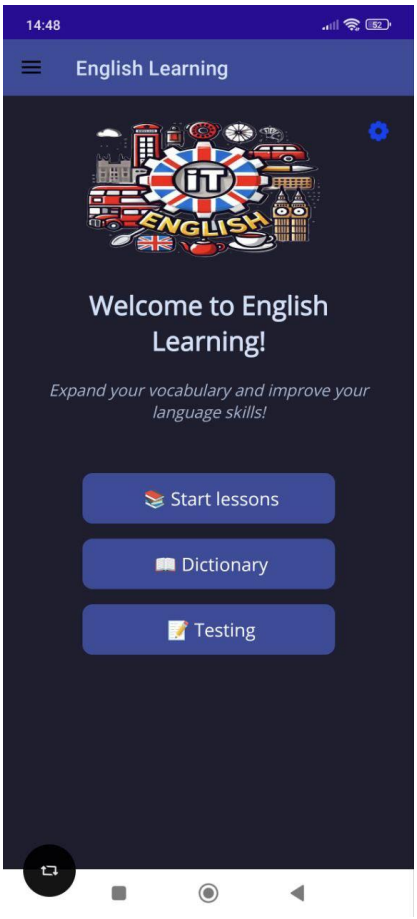

Рисунок 3.11. Головне (навігаційне) вікно мобільного застосунку

Вікно містить 3 кнопки, які дозволяють працювати з програмою в трьох різних режимах: проходження уроків, тестування та додавання нових слів.

Після натискання на кнопку "Lessons" користувач переходить до списку уроків (рис.3.12). На екрані зображені кнопки з назвами уроків. Також користувач бачить свій прогрес – кількість пройдених уроків.

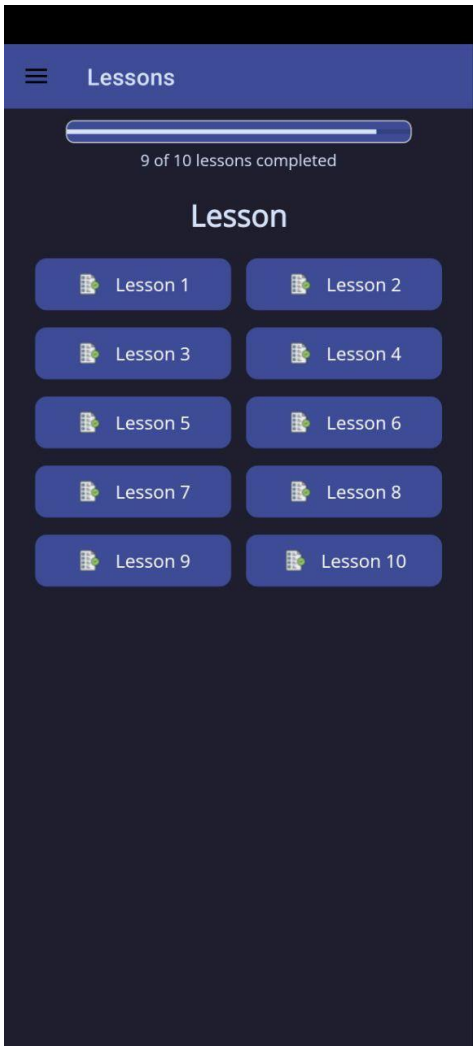

Рисунок 3.12. Екран зі списком уроків

Користувач може обрати потрібний урок та пройти його. При проходженні нового уроку кількість пройдених уроків в елементі Прогрес збільшиться на 1.

При виборі кнопки "Testing". Користувач переходить до екрану тестування (рис.3.13).

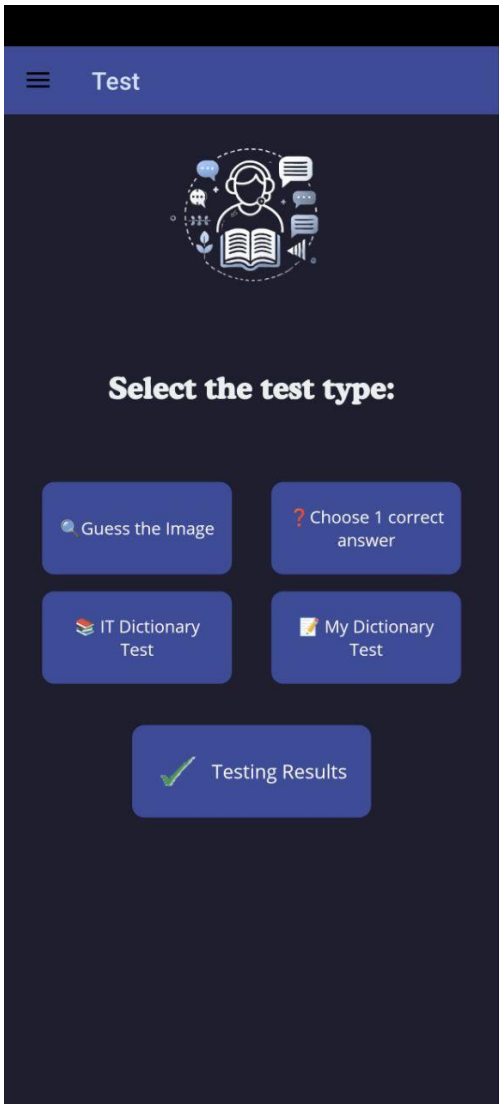

Рисунок 3.13. Екран з варіантами тестів

При виборі тесту "Guess the image" користувач переходить до тесту за картинками. Приклад питання даного тесту наведений на рис.3.14.

Користувачу показується зображення, яке пов’язане з IT термінологією. Користувач повинен ввести правильну відповідь англійською мовою. Також на екрані відображається прогрес користувача при проходженні даного тесту. Після відповіді користувачу потрібно натиснути кнопку "Next".

