МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ВИВЧЕННЯ ІНОЗЕМНИХ МОВ ЗАСОБАМИ FLUTTER»**

Виконав: студент 2 курсу, групи 8.1214-іпз-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
Є.В. Рибас
(ініціали та прізвище)

Керівник завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Решевська К.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

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

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Рибасу Євгенію Віталійовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка мобільного застосунку для вивчення іноземних мов засобами Flutter

керівник роботи Гребенюк Сергій Миколайович, д.т.н., професор
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 28 » лютого 2025 року № 326-с

2. Строк подання студентом роботи 05.12.2025 р.

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Основні теоретичні відомості.
2. Проєктування та розробка.
3. Тестування та оцінка ефективності застосунку.

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 07.03.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	07.04.2025	
2.	Збір вихідних даних.	09.05.2025	
3.	Обробка методичних та теоретичних	23.06.2025	
	джерел.		
4.	Розробка першого та другого розділу.	08.09.2025	
5.	Розробка третього розділу.	24.11.2025	
6.	Оформлення та нормоконтроль	01.12.2025	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	19.12.2025	

Студент _____ Є.В. Рибас
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.М. Гребенюк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

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

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка мобільного застосунку для вивчення іноземних мов засобами Flutter»: 59 с., 18 рис., 4 табл., 9 джерел.

АДАПТИВНЕ НАВЧАННЯ, БАЗА ДАНИХ, ВИВЧЕННЯ МОВ, ГЕЙМІФІКАЦІЯ, КРОСПЛАТФОРМЕНА РОЗРОБКА, МОБІЛЬНИЙ ЗАСТОСУНОК, FLUTTER, MATERIAL DESIGN, UX/UI ДИЗАЙН.

Об’єкт дослідження – процес проєктування, розробки та тестування мобільних застосунків для освітніх цілей.

Мета роботи: підвищення ефективності процесу вивчення іноземних мов шляхом розробки кросплатформеного мобільного застосунку з використанням фреймворку Flutter, що враховує сучасні методики навчання та гейміфікації.

Методи дослідження – аналіз, синтез, порівняння, проєктування, моделювання, тестування.

У роботі проведено аналіз сучасних методик вивчення мов та їх адаптації для мобільних платформ. Виконано огляд ринку існуючих додатків, виявлено їх переваги та недоліки. Обґрунтовано вибір фреймворку Flutter для реалізації проєкту. Спроєктовано архітектуру мобільного застосунку, розроблено дизайн користувацького інтерфейсу та реалізовано основний функціонал, що включає модулі навчання, систему відстеження прогресу та локальну базу даних. Проведено комплексне тестування застосунку та надано оцінку його ефективності.

SUMMARY

Master’s qualifying paper “Development of a Mobile Application for Foreign Languages Learning Using Flutter”: 59 pages, 18 figures, 4 tables, 9 references.

ADAPTIVE LEARNING, CROSS-PLATFORM DEVELOPMENT, DATABASE, FLUTTER, GAMIFICATION, LANGUAGE LEARNING, MATERIAL DESIGN, MOBILE APPLICATION, UX/UI DESIGN.

The object of the study is the process of designing, developing, and testing mobile applications for educational purposes.

The aim of the study is to increase the efficiency of the foreign language learning process by developing a cross-platform mobile application using the Flutter framework, which incorporates modern teaching methods and gamification.

The method of research are analysis, synthesis, comparison, design, modeling, testing.

The paper analyzes modern language learning techniques and their adaptation for mobile platforms. A review of the existing applications market was carried out, identifying their strengths and weaknesses. The choice of the Flutter framework for the project implementation was substantiated. The mobile application architecture was designed, the user interface design was developed, and the core functionality was implemented, including learning modules, a progress tracking system, and a local database. Comprehensive testing of the application was conducted, and its effectiveness was evaluated.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 7
1 Теоретичні основи розробки мобільних застосунків для навчання 9
1.1 Аналіз сучасних методик вивчення іноземних мов та їх адаптація до мобільних платформ 9
1.2 Огляд існуючих мобільних додатків для вивчення мов (Duolingo, Memrise, Babbel)..... 16
1.3 Обґрунтування вибору технології Flutter: порівняльний аналіз фреймворків та архітектурних особливостей 21
2 Проєктування та розробка мобільного застосунку..... 28
2.1 Аналіз вимог, визначення цільової аудиторії та проєктування архітектури додатку..... 28
2.2 Розробка інтерфейсу користувача та UX-дизайну з використанням Material Design..... 34
2.3 Імплементація основного функціоналу: модулі навчання, система прогресу, інтеграція бази даних 39
3 Тестування та оцінка ефективності застосунку “LearnZone”..... 44
3.1 Тестування функціоналу та забезпечення якості (unit, integration, widget тести) 44
3.2 Оптимізація продуктивності, споживання ресурсів та користувацького досвіду..... 49
3.3 Аналіз досягнутих результатів, оцінка функціональності та відповідності поставленим цілям..... 52
Висновки 57
Перелік посилань..... 59

ВСТУП

Актуальність теми. В епоху глобалізації та цифрової трансформації суспільства володіння іноземними мовами перетворюється з бажаної навички на критичну необхідність. Це відкриває нові можливості для освіти, кар'єрного зростання та міжкультурної комунікації. Паралельно з цим, мобільні технології стали невід'ємною частиною повсякденного життя. Смартфони та планшети надають унікальну платформу для навчання «на ходу» (microlearning), дозволяючи користувачам оптимізувати свій час та навчатися у будь-якому зручному місці.

Ринок мобільних освітніх додатків, зокрема для вивчення мов, демонструє стрімке зростання. Існуючі лідери, такі як Duolingo, Babbel та Memrise, довели ефективність гейміфікованого та інтерактивного підходів. Проте, незважаючи на їхній успіх, залишається простір для інновацій та задоволення специфічних потреб користувачів. Багато існуючих рішень страждають від браку гнучкості, недостатньої персоналізації навчального процесу або фокусуються лише на початкових рівнях володіння мовою.

Розробка нового мобільного застосунку, що поєднує перевірені методики викладання з гнучкістю та продуктивністю сучасних кросплатформених технологій, таких як Flutter, є актуальною науково-технічною задачею. Це дозволить створити доступний, ефективний та привабливий інструмент для широкої аудиторії, що прагне покращити свої мовні навички.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи є підвищення ефективності процесу вивчення іноземних мов шляхом розробки кросплатформеного мобільного застосунку з використанням фреймворку Flutter.

Для досягнення поставленої мети необхідно було вирішити наступні **задачі**:

- провести аналіз сучасних методик вивчення іноземних мов та дослідити можливості їх адаптації до мобільних платформ;
- виконати огляд та порівняльний аналіз існуючих мобільних додатків для вивчення мов, виявити їхні функціональні переваги та недоліки;
- обґрунтувати вибір технологічних засобів розробки, зокрема фреймворку Flutter, шляхом порівняння з альтернативними рішеннями;
- сформулювати функціональні та нефункціональні вимоги до застосунку, визначити цільову аудиторію та спроєктувати архітектуру системи;
- розробити інтуїтивно зрозумілий та естетично привабливий дизайн користувацького інтерфейсу (ui/ux) на основі принципів material design;
- реалізувати основний функціонал застосунку, включаючи навчальні модулі, систему відстеження прогресу та інтеграцію з базою даних;
- провести комплексне тестування розробленого програмного продукту для забезпечення його якості та надійності;
- проаналізувати досягнуті результати та оцінити відповідність розробленого застосунку поставленим цілям.

Об’єкт і предмет дослідження. Об’єктом дослідження є процес проєктування, розробки та тестування мобільних застосунків для освітніх цілей. Предметом дослідження є методики та інструментальні засоби розробки кросплатформеного мобільного застосунку для вивчення іноземних мов.

Практичне значення одержаних результатів. Розроблений мобільний застосунок є готовим програмним продуктом, який можна використовувати як інструмент для самостійного вивчення іноземних мов. Результати дослідження, зокрема проведений аналіз технологій та спроєктована архітектура, можуть бути корисними для розробників, що працюють над створенням аналогічних освітніх програмних продуктів.

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ МОБІЛЬНИХ ЗАСТОСУНКІВ ДЛЯ НАВЧАННЯ

1.1 Аналіз сучасних методик вивчення іноземних мов та їх адаптація до мобільних платформ

Процес вивчення іноземної мови зазнав значних трансформацій за останні десятиліття. Перехід від традиційних граматико-перекладних методів до комунікативних та інтерактивних підходів змінив фокус навчання з механічного запам’ятовування правил на розвиток практичних навичок спілкування. Сучасні мобільні технології стали каталізатором для подальшої еволюції цих методик, пропонуючи нові форми подачі та засвоєння матеріалу.

Комунікативний підхід (Communicative Language Teaching – CLT) – CLT є домінуючим методом у сучасному викладанні мов [1]. Його основна ідея полягає в тому, що мова вивчається для комунікації. Навчання зосереджено на виконанні комунікативних завдань, а не на вивченні граматики як самоцілі.

Адаптація до мобільних платформ: мобільні додатки ідеально підходять для реалізації CLT. Це може бути реалізовано через:

- інтерактивні діалоги з віртуальним співрозмовником (чат-боти);
- вправи на аудіювання з реальними життєвими сценаріями (замовлення в кафе, купівля квитка);
- можливість запису власної вимови та її порівняння з еталонною.

Метод занурення (Immersion Method) – цей метод передбачає максимальне «занурення» студента в мовне середовище. В ідеалі, все спілкування та споживання контенту відбувається виключно цільовою мовою.

Адаптація до мобільних платформ: мобільний додаток може симулювати середовище занурення:

- інтеграція автентичного контенту: новинних статей, подкастів, відеороликів;

– налаштування інтерфейсу додатку виключно цільовою мовою;
– вправи, що вимагають реакції на ситуації без використання рідної мови для перекладу.

Гейміфікація (Gamification) – це використання ігрових механік (бали, рівні, нагороди, таблиці лідерів) у неігровому контексті для підвищення мотивації та залученості [2, 3].

Адаптація до мобільних платформ: це одна з найсильніших сторін мобільних додатків:

– система «очок досвіду» (xp) за виконані уроки;
– «стріки» (streaks) – безперервна серія днів занять;
– відкриття нових рівнів або бонусного контенту;
– віртуальні нагороди (ачівки, бейджі) за досягнення.

Мікронавчання (Microlearning) – методика подачі навчального матеріалу короткими, концентрованими блоками (зазвичай 3-10 хвилин). Це ідеально відповідає сучасному ритму життя та моделям використання смартфонів.

Адаптація до мобільних платформ: мобільні додатки є нативною платформою для мікронавчання:

– короткі уроки, які можна пройти в черзі або транспорті;
– флеш-картки (flashcards) для швидкого повторення слів;
– короткі тести та вікторини.

Адаптивне навчання (Adaptive Learning) – використання алгоритмів для персоналізації навчального процесу на основі індивідуальних успіхів, помилок та темпу студента. Система аналізує відповіді користувача та динамічно підлаштовує складність і тип наступних завдань.

Адаптація до мобільних платформ:

– система інтервального повторення (spaced repetition system - srs), яка показує складні слова частіше, а легкі – рідше [4];
– автоматичне налаштування складності граматичних вправ.

Для систематизації проаналізованих методик та їх придатності для мобільної реалізації, складемо порівняльну таблицю (див. табл. 1.1).

Таблиця 1.1 – Порівняльний аналіз методик вивчення

Методика	**Ключовий фокус**	**Мобільна адаптивність**	**Приклад реалізації**
Комунікативна	Спілкування	Висока	Чат-боти, діалоги
Гейміфікація	Мотивація	Дуже висока	Бали, рівні, бейджі
Мікронавчання	Короткі сесії	Дуже висока	Флеш-картки, тести
Адаптивна	Персоналізація	Середня	Алгоритми (SRS)
Занурення	Контекст	Середня	Медіа-контент

Метод Total Physical Response (TPR). Поряд з розглянутими вище методиками, окремої уваги заслуговує метод повної фізичної реакції (Total Physical Response), розроблений Джеймсом Ашером у 1960-х роках. Цей підхід базується на координації мови та фізичного руху, що особливо ефективно працює на початкових етапах вивчення мови. Основна ідея TPR полягає в тому, що учні спочатку слухають команди цільовою мовою та виконують відповідні дії, перш ніж почати активно говорити.

Нейропсихологічні дослідження підтверджують, що фізична активність під час навчання активізує різні ділянки мозку, що сприяє кращому запам’ятовуванню. Моторна кора головного мозку тісно пов’язана з мовними центрами (зона Брока), тому асоціація нового слова з конкретним рухом створює міцніші нейронні зв’язки порівняно з пасивним запам’ятовуванням через візуальні або аудіальні стимули.

Адаптація до мобільних платформ. Незважаючи на певні обмеження мобільного формату (користувач часто знаходиться в громадських місцях, де фізичні рухи недоцільні), TPR може бути частково інтегрований через:

- міні-ігри з елементами жестового управління (swipe, tap, shake), де конкретний жест відповідає певному дієслову чи дії;
- AR-функціонал (доповнена реальність), де користувач може через камеру смартфону взаємодіяти з віртуальними об’єктами, виконуючи команди (наприклад, “Put the apple on the table”);
- голосові команди з необхідністю не лише промовити фразу, але й виконати відповідну дію в реальному світі (тренування імперативу).

Метод “Silent Way” (Мовчазний шлях). Цей радикальний підхід, створений Калебом Гаттеньо, передбачає мінімальне втручання викладача в процес навчання. Учні самостійно «відкривають» правила мови через експериментування з кольоровими картками (Cuisenaire rods) та іншими матеріалами. Вчитель говорить якомога менше, спонукаючи студентів до самостійного пошуку рішень.

Хоча цей метод рідко використовується в чистому вигляді, його принцип «навчання через відкриття» (discovery learning) є дуже цінним. Коли учень самостійно формулює граматичне правило, а не отримує його в готовому вигляді, воно краще засвоюється та довше зберігається в довготривалій пам’яті.

Адаптація до мобільних платформ:

– інтерактивні головоломки, де потрібно самостійно скласти граматичну конструкцію, експериментуючи з послідовністю слів;
– система підказок «від меншого до більшого» – спочатку мінімальна підказка, потім більш детальна, заохочуючи користувача спробувати вирішити завдання самостійно;
– вбудований «режим експерименту», де неправильна відповідь не карається, а показуються наслідки (наприклад, як змінюється сенс речення при зміні порядку слів).

Suggestopedia (Сугестопедія). Метод болгарського психотерапевта Георгія Лозанова базується на створенні розслабленої, комфортної атмосфери навчання з використанням музики, яскравих візуальних образів та позитивних сугестивних установок. Лозанов стверджував, що в стані релаксації мозок здатний засвоювати інформацію набагато ефективніше, оскільки знімаються психологічні бар’єри та страх помилки.