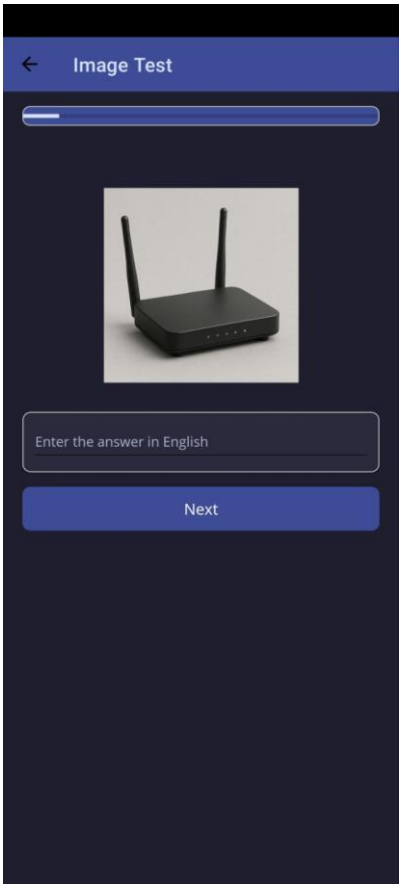

Рисунок 3.15. Приклад питання з зображенням

Користувач може повернутися на попередній екран, натиснувши кнопку стрілка.

При виборі кнопки "Choose 1 correct answer" користувачу потрібно вибрати одну правильну відповідь серед 4 варіантів. Приклад питання даного типу наведений на рис.3.16.

Після натискання кнопки з відповіддю користувач переходитьдо наступного питання.

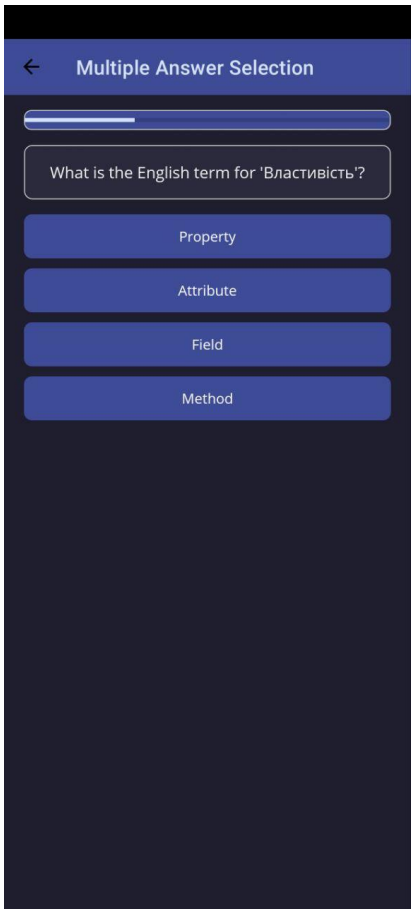

Рисунок 3.116. Приклад питання з вибором однієї правильної відповіді

При виборі кнопки "IT Dictionary test" користувачу пропонується слово українською мовою. Він повинен внести англійський еквівалент (рис.3.17).

Після того, як користувач натискає кнопку "Next", він переходить до наступного слова.

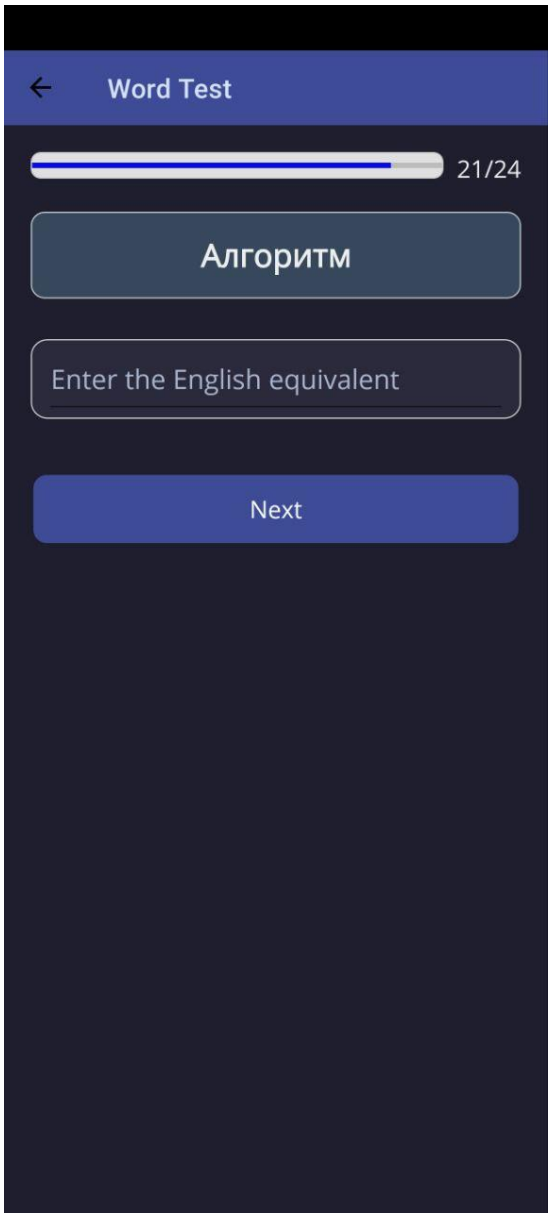

Рисунок 3.17. Приклад питання з необхідністю введення відповіді.

Всі тести мають можливість повернути ся на попередній екран. Також на екрані відображається лінія прогресу, яка відмічає кількість пройдених питання і скільки питань залишилось.

Також на екрані тестування користувач може переглянути результати своїх тестів (рис.3.18).

Для кожного пройденого тесту наводиться його назва, кількість правильних відповідей, загальна кількість питань та дата проходження. Це дає змогу відслідковувати свій прогрес.