Класична сугестопедія передбачає заняття в спеціально обладнаних кімнатах з м’якими кріслами, приглушеним світлом та класичною музикою (переважно бароко – Бах, Вівальді). Хоча повноцінна реалізація цього методу в мобільному додатку неможлива, окремі його елементи можуть значно покращити UX:

- режим «вечірнього навчання» з темною темою інтерфейсу, заспокійливою фоновою музикою та плавними анімаціями без різких переходів;
- відсутність негативних візуальних сигналів при помилці (замість червоного “X” – нейтральна підказка);
- голосове супроводження завдань спокійним, підбадьорюючим тоном;
- інтеграція з музичними сервісами (Spotify, Apple Music) для автоматичного програвання релаксаційних треків під час занять.

Метод “Lexical Approach” (Лексичний підхід). Майкл Льюїс у 1990-х роках запропонував радикально переглянути роль лексики у вивченні мови. Замість того, щоб вивчати окремі слова та потім комбінувати їх за граматичними правилами, Льюїс стверджує, що мова складається з готових «блоків» – усталених словосполучень (collocations, chunks), ідіом та фразових дієслів. Саме ці блоки є природними одиницями мови, якими оперують носії.

Дослідження корпусної лінгвістики підтверджують, що близько 70% усного та письмового мовлення складається з повторюваних патернів. Наприклад, носії англійської ніколи не кажуть “do a photo”, а завжди “take a photo”; не “strong tea” змінюється на “powerful tea”, а залишається саме “strong tea”. Вивчення таких усталених зв’язків дозволяє говорити природніше та швидше формулювати думки.

Адаптація до мобільних платформ:

- база даних «колокацій» – система пропонує вивчати не окреме слово “make”, а серію фраз: “make a decision”, “make progress”, “make a mistake”;
- контекстне вивчення слів – нове слово завжди з’являється в природному контексті, а не ізольовано;
- вправи на підстановку – “I need to ___ a decision” (make / do / take);
- аналіз помилок – коли користувач обирає неправильну колокацію, система пояснює, чому саме така комбінація неприродна для носіїв мови.

Content and Language Integrated Learning (CLIL). CLIL – це підхід, при якому іноземна мова вивчається не як окремий предмет, а як засіб вивчення інших дисциплін (історії, географії, біології). Цей метод широко використовується в європейських школах та університетах. Його основна перевага – створення автентичного контексту для використання мови: замість штучних діалогів типу “Hello, how are you?” студенти обговорюють реальні теми, що їх цікавлять.

Адаптація до мобільних платформ:

- тематичні модулі з реальним контентом: «Історія Другої світової війни англійською», «Основи економіки німецькою»;
- інтеграція з Wikipedia, TED Talks, документальними відео;
- вправи, побудовані навколо автентичних матеріалів (прочитати статтю про екологію, відповісти на питання, обговорити проблему);
- «Мовне занурення через хобі» – користувач вибирає свою сферу інтересів (кулінарія, програмування, мандрівки), і додаток генерує контент відповідної тематики цільовою мовою.

Task-Based Language Teaching (TBLT). Цей комунікативний підхід структурує навчання навколо виконання конкретних завдань (tasks), а не вивчення граматичних структур. Завдання має відповідати трьом критеріям: фокус на значенні (а не на формі), наявність інформаційного розриву (information gap), який потрібно заповнити, та чітко визначений результат. Наприклад, замість вправи «Вставте правильну форму дієслова» студенти отримують завдання «Спланувати маршрут подорожі» або «Порівняти два продукти та вибрати кращий».

Адаптація до мобільних платформ:

- рольові сценарії: користувач грає роль туриста, який має забронювати готель, замовити їжу, запитати дорогу (AI-чат-бот виконує роль співрозмовника);
- проблемні ситуації: «Ваш рейс скасували. Поспілкуйтеся з працівником авіакомпанії та знайдіть альтернативу»;

– проєктна робота: «Створіть презентацію про вашу країну» – користувач записує аудіо/відео, додаток аналізує мову та дає фідбек.

Psycholinguistic Perspective: Working Memory and Language Acquisition. Сучасні дослідження в галузі когнітивної психології та нейролінгвістики вказують на критичну роль робочої пам'яті (working memory) у процесі засвоєння мови. Робоча пам'ять – це обмежена за обсягом система, яка тримає та обробляє інформацію в режимі реального часу. Згідно з моделлю Баддлі та Хітча, вона складається з кількох компонентів:

– фонологічна петля (phonological loop) – відповідає за утримання звукової інформації (слів, звуків);
– візуально-просторовий блокнот (visuospatial sketchpad) – обробляє візуальні образи та просторові відношення;
– центральний виконавчий компонент (central executive) – координує роботу інших компонентів та приймає рішення.

Дослідження показують, що індивідуальні відмінності в обсязі робочої пам'яті (особливо фонологічної петлі) є одним з найсильніших предикторів успіху у вивченні другої мови. Люди з більшим обсягом робочої пам'яті краще запам'ятовують нові слова, швидше засвоюють граматичні структури та успішніше розуміють усне мовлення.

Практичні імплікації для мобільного навчання:

– обмеження когнітивного навантаження (Cognitive Load Theory): вправи повинні бути розроблені так, щоб не перевантажувати робочу пам'ять (це означає: подавати інформацію невеликими порціями (3-5 нових слів за раз), уникати зайвих візуальних/звукових відволікань під час завдання, використовувати техніку "chunking" – групування інформації в осмислені блоки);
– тренування робочої пам'яті: спеціальні вправи типу n-back (запам'ятай послідовність слів та повтори n елементів назад), shadowing (повторення за диктором з мінімальною затримкою), digit span (запам'ятовування послідовності цифр);

– мультимодальна подача матеріалу: одночасне використання тексту, зображень та звуку активує різні компоненти робочої пам’яті, що покращує кодування інформації в довготривалу пам’ять (dual coding theory).

Отже, аналіз показав, що сучасні мобільні платформи є потужним інструментом для реалізації найефективніших методик вивчення мов. Найбільший потенціал мають додатки, що поєднують кілька підходів: комунікативні завдання, подані у форматі мікронавчання, підсилені механіками гейміфікації та персоналізовані за допомогою алгоритмів адаптивного навчання. Саме на такому поєднанні і буде базуватися проєктований у даній роботі застосунок.

1.2 Огляд існуючих мобільних додатків для вивчення мов (Duolingo, Memrise, Babbel)

Світовий ринок мобільних додатків для вивчення мов демонструє безпрецедентне зростання, зумовлене глобалізацією та потребою в доступній освіті. Згідно з останніми дослідженнями, цей сегмент є одним з найбільш прибуткових у категорії «Освіта» магазинів App Store та Google Play. Проте, велика кількість пропозицій не завжди означає якість, і багато додатків страждають від методичних помилок або слабкого залучення користувачів. Для розуміння сучасних стандартів якості та виявлення потенційних ніш для розробки, доцільно провести детальний аналіз трьох лідерів ринку: Duolingo, Memrise та Babbel, кожен з яких представляє відмінний підхід до навчання.