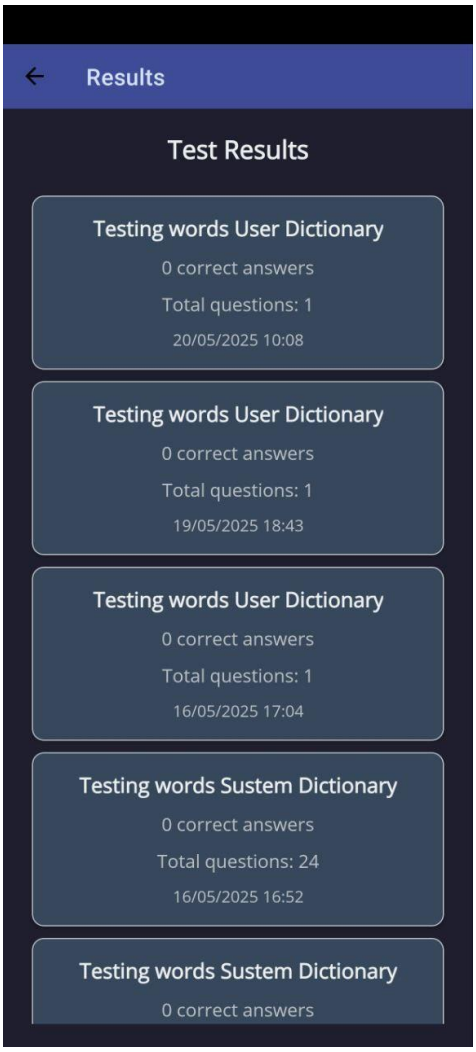

Рисунок 3.18. Екран результатів тестування

При виборі кнопки "Dictionary" користувач може переглянути список введених в систему слів та фраз (рис.3.19).

Користувач має змогу додати або видалити слово чи фразу з даного словника.

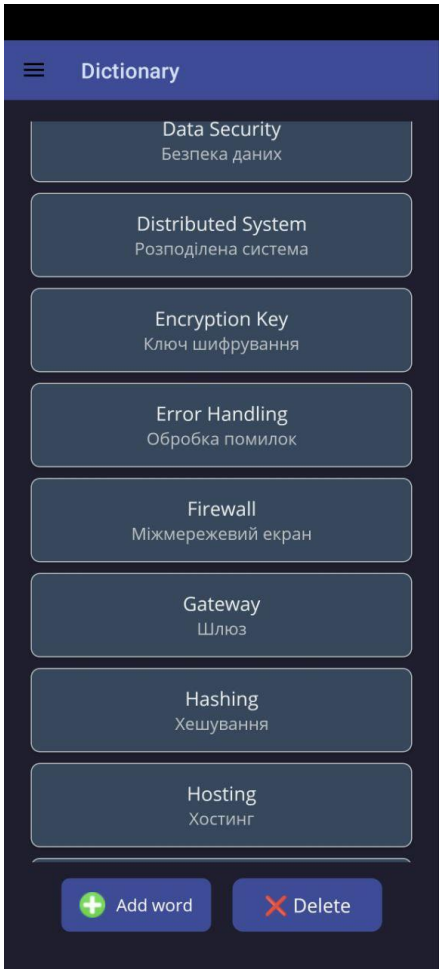

Рисунок 3.19. Екран зі списком слів

Після натискання кнопки "Add" користувач на наступному екрані вводить український та англійський варіант слова чи фрази (рис.3.20). Після введення слова користувач натискає кнопку "Add" і слово додається в словник.

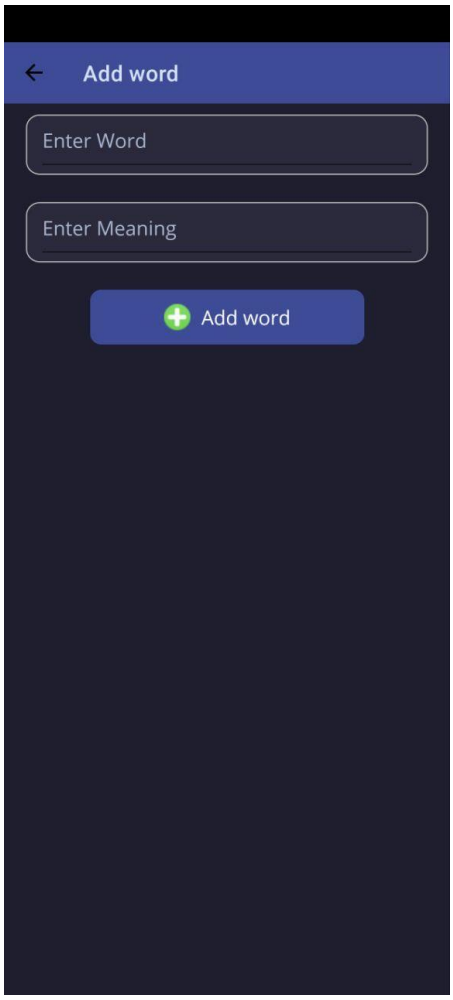

Рисунок 3.20. Екран додавання нового слова

Отже, результати тестування показали, що розроблений мобільний застосунок відповідає визначеним вимогам до застосунку з вивчення англійської мови ІТ-фахівцями.

Висновок до розділу 3

В третьому розділі був проведений UI/UX-дизайн, який забезпечив створення застосунку з інтуїтивно зрозумілим та мінімалістичним інтерфейсом. Підбір кольорової гами та визначення елементів екранних форм відбувався з врахуванням цільового призначення даного застосунку.

За результатами проєктування була здійснена реалізація мобільного застосунку. При реалізації програмної системи була використана асинхронна логіка керування інтерфейсом та патерни конструювання (зокрема патерн Стратегія), що забезпечило ефективну реалізацію адаптивного навчального інструмента для вивчення термінології в галузі інформаційних технологій.

Проведене тестування показало, що розроблений застосунок реалізує всі зазначені функції освітньої програми, а саме, вивчення підготовленого матеріалу, тестування з використанням тестів різного виду та можливість рефлексії за результатами тестування, додавання нового контенту.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено мобільний застосунок, призначений для підтримки та покращення процесу вивчення англійської мови, орієнтованого на фахівців галузі інформаційних технологій.

Були досягнуті наступні результати.

1. Обґрунтовано актуальність створення засобів, які поєднують вивчення англійської мови з галузевою специфікою IT, а також враховують обмежений вільний час користувачів і потребу в мобільності.

2. Розроблено архітектуру застосунку, яка включає функціональні модулі для перегляду запитань, тестування знань, перегляду результатів та додавання нової лексики до бази. Навігація між сторінками реалізована із застосуванням стекової асинхронної моделі, що відповідає принципам побудови UX у кросплатформенних рішеннях. Якісний UX/UI-дизайн не лише покращує зовнішнє враження від застосунку, але й сприяє досягненню основної функціональної мети – ефективному та комфортному засвоєнню знань користувачем. Його інтеграція в процес розробки є необхідною умовою створення конкурентоспроможного та затребуваного цифрового продукту

3. Імплементовано систему тестування, яка дозволяє користувачеві проходити інтерактивні опитування з варіантами відповідей, отримувати миттєвий зворотний зв’язок щодо правильності відповіді та бачити статистику результатів.

4. Здійснено інтеграцію з локальною базою даних SQLite, що дає змогу зберігати додані користувачем слова, що розширює персоналізований словник, та забезпечує автономну роботу застосунку без постійного доступу до Інтернету. Застосування .NET MAUI у поєднанні з SQLite дозволяє створити функціонально повноцінний, адаптивний і кросплатформний мобільний застосунок для вивчення англійської мови, що відповідає сучасним

вимогам до зручності, надійності та якості користувацького досвіду в освітній сфері.

5. Реалізоване рішення може бути використане як допоміжний інструмент у процесі самонавчання або професійної підготовки ІТ-спеціалістів.

6. Подальший розвиток застосунку передбачає розширення функціоналу за рахунок додавання аудіо- та відео-матеріалів, статистичного аналізу успішності навчання, а також синхронізації з хмарними сервісами для збереження прогресу та обміну даними між пристроями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. How to Create an Educational App: Process, Costs and Features URL: https://inoxoft.com/blog/how-to-create-an-educational-app-process-costs-and-features/ (дата звернення 25.07.2025)

2. The Complete Guide To Building Educational Apps In 2024: From Concept To Launch. URL:https://soloway.tech/blog/the-complete-guide-to-building-educational-apps/ (дата звернення 25.07.2025)

3. Çakmak F. Mobile Learning and Mobile Assisted Language Learning in Focus. 2009. C. 30-48.

4. Alahmad M. The Role of Form-Focused Instruction on Foreign Language Learners. *Budapest International Research and Critics in Linguistics and Education (BirLE) Journal*. 2019. T. 2, № 4. C. 44-53. URL: https://www.bircu-journal.com/index.php/birle (дата звернення 25.07.2025)

5. Lai C., Zheng D. Self-directed use of mobile devices for language learning beyond the classroom. *ReCALL*. 2018. Vol. 30(3). P. 299-318. DOI: 10.1017/S0958344017000258 (дата звернення 25.07.2025)

6. Godwin-Jones R. Smartphones and language learning. *Language Learning & Technology*. 2017. Vol. 21(2). P. 3-17. DOI: 10125/44607 (дата звернення 25.07.2025)

7. Castañeda D.A., Cho M.H. Use of a Game-like Application on a Mobile Device to Improve Accuracy in Conjugating Spanish Verbs. *Computer Assisted Language Learning*. 2016. Vol. 29(7). P. 1195-1204.

8. Aharony N., Bar-Ilan J. Students' Perceptions on MOOCs: An Exploratory Study. *Interdisciplinary Journal of e-Skills and Lifelong Learning*. 2016. Vol. 12. P. 145-162. DOI: 10.28945/3540

9. Duolingo. URL:https://uk.duolingo.com/ (дата звернення 25.07.2025)

10. Memrise. URL:https://www.memrise.com/en-us/ (дата звернення 25.07.2025)

11. Busuu. URL:https://www.busuu.com/en (дата звернення 25.07.2025)

12. Babbel. URL: https://ua.babbel.com/ (дата звернення 25.07.2025)

13. EWA: Learn English URL:https://appewa.com/ (дата звернення 25.07.2025)

14. Пономарьов І. В. Технології розробки кросплатформених веб-додатків: конспект лекцій. Дніпро : ДНУ, 2023. 113 с.

15. Грицюк Ю. Аналіз вимог до програмного забезпечення. Львів: Львівська Політехніка, 2018. 456 с.

16. Mobile & Tablet Operating System Market Share Worldwide URL:https://gs.statcounter.com/os-market-share/mobile-tablet/worldwide/ (дата звернення 25.07.2025)

17. Sundara Krishna Y.K., Mohan Deverkonda G.K. A Survey on Architectures of Mobile Operating Systems: Challenges and Issues. *International Journal of Research Studies in Computer Science and Engineering*. 2015. Vol. 2(3). P. 73-76.

18. Давидов М.В., Демчук А.Б., Лозинська О.В. Програмне забезпечення мобільних пристроїв: навчальний посібник. Львів: Новий Світ-2000, 2020. 218 с.

19. Дворецький М.Л., Нездолій Ю.О., Дворецька С.В., Кандиба І.О. Розробка мобільних застосунків для OS Android : навч. посіб. Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2021. 140 с.

20. Кузьма К.Т. Програмування мобільних пристроїв: навч. посіб. для дистанційного навчання. Миколаїв : СПД Румянцева Г.В., 2021. 128 с.

21. Проектування інформаційних систем: Загальні питання теорії проектування ІС (конспект лекцій) [Електронний ресурс]: навч. посіб. для студ. спеціальності 122 «Комп’ютерні науки» / КПІ ім. Ігоря Сікорського; уклад.: О. С. Коваленко, Л. М. Добровська. Електронні текстові дані (1 файл: 2,02 Мбайт). Київ : КПІ ім. Ігоря Сікорського, 2020. 192с

22. Проектування інформаційних систем: Посібник /За ред. В.С.Пономаренка. К.: «Академія», 2002. 486 с.