Duolingo є беззаперечним лідером ринку, чия стратегія базується на агресивній та глибокій гейміфікації навчального процесу. Додаток перетворює вивчення мови на гру, використовуючи механіки, характерні для мобільних ігор жанру RPG та аркад. Основою успіху Duolingo є використання короткострокових циклів зворотного зв’язку, коли користувач отримує

миттєву винагороду (звукові ефекти, анімація, бали) за кожну правильну відповідь, що стимулює викид дофаміну. Механіка «стріків» (streaks) – безперервних серій днів навчання – експлуатує психологічний ефект страху втрати, змушуючи користувачів повертатися до додатку щодня. Основною перевагою Duolingo є надзвичайно низький поріг входу та здатність утримувати увагу користувача, що робить його ідеальним для початківців. Проте, цей підхід має суттєві недоліки: надмірне спрощення матеріалу, фокус на перекладі ізольованих речень, які часто є семантично дивними, та відсутність систематичних пояснень граматики, що обмежує прогрес на середніх та високих рівнях володіння мовою.

Memrise, на відміну від Duolingo, робить ставку на ефективне запам’ятовування лексики, використовуючи науково обґрунтовані методики мнемотехніки та інтервальних повторень (Spaced Repetition System – SRS). Алгоритм додатка відстежує, як давно і наскільки успішно користувач повторив те чи інше слово, і пропонує його для повторення саме в момент, коли ризик забування є найвищим. Ключовою особливістю Memrise є функція “Learn with Locals”, яка інтегрує в навчальний процес тисячі коротких відеокліпів з носіями мови, записаних у реальних життєвих ситуаціях. Це дозволяє користувачам звикати до реального темпу мовлення, акцентів та невербальної комунікації, що часто ігнорується в інших додатках. Недоліком Memrise є його вузька спеціалізація на лексиці; граматичні структури та навички побудови речень часто залишаються поза увагою, що робить додаток чудовим допоміжним інструментом, але недостатнім для повноцінного вивчення мови.

Babbel позиціонує себе як більш традиційний та академічний інструмент, орієнтований на серйозну аудиторію, яка готова платити за якісний контент. Методика Babbel базується на дидактичному підході, де кожен урок має чітку мету (наприклад, навчитися замовляти їжу в ресторані) і супроводжується детальними граматичними поясненнями. Додаток активно використовує діалоги, максимально наближені до реальних ситуацій, і

пропонує вправи на тренування вимови за допомогою технології розпізнавання мови. Модель монетизації Babbel (платна підписка без безкоштовної версії) фільтрує аудиторію, залишаючи лише вмотивованих користувачів. Хоча Babbel забезпечує глибше розуміння мови, його інтерфейс та структура уроків можуть здатися менш захопливими порівняно з Duolingo, що вимагає від користувача вищого рівня самодисципліни (див. табл. 1.2).

Таблиця 1.2 – Характеристика додатків-аналогів

Критерій	**Duolingo**	**Memrise**	**Babbel**
Фокус	Гра/Мотивація	Лексика/Пам’ять	Граматика/Діалоги
Гейміфікація	Висока	Середня	Низька
Контент	Універсальний	Картки слів	Практичний
Вартість	Безкоштовно	Freemium	Платний
Аудиторія	Масова	Візуали	Свідомі учні

Детальний аналіз монетизаційних стратегій конкурентів. Розуміння бізнес-моделей провідних гравців ринку є критично важливим для планування власної стратегії монетизації. Розглянемо детальніше фінансові аспекти роботи Duolingo, Memrise та Babbel.

Duolingo: freemium-модель з акцентом на рекламі. Duolingo є яскравим прикладом успішної freemium-моделі. Згідно з фінансовими звітами компанії за 2024 рік, близько 8% користувачів мають платну підписку Duolingo Plus (Super), а решта 92% користуються безкоштовною версією з рекламою. Розглянемо основні джерела доходу.

Реклама (≈50% доходу): Duolingo інтегрує рекламу дуже обережно, щоб не відштовхнути користувачів. Реклама з’являється між уроками (interstitial ads), але не під час самого процесу навчання. Також є нативна реклама – спонсоровані «лінготи» (внутрішньоігрова валюта) від партнерів.

Підписка Duolingo Super (≈45% доходу): $6.99/місяць або $83.99/рік. Переваги включають: відсутність реклами, необмежену кількість «сердець»

(можливість помилятися), офлайн-доступ до уроків, персоналізовані практичні сесії, прогрес-тести для відстеження рівня.

Duolingo English Test (≈5% доходу): окремий продукт – сертифікаційний тест англійської мови ($49 за спробу), який визнається багатьма університетами як альтернатива TOEFL/IELTS. Це демонструє стратегію горизонтальної диверсифікації доходів.

Ключовою метрикою успіху Duolingo є DAU/MAU ratio (співвідношення денних активних користувачів до місячних), яке становить близько 28% – один з найвищих показників у галузі освітніх додатків. Це означає високий рівень «звички» (habit formation) – користувачі відкривають додаток не епізодично, а регулярно.

Memrise: гібридна модель з фокусом на преміум-контент. Memrise змінив свою стратегію у 2019 році, перейшовши від повністю безкоштовної моделі до freemium. На відміну від Duolingo, безкоштовна версія Memrise досить обмежена:

– доступ лише до офіційних курсів (створених командою Memrise);
– обмежена кількість уроків на день;
– відсутність функції “Learn with Locals” (відео з носіями).

Memrise Pro (підписка) коштує $8.99/місяць або $89.99/рік та надає:

– необмежений доступ до всіх курсів;
– режим “Difficult Words” – персоналізовані сесії повторення складних слів;
– граматичні боти для практики діалогів;
– режим офлайн;
– детальну статистику прогресу.

Конверсія з безкоштовної версії в платну у Memrise вища, ніж у Duolingo (≈12%), оскільки обмеження відчутніші. Компанія також активно експериментує з B2B-сегментом, пропонуючи корпоративні ліцензії для компаній, які хочуть навчати своїх працівників іноземним мовам.

Babbel: Чисто платна модель з акцентом на якість. Babbel з самого початку обрав стратегію преміум-продукту без безкоштовної версії (є лише

пробна версія – перший урок кожного курсу). Це позиціонує додаток як серйозний освітній інструмент для вмотивованої аудиторії. Ціноутворення:

– 1 місяць: $13.95;
– 3 місяці: $9.95/місяць ($29.85 разом);
– 6 місяців: $8.45/місяць ($50.70 разом);
– 12 місяців: $6.95/місяць ($83.40 разом);
– довічний доступ (Lifetime): одноразово $299.99.

Babbel активно інвестує в маркетинг через партнерства з корпораціями та освітніми інституціями. Компанія також створила окремий продукт Babbel for Business для корпоративного сегменту. Важливо відзначити, що Babbel не використовує агресивні retention-механіки (стріки, лідерборди), покладаючись на якість контенту та самомотивацію користувачів (див. табл. 1.3).

Таблиця 1.3 – Порівняльна таблиця ARPU (Average Revenue Per User)

Додаток	**ARPU(річний)**	**Conversion Rate**	**Retention (90 днів)**
Duolingo	15-20$	8%	35%
Memrise	25-30$	12%	28%
Babbel	85-95$	N/A (платний)	45%

Emerging Competitors: Busuu, Mondly, Rosetta Stone. Окрім трьох лідерів, на ринку активно розвиваються інші гравці, кожен з яких пропонує унікальну цінність.

Busuu позиціонує себе як «соціальну мережу для вивчення мов». Його ключова особливість – можливість відправляти письмові та усні вправи на перевірку носіям мови, які вивчають вашу рідну мову. Це створює взаємовигідну екосистему мовного обміну. Busuu також пропонує офіційні сертифікати, прив’язані до рівнів CEFR (A1-C2), що визнаються роботодавцями.

Mondly робить ставку на технології віртуальної (VR) та доповненої (AR) реальності. Додаток пропонує унікальний досвід: навчання через віртуальні

симуляції реальних ситуацій (замовлення в ресторані, переговори в офісі) з використанням VR-гарнітур. Хоча VR-функціонал доступний лише власникам спеціалізованих пристроїв, це демонструє потенційний напрямок еволюції галузі.

Rosetta Stone – ветеран ринку (заснована у 1992 році) намагається адаптуватися до мобільної ери. Їхня методика “Dynamic Immersion” базується на візуальних асоціаціях без перекладу рідною мовою. Користувачі бачать зображення та чують слово/фразу цільовою мовою, самостійно виводячи значення. Це повільніше, ніж традиційні методи, але створює міцніші асоціації.

Проведений аналіз лідерів ринку демонструє чітку сегментацію: Duolingo захоплює масову аудиторію через розвагу, Memrise фокусується на лексичному запасі, а Babbel задовольняє потреби в систематичному вивченні. Однак, на ринку існує помітний розрив – відсутність рішення, яке б ефективно поєднувало високу мотиваційну складову гейміфікації з глибиною граматичного та комунікативного навчання. Користувачі часто змушені обирати між «цікаво, але поверхнево» та «корисно, але нудно». Це створює перспективну нішу для розробки мобільного застосунку, який інтегрує найкращі практики конкурентів: адаптивні алгоритми повторення, ігрові механіки утримання уваги та структурований підхід до граматики, що стане основою для проєктування власного програмного продукту.

1.3 Обґрунтування вибору технології Flutter: порівняльний аналіз фреймворків та архітектурних особливостей

Вибір технологічного стеку для розробки мобільного застосунку є критичним етапом, що визначає не лише швидкість виходу продукту на ринок (Time-to-Market), але й його подальшу масштабованість, продуктивність та вартість підтримки. В умовах сучасного ринку, де домінування поділене між

платформами iOS та Android, розробники стикаються з дилемою вибору між нативною розробкою та кросплатформеними рішеннями. Кожен з підходів має свої архітектурні особливості, які суттєво впливають на кінцевий результат.

Нативна розробка, що передбачає створення окремих додатків мовами Swift (для iOS) та Kotlin (для Android), традиційно вважається еталоном продуктивності та надійності. Цей підхід надає розробникам прямий доступ до всіх апаратних можливостей пристрою (камери, GPS, сенсорів) та найновіших API операційної системи без будь-яких посередників. Інтерфейс нативних додатків завжди виглядає і відчувається «рідним» для платформи, що забезпечує найкращий користувацький досвід (UX). Однак, головним недоліком цього підходу є необхідність підтримувати дві окремі кодові бази. Це фактично подвоює час розробки, витрати на тестування та підтримку, а також вимагає залучення вузькопрофільних спеціалістів для кожної платформи, що може бути критичним обмеженням для стартапів або дипломних проєктів з обмеженими ресурсами.

Альтернативою є кросплатформені фреймворки, серед яких найбільшої популярності набули React Native від Meta та Flutter від Google. React Native, базуючись на мові JavaScript, використовує архітектуру «моста» (Bridge). У цьому підході бізнес-логіка виконується в JavaScript-потоці, а для відображення інтерфейсу фреймворк звертається через «міст» до нативних компонентів платформи (OEM widgets). Це дозволяє додатку виглядати як нативний, але створює вузьке місце (bottleneck) у продуктивності: при складних анімаціях або інтенсивній передачі даних через «міст» можуть виникати затримки та падіння частоти кадрів (FPS). Крім того, залежність від нативних компонентів означає, що вигляд додатку може відрізнятися на різних версіях ОС, що ускладнює забезпечення візуальної консистентності.

Flutter пропонує кардинально інший архітектурний підхід, який усуває проблему «моста» [6]. Замість використання нативних компонентів, Flutter використовує власний високоефективний графічний рушій Skia (на якому також побудований Google Chrome), що дозволяє малювати кожен піксель

інтерфейсу самостійно. Програми на Flutter компілюються безпосередньо в машинний код (ARM або x86) за допомогою мови Dart, що забезпечує продуктивність, близьку до нативної. Концепція «Все є віджет» дозволяє створювати складні, кастомні інтерфейси, які виглядають і працюють ідентично на iOS та Android, незалежно від версії системи. Додатковою перевагою є функція Stateful Hot Reload, яка дозволяє розробникам бачити зміни в коді миттєво без перезавантаження всього додатку, що експоненціально прискорює процес розробки та експериментування з UI (див. табл. 1.4).

Таблиця 1.4 – Порівняння технологій розробки

Параметр	**Native (Kotlin/Swift)**	**React Native**	**Flutter**
Швидкість	Максимальна	Середня	Висока
Кодова база	Окрема (2 шт)	Спільна (JS)	Спільна (Dart)
UI-стабільність	Залежить від ОС	Залежить від ОС	Ідентична всюди
Розробка	Довго	Швидко	Дуже швидко
API	Прямий доступ	Через міст	Через канали

Глибинний технічний аналіз Flutter: архітектура та переваги. Для повного розуміння переваг Flutter необхідно розглянути його архітектуру на рівні компонентів та те, як вона відрізняється від конкурентів. Архітектура Flutter побудована за принципом багатошарової моделі, де кожен шар відповідає за певний аспект функціонування системи.

Шари архітектури Flutter. Верхній шар архітектури, відомий як Framework Layer, написаний мовою Dart і є тим рівнем, з яким безпосередньо працює розробник. Цей шар включає Material та Cupertino віджети, що реалізують дизайн-системи Android та iOS відповідно. Крім того, Framework Layer містить Rendering layer як абстракцію над графічним рушієм, Widgets layer для композиційної моделі UI, а також Gestures та Animation API для обробки користувацьких взаємодій та створення анімацій.

Серцем Flutter є Engine Layer, написаний на C++ для забезпечення максимальної продуктивності. Основним компонентом цього шару є Skia Graphics Engine, що відповідає за растеризацію та перетворення векторних команд у пікселі. Engine Layer також включає Dart Runtime як віртуальну машину для виконання коду, складну систему рендерингу тексту з підтримкою Unicode та двонаправленого тексту для таких мов як арабська та іврит, а також Platform channels як механізм комунікації з нативним кодом iOS та Android.

Найнижчим рівнем є Embedder Layer, що представляє собою платформоспецифічний код для інтеграції Flutter engine у конкретну операційну систему. Окремі реалізації embedder існують для Android на базі Java та Kotlin, iOS на базі Objective-C та Swift, вебплатформи через JavaScript, а також десктопних систем Windows, macOS та Linux.

На відміну від React Native або Xamarin, які використовують нативні UI-компоненти операційної системи через посередницький «міст», Flutter малює кожен піксель інтерфейсу самостійно за допомогою графічного рушія Skia. Це архітектурне рішення забезпечує абсолютну візуальну консистентність між платформами, оскільки додаток виглядає ідентично незалежно від версії операційної системи. Така незалежність особливо важлива для підтримки старіших пристроїв, де нові Material Design компоненти доступні навіть на Android 6.0 без необхідності оновлення системи.

Відсутність залежності від нативних компонентів надає розробникам можливість створювати повністю кастомізовані інтерфейси без обмежень, які накладають стандартні UI-елементи платформ. Більш того, відсутність «мосту» між бізнес-логікою та шаром відображення означає, що інтенсивні операції, такі як анімації з частотою 60 кадрів на секунду, виконуються без додаткових накладних витрат на серіалізацію та передачу даних.

Dart – недооцінена мова програмування. Вибір мови Dart для Flutter часто викликає запитання серед розробників, оскільки вона значно менш популярна порівняно з JavaScript або Java. Проте детальний аналіз показує, що Dart має унікальні характеристики, які роблять його ідеально підходящим для розробки користувацьких інтерфейсів.