23. Nolan G., Cinar O., Truxall D. Android Best Practices. Springer, 2014. 222 p.

24. .NET MAUI URL: https://learn.microsoft.com/eu-eu/dotnet/maui/what-is-maui?view=net-maui-9.0 (дата звернення 25.07.2025)

25. .NET MAUI URL: https://abitap.com/category/net-maui/ (дата звернення 25.07.2025)

26. Mobile UI/UX Designs: Captivating & Innovative — vol. 212 URL: https://medium.com/@theymakedesign/mobile-ui-ux-designs-vol-212-23aa944b83d3 (дата звернення 25.07.2025)

27. Mobile App Design: UI/UX Principles, Best Practices & Examples [Електронний ресурс]. – Режим доступу: https://www.bairesdev.com/blog/mobile-app-design/ (дата звернення 25.07.2025)

28. Joshi D. The Role of UI/UX Design in Mobile App Success, 2023 URL: https://quokkalabs.com/blog/uiux-design-services-in-mobile-app-success/ (дата звернення 25.07.2025)

29. What is .NET MAUI? – .NET Multi-platform App UI documentation | Microsoft Docs URL: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-7.0 (дата звернення 25.07.2025)

30. Controls – .NET Multi-platform App UI documentation | Microsoft Docs URL: https://learn.microsoft.com/en-us/dotnet/maui/user-interface/controls/?view=net-maui-6.0 (дата звернення 25.07.2025)

31. Clean Architecture Tutorial for Android: Getting Started URL: https://www.raywenderlich.com/3595916-clean-architecture-tutorial-for-android-getting-started (дата звернення 25.07.2025)

32. Фрімен Е., Робсон Е. Head First. Патерни проєктування. Київ : Фабула, 2020. 672 с.

ДОДАТКИ

Додаток А

MainPage.xaml

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:resx="clr-namespace:MauiApp1.Resources.Lang"
             xmlns:local="clr-namespace:MauiApp1"
             x:Class="MauiApp1.MainPage"
             BackgroundColor="#1E1E2F">

    <Grid>

        <ScrollView>
            <VerticalStackLayout Padding="30,10,30,20" Spacing="20"
                                 HorizontalOptions="Center" VerticalOptions="Start">
                <Image Source="logotip.png" HeightRequest="160" Aspect="AspectFit"
                       SemanticProperties.Description="Logo for the English learning
app"
                       HorizontalOptions="Center" />

                <Label Text="{x:Static resx:Strings.WelcomeTitle}"
                       FontSize="26"
                       FontAttributes="Bold"
                       TextColor="#D3E3FC"
                       HorizontalTextAlignment="Center"
                       HorizontalOptions="Center"
                       VerticalOptions="Center" />

                <Label Text="{x:Static resx:Strings.WelcomeSubtitle}"
                       FontSize="16"
                       TextColor="#A9B8D3"
                       FontAttributes="Italic"
                       HorizontalTextAlignment="Center"
                       HorizontalOptions="Center" />

                <VerticalStackLayout Spacing="15"
                                     HorizontalOptions="Center"
                                     VerticalOptions="CenterAndExpand"
                                     Margin="0,30,0,0">
                    <Button x:Name="StartLessonButton"
                            Text="{x:Static resx:Strings.StartLessonsButton}"
                            Clicked="OnStartLessonClicked"
                            BackgroundColor="#3A4D9C"
                            TextColor="White"
                            FontSize="18"
                            CornerRadius="10"
                            HeightRequest="50"
                            WidthRequest="250"/>

                    <Button x:Name="OpenDictionaryButton"
                            Text="{x:Static resx:Strings.OpenDictionaryButton}"
                            Clicked="OnOpenDictionaryClicked"
                            BackgroundColor="#3A4D9C"
                            TextColor="White"
                            FontSize="18"
                            CornerRadius="10"
                            HeightRequest="50"
                            WidthRequest="250"/>

                    <Button x:Name="StartTestButton"
                            Text="{x:Static resx:Strings.StartTestingButton}"
                            Clicked="OnStartTestClicked"
                            BackgroundColor="#3A4D9C"
                            TextColor="White"
                            FontSize="18"
                            CornerRadius="10"
```

```
                            HeightRequest="50"
                            WidthRequest="250"/>
                    </VerticalStackLayout>
                </VerticalStackLayout>
            </ScrollView>

            <Button
                Clicked="OnSettingsClicked"
                WidthRequest="50"
                HeightRequest="50"
                BackgroundColor="Transparent"
                FontSize="24"
                HorizontalOptions="End"
                VerticalOptions="Start"
                Margin="10">
                <Button.ImageSource>
                    <FileImageSource File="settings_icon.png"/>
                </Button.ImageSource>
            </Button>
        </Grid>

</ContentPage>
```

Додаток В
MainPage.xaml.cs

```
using CommunityToolkit.Maui.Alerts;
using MauiApp1.Pages;

namespace MauiApp1
{
    public partial class MainPage : ContentPage
    {
        public MainPage()
        {
            InitializeComponent();
        }
        private async void OnStartLessonClicked(object sender, EventArgs e)
        {
            await Shell.Current.GoToAsync("//LessonsPage");
        }
        private async void OnOpenDictionaryClicked(object sender, EventArgs e)
        {
            await Shell.Current.GoToAsync("//DictionaryPage");
        }
        private async void OnStartTestClicked(object sender, EventArgs e)
        {
            await Shell.Current.GoToAsync("//QuizPage");
        }
        private async void OnSettingsClicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SettingsPage());
        }

    }
}
```

Додаток С
Word.cs

```
using SQLite;

namespace MauiApp1.Models
{
    public class Word
    {
        [PrimaryKey, AutoIncrement]
        public int Id { get; set; }

        public string WordText { get; set; }
        public string Meaning { get; set; }
    }
}
```

Додаток D

TestResult.cs