Dart розроблявся компанією Google з чітким фокусом на створення інтерфейсів, що відображається в його синтаксисі. Декларативний підхід мови дозволяє дуже компактно описувати вкладені структури віджетів, що особливо важливо при побудові складних UI. Наприклад, створення форми з валідацією полів займає значно менше рядків коду порівняно з імперативними підходами в Java або Swift, при цьому код залишається читабельним та зрозумілим.

Унікальною особливістю Dart є підтримка двох режимів компіляції, що оптимізовані для різних етапів розробки. Під час розробки використовується JIT компіляція, що забезпечує роботу функції Hot Reload, коли зміни в коді миттєво відображаються на екрані без повного перезавантаження додатку. Для production-білдів застосовується AOT компіляція, при якій код перетворюється безпосередньо в машинний код ARM або x86, що забезпечує швидкість роботи, порівнянну з повністю нативними додатками.

Починаючи з версії Dart 2.12, випущеної у 2021 році, мова запровадила систему sound null safety. Це статична система типів, яка на етапі компіляції гарантує відсутність помилок NullPointerException, що є причиною приблизно 70% всіх крешів у мобільних додатках згідно з дослідженнями Google. Така превентивна система значно підвищує надійність кінцевого продукту.

Dart також має вбудовану підтримку асинхронного програмування через конструкції Future та Stream, що критично важливо для мобільної розробки з її численними мережевими запитами та операціями доступу до бази даних. Для CPU-інтенсивних задач мова пропонує концепцію Isolates, окремих потоків виконання з власним сегментом пам’яті, які комунікують через систему повідомлень за принципом actor model. Такий підхід є безпечнішим та простішим у використанні порівняно з класичними threads у Java, де розробник повинен вручну управляти синхронізацією та блокуванням ресурсів.

Екосистема та інструментарій Flutter. Центральним елементом екосистеми Flutter є пакетний менеджер Pub.dev, що виконує роль аналогічну npm для JavaScript або Maven для Java. Станом на кінець 2024 року

репозиторій містить понад 50,000 пакетів, що охоплюють практично всі аспекти мобільної розробки. Найбільш затребованими категоріями є рішення для управління станом додатку, серед яких Provider, Bloc, Riverpod та GetX, HTTP-клієнти як dio та http для роботи з мережевими запитами, бази даних включаючи sqflite, hive та isar, пакети для інтеграції з Firebase, а також великий вибір UI-компонентів для анімацій, графіків та кастомних віджетів.

Flutter DevTools представляє собою комплексний набір інструментів для діагностики та оптимізації додатків. Widget Inspector надає візуальне представлення дерева віджетів з можливістю інспекції властивостей кожного елемента в реальному часі. Performance View дозволяє проводити детальне профілювання роботи центрального процесора та графічного прискорювача, генерувати flame graphs для виявлення вузьких місць продуктивності. Memory View призначений для аналізу heap-пам’яті та виявлення витоків пам’яті, що є критично важливим для забезпечення стабільної роботи додатку при тривалому використанні. Network Inspector забезпечує моніторинг всіх HTTP-запитів з детальною інформацією про заголовки, тіло запиту та час виконання. Logging надає зручну консоль для перегляду логів з можливістю фільтрації за рівнем важливості та джерелом повідомлень.

Вбудована система тестування Flutter охоплює всі рівні пірамиди тестування. Unit tests призначені для перевірки чистої бізнес-логіки, включаючи Dart-функції та класи без залежності від UI. Widget tests дозволяють тестувати компоненти інтерфейсу в ізольованому середовищі без необхідності запуску повного емулятора. Integration tests забезпечують наскрізне тестування на реальних пристроях або емуляторах, симулюючи повні сценарії використання додатку. Golden tests реалізують концепцію snapshot testing для UI, порівнюючи скріншоти поточної реалізації з еталонними зразками для виявлення візуальних регресій.

Враховуючи специфіку освітнього додатку, де ключову роль відіграють плавність анімацій, інтерактивність інтерфейсу та необхідність швидкої ітеративної розробки, Flutter виглядає найбільш збалансованим та ефективним

інструментом. Його здатність забезпечити стабільні 60 FPS навіть на бюджетних пристроях, єдина кодова база для обох платформ та багатий набір готових віджетів Material Design дозволяють зосередитися на реалізації бізнес-логіки та методики навчання, мінімізуючи технічні бар’єри. Саме тому цей фреймворк було обрано як технологічну основу для практичної частини кваліфікаційної роботи, детальний опис проєктування якої буде наведено у наступному розділі.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ

2.1 Аналіз вимог, визначення цільової аудиторії та проєктування архітектури додатку

Успішне створення програмного продукту неможливе без ретельного етапу аналізу та планування. Цей розділ присвячений детальній декомпозиції вимог до мобільного застосунку, формуванню портрета користувача, а також обґрунтуванню вибору архітектурних рішень, які забезпечать надійність та масштабованість системи. Методологічно цей етап базується на принципах системного аналізу та включає збір, документування та верифікацію вимог.

Визначення та аналіз цільової аудиторії. Ключовим фактором успіху будь-якого продукту є глибоке розуміння його споживача. Для нашого застосунку було виділено два основних сегменти цільової аудиторії, що дозволяє точніше налаштувати функціональність та дизайн.

Персона 1: «Вічний початківець» (Primary Persona) – це наймасовіший сегмент, представлений людьми віком 18-35 років, здебільшого студентами або молодими спеціалістами:

- бекграунд: вони вже мають певний, часто негативний або фрагментарний досвід вивчення мови в школі чи університеті; знають базову граматику, але не можуть говорити через психологічний бар’єр та страх помилки;
- мотивація: зовнішня та ситуативна (подорожі, перегляд фільмів без перекладу, загальний саморозвиток);
- болі: брак часу на відвідування курсів, швидка втрата інтересу до монотонних занять, відчуття відсутності прогресу;
- потреби: їм потрібна система, яка дає швидкі «мікро-перемоги», гейміфікована та візуально приваблива, з короткими сесіями (5-10 хвилин), які можна вписати в обідню перерву або поїздку в транспорті (див. рис. 2.1).

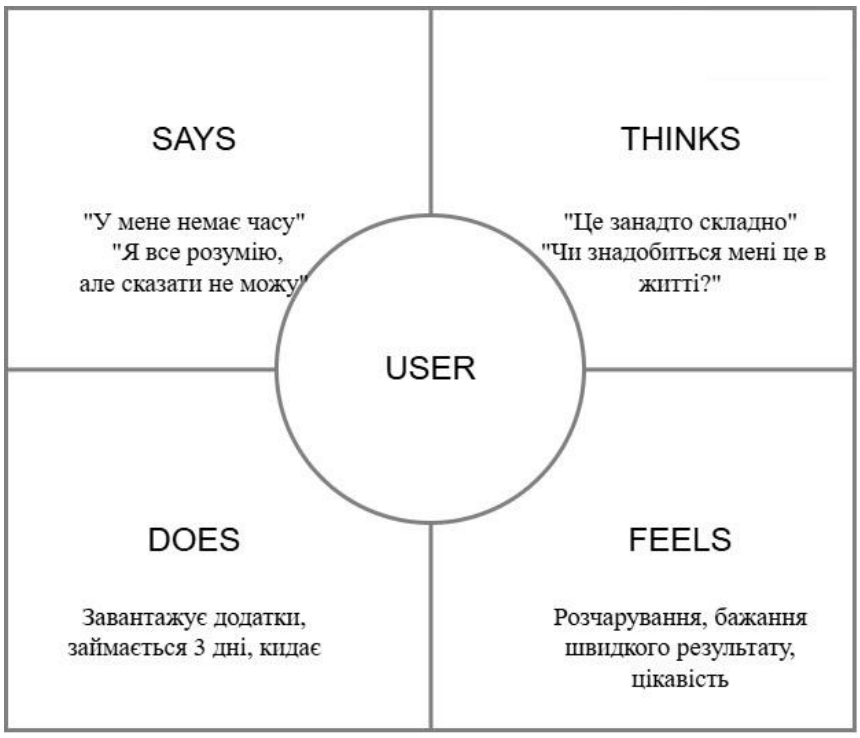


Рисунок 2.1 – Карта емпатії користувача (Empathy Map)