```
       using SQLite;
using System;

namespace MauiApp1.Models
{
    public class TestResult
    {
        [PrimaryKey, AutoIncrement]
        public int Id { get; set; }
        public int Score { get; set; }
        public int TotalQuestions { get; set; }
        public DateTime TestDate { get; set; }
        public string TestType { get; set; }
    }
}
```

Додаток Е

LessonProgress.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MauiApp1.Models
{
    public class LessonProgress
    {
        public string LessonName { get; set; }
        public bool IsCompleted { get; set; }
    }
}
```

Додаток F
ReadOnlyDatabase.cs

```
using SQLite;
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;
using MauiApp1.Models;
using Microsoft.Maui.Storage;

namespace MauiApp1.Services
{
    public class ReadOnlyDatabase
    {
        private readonly string _databaseFilePath;

        public ReadOnlyDatabase(string databaseFilePath)
        {
            _databaseFilePath = databaseFilePath;
            CreateTable();
        }

        private void CreateTable()
        {
            using (var connection = new SQLiteConnection(_databaseFilePath))
            {
                connection.CreateTable<Word>();
            }
        }
        public void AddWord(string wordText, string meaning)
        {
            try
            {
                using (var connection = new SQLiteConnection(_databaseFilePath))
                {
                    var word = new Word { WordText = wordText, Meaning = meaning };
                    connection.Insert(word);
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error while adding word: {ex.Message}");
            }
        }


        public async Task<List<Word>> GetWordsAsync()
        {
            var words = new List<Word>();

            try
            {
                using (var connection = new SQLiteConnection(_databaseFilePath))
                {
                    words = connection.Table<Word>().ToList();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error while reading database: {ex.Message}");
            }

            return words;
        }

        public void ClearDatabase()
        {
            try
```

```
            {
                using (var connection = new SQLiteConnection(_databaseFilePath))
                {
                    connection.DeleteAll<Word>();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error while clearing database: {ex.Message}");
            }
        }

        public static string GetDatabaseFilePath()
        {
            var file = Path.Combine(FileSystem.AppDataDirectory, "dictionary.db");

            if (!File.Exists(file))
            {
                var resourceFile = "MauiApp1.Resources.Raw.dictionary.db";
                using (var rawFileStream =
FileSystem.OpenAppPackageFileAsync(resourceFile).Result)
                {
                    byte[] rawFileBytes = new byte[rawFileStream.Length];
                    rawFileStream.Read(rawFileBytes, 0, rawFileBytes.Length);
                    File.WriteAllBytes(file, rawFileBytes);
                }
            }

            return file;
        }
    }
}
```

Додаток G

TestResultService.cs

```
using MauiApp1.Models;
using SQLite;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;

namespace MauiApp1.Services
{
    public class TestResultService
    {
        private static TestResultService _instance;
        private readonly SQLiteConnection _database;
        private const int MaxResults = 15;

        private TestResultService()
        {
            var databasePath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData),
"test_results.db");
            _database = new SQLiteConnection(databasePath);
            _database.CreateTable<TestResult>();
        }

        public static TestResultService Instance
        {
            get
            {
                if (_instance == null)
                {
                    _instance = new TestResultService();
                }
                return _instance;
            }
        }

        public async Task SaveResultAsync(TestResult result)
        {
            var results = _database.Table<TestResult>().ToList();


            if (results.Count >= MaxResults)
            {
                var oldestResult = results.OrderBy(r => r.TestDate).First();
                _database.Delete<TestResult>(oldestResult.Id);
            }

            _database.Insert(result);
        }

        public List<TestResult> GetAllResults()
        {
            return _database.Table<TestResult>().OrderByDescending(r =>
r.TestDate).ToList();
        }
    }
}
```

Додаток К

LogProgBar.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text.Json;
using System.Threading.Tasks;
using MauiApp1.Models;
using Microsoft.Maui.Storage;

namespace MauiApp1.Services
{
    public class LogProgBar
    {
        private readonly string _progressFilePath =
Path.Combine(FileSystem.AppDataDirectory, "lessonProgress.json");
        private List<LessonProgress> _lessonProgressList;

        public LogProgBar()
        {

            _lessonProgressList = new List<LessonProgress>();
        }


        public async Task InitializeProgressAsync()
        {
            if (File.Exists(_progressFilePath))
            {
                var json = await File.ReadAllTextAsync(_progressFilePath);
                _lessonProgressList =
JsonSerializer.Deserialize<List<LessonProgress>>(json) ?? new List<LessonProgress>();
            }
        }

        public async Task UpdateProgressAsync(string lessonName, bool isCompleted)
        {
            var lesson = _lessonProgressList.Find(l => l.LessonName == lessonName);
            if (lesson != null)
            {
                if (lesson.IsCompleted)
                {
                    Console.WriteLine($"Lesson {lessonName} already completed.");
                    return;
                }

                lesson.IsCompleted = isCompleted;
            }
            else
            {
                _lessonProgressList.Add(new LessonProgress { LessonName = lessonName,
IsCompleted = isCompleted });
            }

            await SaveProgressAsync();
        }

        public async Task<bool> GetLessonProgressAsync(string lessonName)
        {
            var lesson = _lessonProgressList.Find(l => l.LessonName == lessonName);
            return lesson?.IsCompleted ?? false;
        }


        public List<LessonProgress> GetLessonProgressList()
        {
            return _lessonProgressList;
```