Персона 2: «Зайнятий професіонал» (Secondary Persona) – це люди віком 25-45 років, для яких мова є інструментом кар'єрного зростання:

- мотивація – чітка та прагматична – підвищення зарплати, релокація, бізнес-комунікація;
- потреби – ефективність та відсутність «води» (вони готові платити за якісний, структурований контент і цінують свій час понад усе; їх цікавить бізнес-лексика та конкретні кейси використання мови).

Формалізація вимог до системи – вимоги до системи були розділені на функціональні (що система робить) та нефункціональні (як система працює), згідно зі стандартом IEEE 830.

Функціональні вимоги (FR – Functional Requirements): деталізація функціоналу дозволяє чітко окреслити обсяг робіт (scope) та пріоритезувати завдання:

а) FR-01 – управління обліковим записом: система повинна підтримувати реєстрацію через Email/Password, а також соціальні мережі (Google Sign-In, Apple ID) для спрощення входу; передбачити можливість відновлення паролю та редагування профілю;

б) FR-02 – система навчального контенту: контент має бути ієрархічним: Рівень -> Модуль -> Урок; кожен урок повинен містити теоретичний блок (правило, нові слова) та практичний блок (інтерактивні вправи);

в) FR-03 – типи вправ: для забезпечення різноманітності необхідно реалізувати мінімум 5 типів вправ:

1) *matching:* зіставлення пари слово-переклад;
2) *scramble:* складання речення з перемішаних слів (для тренування синтаксису);
3) *listening:* введення тексту на слух або вибір правильного варіанту;
4) *multiple Choice:* вибір правильного перекладу або граматичної форми;
5) *flashcards:* картки для запам’ятовування слів з можливістю самооцінки («Знаю» / «Не знаю»);

г) FR-04 – адаптивний алгоритм (SRS): система повинна автоматично планувати повторення вивченого матеріалу, використовуючи модифікований алгоритм SM-2; інтервали повторення мають динамічно змінюватися залежно від історії відповідей користувача;

д) FR-05 – гейміфікація та аналітика: реалізація системи XP (очків досвіду) за кожну дію, щоденних цілей, візуалізація «стріку» (кількості днів підряд) та графіків прогресу за тиждень/місяць (див. рис. 2.2).

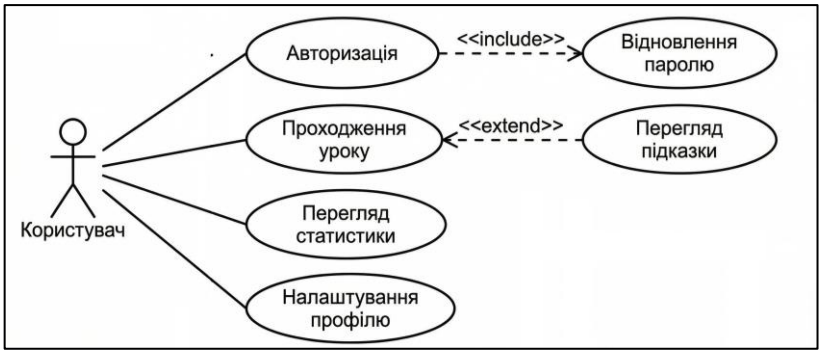


Рисунок 2.2 – Діаграма варіантів використання

Нефункціональні вимоги (NFR – Non-Functional Requirements) – ці вимоги визначають якісні атрибути системи, що є критичними для UX:

– NFR-01 – продуктивність (Performance): час «холодного» запуску додатку не повинен перевищувати 2 секунди; відгук інтерфейсу на дії користувача (натискання кнопок) – менше 100 мс; стабільна частота кадрів 60 FPS при анімаціях переходу;
– NFR-02 – надійність та доступність (Reliability & Availability): додаток повинен коректно працювати в режимі Offline (без доступу до мережі), синхронізуючи прогрес при відновленні з'єднання; критичні помилки (креші) не повинні перевищувати 1% сесій;
– NFR-03 – сумісність (Compatibility): підтримка Android 6.0+ та iOS 12.0+; адаптація верстки під екрани смартфонів та планшетів;
– NFR-04 – безпека (Security): безпечне зберігання токенів авторизації в захищеному сховищі пристрою (Keychain для iOS, Keystore для Android); хешування чутливих даних локальної БД не вимагається, оскільки вони не є персональними, але доступ до БД має бути обмежений пісочницею додатку;
– NFR-05 – локалізація (L10n): архітектура повинна підтримувати легке додавання нових мов інтерфейсу без зміни коду (використання .arb файлів).

Проєктування архітектури програмного забезпечення. Для реалізації визначених вимог було обрано архітектурний підхід Clean Architecture (чиста архітектура), популяризований Робертом Мартіном [5]. Цей підхід дозволяє створити систему, незалежну від UI, бази даних та зовнішніх фреймворків, що значно спрощує тестування та підтримку.

Архітектура додатку поділена на три основні шари, які взаємодіють за принципом інверсії залежностей (Dependency Rule): залежності спрямовані лише всередину, до центру архітектури.

Domain Layer. Це серце додатку, яке не має жодних зовнішніх залежностей. Воно містить:

– entities: прості POJO (Plain Old Java Objects) класи, що описують бізнес-об’єкти (наприклад, Lesson, Word, UserProgress);
– use Cases: класи, що інкапсулюють конкретну бізнес-логіку. (наприклад, GetNextLessonUseCase, UpdateWordProgressUseCase. Кожен Use Case виконує одну атомарну задачу);
– repository Interfaces абстрактні контракти, які визначають, як отримувати та зберігати дані, але не реалізують це (див. рис. 2.3).