```
        }

        private async Task SaveProgressAsync()
        {
            var json = JsonSerializer.Serialize(_lessonProgressList);
            await File.WriteAllTextAsync(_progressFilePath, json);
        }

        public async Task ResetProgressAsync()
        {
            _lessonProgressList.Clear();
            if (File.Exists(_progressFilePath))
            {
                await Task.Run(() => File.Delete(_progressFilePath));
            }
        }
    }
}
```

Додаток L
ITWordTest.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:resx="clr-namespace:MauiApp1.Resources.Lang"
             x:Class="MauiApp1.Pages.ITWordTest"
             Title="{x:Static resx:Strings.WordTestTitle}"
             BackgroundColor="#1E1E2F">

    <Grid RowDefinitions="*,Auto" Padding="20">
        <ScrollView Grid.Row="0">
            <VerticalStackLayout Spacing="20">
                <StackLayout Orientation="Horizontal" HorizontalOptions="FillAndExpand"
Spacing="10">
                    <Frame CornerRadius="7"
                           HeightRequest="15"
                           BackgroundColor="#E0E0E0"
                           Padding="0"
                           HorizontalOptions="FillAndExpand"
                           VerticalOptions="Start">
                        <ProgressBar x:Name="progressBar"
                                     Progress="0"
                                     HeightRequest="15"
                                     BackgroundColor="Transparent"
                                     ProgressColor="#0709FF" />
                    </Frame>
                    <Label x:Name="QuestionCounterLabel"
                           Text="1/12"
                           FontSize="18"
                           HorizontalOptions="Center"
                           VerticalOptions="Center"
                           TextColor="#ECF0F1"/>
                </StackLayout>

                <Frame Padding="15"
       BackgroundColor="#34495E"
       BorderColor="#BDC3C7"
       CornerRadius="10"
       HorizontalOptions="FillAndExpand"
                    VerticalOptions="StartAndExpand">
                    <Label x:Name="WordLabel"
      Text="{x:Static resx:Strings.WordLabelText}"
        FontSize="24"
        FontAttributes="Bold"
        TextColor="#ECF0F1"
        HorizontalOptions="Center"
        VerticalOptions="Center"
        LineBreakMode="WordWrap"
        HorizontalTextAlignment="Center"
        VerticalTextAlignment="Center"
        MaxLines="4" />
                </Frame>



                <Frame BackgroundColor="#2A2A3D"
       CornerRadius="12"
       Padding="10,0"
       HeightRequest="60"
       Margin="0,10"
       HorizontalOptions="FillAndExpand">
                    <Entry x:Name="AnswerEntry"
         Placeholder="{x:Static resx:Strings.AnswerPlaceholder}"
        FontSize="20"
        TextColor="#D3E3FC"
        PlaceholderColor="#A9B8D3"
        HorizontalOptions="FillAndExpand"
```

```
                VerticalOptions="Center"
            MinimumHeightRequest="60"
            MaxLength="100"/>
                </Frame>

                <Button x:Name="CheckAnswerButton"
                        Text="{x:Static resx:Strings.NextButton}"
                        Clicked="OnCheckAnswerButtonClicked"
                        BackgroundColor="#3A4D9C"
                        TextColor="White"
                        FontSize="18"
                        HorizontalOptions="FillAndExpand"
                        HeightRequest="55"
                        Margin="0,10"
                        CornerRadius="12"
                        BorderColor="#1E1E2F"
                        BorderWidth="2"/>

                <Label x:Name="ResultLabel"
                       Text=""
                       FontSize="18"
                       HorizontalOptions="Center"
                       Margin="0,10"
                       TextColor="#BDC3C7"/>

                <Label x:Name="IncorrectAnswerLabel"
                        Text="{x:Static resx:Strings.WrongAnswerMessage}"
                       FontSize="18"
                       TextColor="Red"
                       HorizontalOptions="Center"
                       VerticalOptions="Center"
                       IsVisible="False"
                       Margin="0,10"/>

                <Label x:Name="FinalResultLabel"
                         Text="{x:Static resx:Strings.TestCompletedMessage}"
                       FontSize="24"
                       HorizontalOptions="Center"
                       IsVisible="False"
                       TextColor="#A9B8D3"
                       Margin="0,20"/>

                <Button x:Name="ReturnToMenuButton"
                    Text="{x:Static resx:Strings.ReturnToMenuButton}"
                        Clicked="OnReturnToMenuButtonClicked"
                        IsVisible="False"
                        BackgroundColor="#3A4D9C"
                        TextColor="White"
                        FontSize="18"
                        HorizontalOptions="Center"
                        HeightRequest="50"/>

            </VerticalStackLayout>
        </ScrollView>
    </Grid>
</ContentPage>
```

Додаток М

ITWordTest.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using MauiApp1.Resources.Lang;
using MauiApp1.Models;
using MauiApp1.Data;
using MauiApp1.Services;
using MauiApp1.Strategies;
using Microsoft.Maui.Controls;
using Microsoft.Maui.Graphics;
using System.Threading.Tasks;

namespace MauiApp1.Pages
{
    public partial class ITWordTest : ContentPage
    {
        private List<Word> _words;
        private Word _currentWord;
        private int score = 0;
        private int currentQuestionIndex = 0;
        private const int totalQuestions = 24;
        private List<int> _usedIndexes = new List<int>();
        private readonly IAnswerCheckStrategy _answerCheckStrategy;

               private TestSession _currentSession;

        public ITWordTest()
        {
            InitializeComponent();

            _words = WordRepository.GetDefaultWords();
            _answerCheckStrategy = new SimpleCheckStrategy();

            _currentSession = new TestSession
            {
                TotalQuestions = totalQuestions,
                CorrectAnswers = 0,
                WrongAnswers = 0,
                FinishedAt = DateTime.Now
            };
        }

        protected override async void OnAppearing()
        {
            base.OnAppearing();

            await DatabaseService.Instance.SaveTestSessionAsync(_currentSession);

            await LoadNextWordAsync();
        }

        private async Task LoadNextWordAsync()
        {
            if (_words.Count == 0)
            {
                WordLabel.Text = "Слова відсутні";
                return;
            }

            if (currentQuestionIndex >= totalQuestions)
            {
                await ShowFinalResult();
                return;
            }
```

```
            string nextWordText = await
DatabaseService.Instance.GetNextWordAsync(_words, _usedIndexes);

            int index = _words.FindIndex(w => w.WordText == nextWordText);


            if (index == -1)
            {
                var random = new Random();
                do
                {
                    index = random.Next(_words.Count);
                } while (_usedIndexes.Contains(index));
            }

            _currentWord = _words[index];
            _usedIndexes.Add(index);

            WordLabel.Text = _currentWord.Meaning;
            AnswerEntry.Text = string.Empty;
            ResultLabel.Text = string.Empty;
            ResultLabel.TextColor = Colors.Black;

            currentQuestionIndex++;
            QuestionCounterLabel.Text = $"{currentQuestionIndex}/{totalQuestions}";
            progressBar.Progress = (double)currentQuestionIndex / totalQuestions;
        }

        private async void OnCheckAnswerButtonClicked(object sender, EventArgs e)
        {
            string userAnswer = AnswerEntry.Text?.Trim().ToLower();
            string correctAnswer = _currentWord.WordText.Trim().ToLower();

            bool isCorrect = _answerCheckStrategy.CheckAnswer(userAnswer,
correctAnswer);

            if (isCorrect)
            {
                ResultLabel.Text = Strings.Right;
                ResultLabel.TextColor = Colors.Green;
                score++;
                _currentSession.CorrectAnswers++;
            }
            else
            {
                ResultLabel.FormattedText = new FormattedString
                {
                    Spans =
                    {
                        new Span { Text = Strings.WrongAnswerMessage, TextColor =
Colors.Red, FontSize = 18 },
                        new Span { Text = correctAnswer, TextColor = Colors.Red,
FontSize = 20, FontAttributes = FontAttributes.Bold }
                    }
                };
                _currentSession.WrongAnswers++;
            }


            var answerRecord = new TestAnswer
            {
                Word = _currentWord.WordText,
                UserAnswer = userAnswer,
                IsCorrect = isCorrect,
                SessionId = _currentSession.Id,
                CreatedAt = DateTime.Now
            };
            await DatabaseService.Instance.SaveTestAnswerAsync(answerRecord);

            CheckAnswerButton.IsEnabled = false;
```

```
            await Task.Delay(1000);
            await LoadNextWordAsync();
            CheckAnswerButton.IsEnabled = true;
        }

        private async Task ShowFinalResult()
        {

            var testResult = new TestResult
            {
                Score = score,
                TotalQuestions = totalQuestions,
                TestDate = DateTime.Now,
                TestType = "Testing words System Dictionary"
            };
            await TestResultService.Instance.SaveResultAsync(testResult);


            _currentSession.FinishedAt = DateTime.Now;
            await DatabaseService.Instance.SaveTestSessionAsync(_currentSession);

            CheckAnswerButton.IsVisible = false;
            AnswerEntry.IsVisible = false;
            WordLabel.IsVisible = false;

            FinalResultLabel.IsVisible = true;
            ResultLabel.Text = $"{Strings.Res} {score}/{totalQuestions}";
            ResultLabel.TextColor = Color.FromArgb("#A9B8D3");

            ReturnToMenuButton.IsVisible = true;
        }

        private async void OnReturnToMenuButtonClicked(object sender, EventArgs e)
        {
            await Navigation.PopToRootAsync();
        }
    }
}
```

Додаток О
ResultsPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:resx="clr-namespace:MauiApp1.Resources.Lang"
             x:Class="MauiApp1.Pages.ResultsPage"
             Title="{x:Static resx:Strings.TitleResults}"
             BackgroundColor="#1E1E2F">

    <Grid RowDefinitions="*,Auto" Padding="20">
        <ScrollView Grid.Row="0">
            <VerticalStackLayout Spacing="20">

                <Label Text="{x:Static resx:Strings.TestRes}"
                       FontSize="22"
                       FontAttributes="Bold"
                       TextColor="#ECF0F1"
                       HorizontalOptions="Center" />

                <CollectionView x:Name="ResultsCollectionView"
                                SelectionMode="None"
                                ItemsSource="{Binding Results}">
                    <CollectionView.ItemTemplate>
                        <DataTemplate>
                            <Frame Padding="15"
                                   Margin="5"
                                   CornerRadius="10"
                                   BorderColor="#BDC3C7"
                                   BackgroundColor="#34495E"
                                   HasShadow="True">
                                <StackLayout Spacing="10">
                                    <Label Text="{Binding TestType}"
                                           FontSize="18"
                                           FontAttributes="Bold"
                                           TextColor="#ECF0F1"
                                           HorizontalOptions="Center" />

                                    <Label Text="{Binding Score, StringFormat={x:Static
resx:Strings.CorrectAnswersFormat}}"
                                    FontSize="16"
                                           TextColor="#BDC3C7"
                                           HorizontalOptions="Center" />

                                    <Label Text="{Binding TotalQuestions,
StringFormat={x:Static resx:Strings.TotalQuestionsFormat}}"
                                    FontSize="16"
                                            TextColor="#BDC3C7"
                                            HorizontalOptions="Center" />

                                    <Label Text="{Binding TestDate,
StringFormat='{0:dd/MM/yyyy HH:mm}'}"
                                           FontSize="14"
                                           TextColor="#BDC3C7"
                                           HorizontalOptions="Center" />
                                </StackLayout>
                            </Frame>
                        </DataTemplate>
                    </CollectionView.ItemTemplate>
                </CollectionView>

            </VerticalStackLayout>
        </ScrollView>


    </Grid>
</ContentPage>
```

Додаток Р

ResultsPage.xaml.cs

```
using MauiApp1.Models;
using MauiApp1.Services;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;

namespace MauiApp1.Pages
{
    public partial class ResultsPage : ContentPage
    {
        public ObservableCollection<TestResult> Results { get; set; }

        public ResultsPage()
        {
            InitializeComponent();

            var results = TestResultService.Instance.GetAllResults();

            foreach (var result in results)
            {
                Console.WriteLine($"TestType: {result.TestType}, Score: {result.Score},
TestDate: {result.TestDate}");
            }

            Results = new ObservableCollection<TestResult>(results);
            BindingContext = this;
        }
    }
}
```