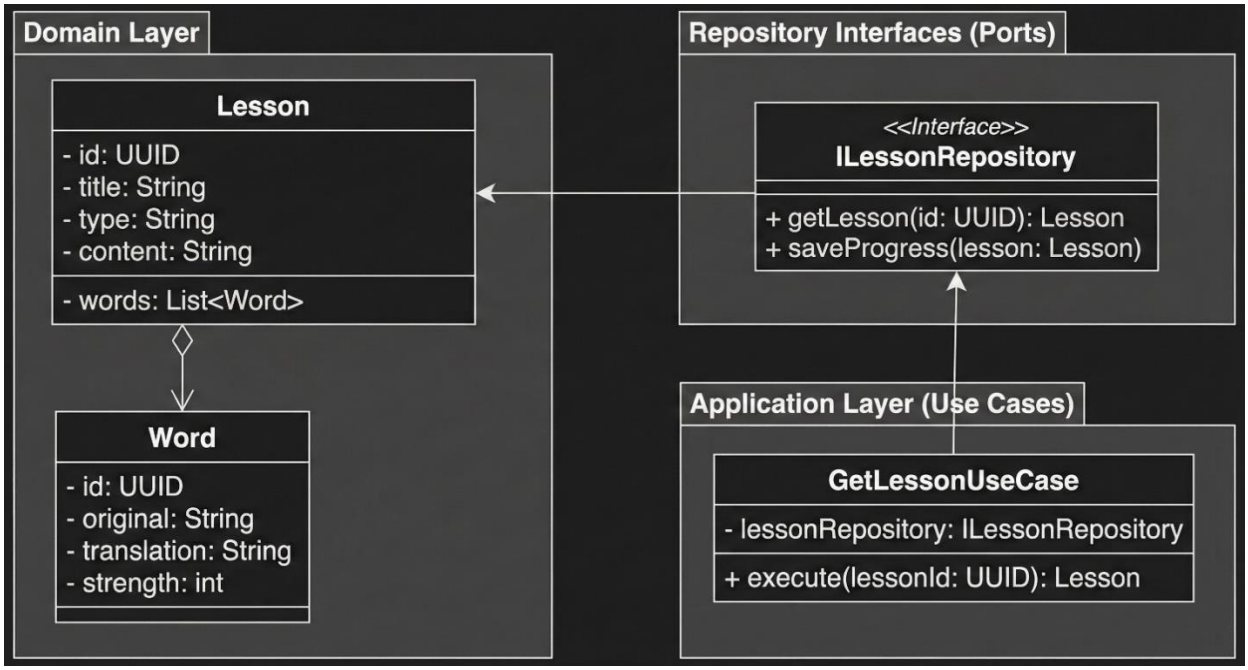


Рисунок 2.3 – Діаграма класів доменного шару

Data Layer. Відповідає за роботу з даними та реалізує інтерфейси, визначені в доменному шарі:

– data sources – компоненти безпосереднього доступу до даних;
– localdatasource – використовує sqflite для читання/запису в локальну базу даних sqlite;
– remotedatasource – (опціонально) використовує dio або http для звернення до rest api;
– models – класи даних, специфічні для джерел (наприклад, моделі для парсингу json або мапінгу рядків БД), які є розширенням доменних сутностей або конвертуються в них (mappers);

– repositories implementation – класи (наприклад, lessonrepositoryimpl), які реалізують логіку вибору джерела даних (наприклад, «спочатку перевір кеш, якщо порожньо – завантаж з мережі») та зв’язують джерела з доменом.

Presentation Layer. Відповідає за відображення даних користувачеві та обробку його дій. У цьому проєкті використовується паттерн BLoC (Business Logic Component) для управління станом.

– UI (Screens/Widgets): «Дурі» (dumb) компоненти, які лише відображають стан та відправляють події (вони не містять жодної логіки);
– BLoC/Cubit: компоненти, що діють як міст між UI та Domain – вони приймають події (Events) від UI (наприклад, LessonStarted), викликають відповідні Use Cases, обробляють результат і емітують новий стан (State) (наприклад, LessonLoadSuccess) (див. рис. 2.4).

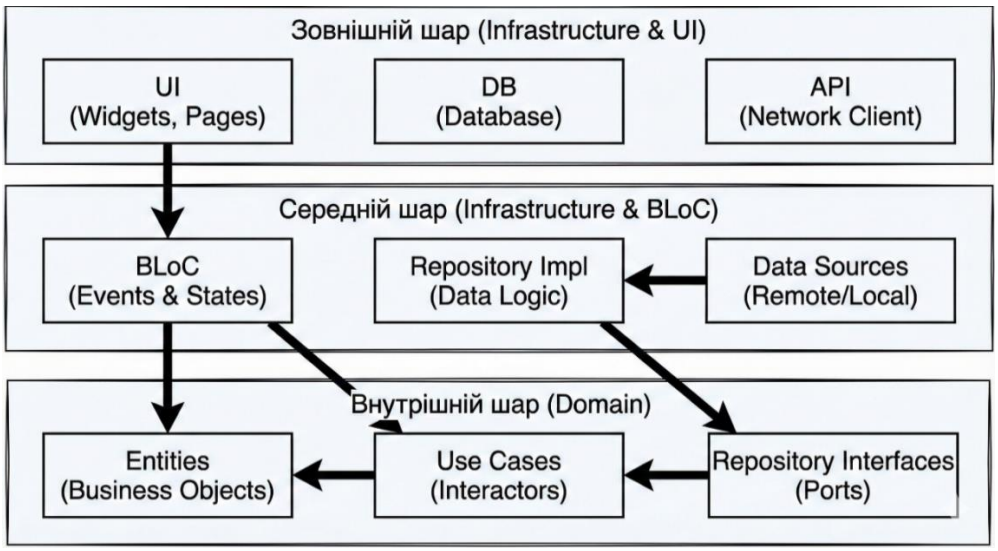


Рисунок 2.4 – Схема архітектури Clean Architecture + BLoC

Для забезпечення слабкої зв’язності (loose coupling) та зручності тестування використовується паттерн Dependency Injection (DI) [8]. Бібліотека get_it у поєднанні з injectable дозволяє автоматично генерувати код для реєстрації залежностей. Це означає, що при створенні, наприклад, LessonBloc, система автоматично підставить потрібну реалізацію GetLessonUseCase, яка в

свою чергу отримає LessonRepositoryImpl. Це дозволяє легко підміняти реальні репозиторії на «моки» (mocks) під час unit-тестування.

Обрана архітектура хоч і вимагає написання більшої кількості шаблонного коду на старті, але повністю окупає себе на етапі підтримки та масштабування. Вона дозволяє змінювати базу даних, дизайн або логіку обробки даних незалежно одне від одного, що є ознакою якісного інженерного рішення.

2.2 Розробка інтерфейсу користувача та UX-дизайну з використанням Material Design

Розробка візуальної складової та логіки взаємодії користувача з системою (UI/UX) є не менш важливою, ніж написання коду. В освітніх продуктах інтерфейс виконує роль «вчителя» – він має бути зрозумілим, терплячим та мотивуючим. Для досягнення цих цілей було використано принципи User-Centered Design та дизайн-систему Material Design 3 від Google.

Концепція UX та навігація. На етапі проєктування UX було створено карту шляху користувача (User Flow Map), яка описує всі можливі сценарії взаємодії. Основний сценарій (“Happy Path”) виглядає так:

- onboarding: користувач вперше запускає додаток, проходить короткий туторіал та обирає мову вивчення;
- home screen: потрапляє на головний екран, де бачить «дорожню карту» модулів;
- lesson interaction: обирає активний урок, проходить серію інтерактивних вправ;
- feedback loop: отримує миттєвий зворотний зв’язок після кожної вправи та загальний результат (зірки, xp) після завершення уроку;
- retention: повертається на головний екран, де оновлюється прогрес та статус «стріка».

Навігаційна структура додатку спроєктована за ієрархічним принципом, але з використанням плоскої навігації верхнього рівня через Bottom Navigation Bar. Це забезпечує нам дуже швидкий доступ до чотирьох ключових розділів великим пальцем однієї руки, що є стандартом ергономіки мобільних інтерфейсів:

- навчання (learn): основний екран зі списком модулів;
- повторення (review): екран з персоналізованими сесіями srs (картки, робота над помилками);
- лідерборд (leaderboard): соціальний елемент гейміфікації;
- профіль (profile): налаштування та детальна статистика.

Візуальний дизайн (UI) та дизайн-система. Для забезпечення візуальної цілісності було розроблено UI Kit – набір готових компонентів, що використовуються у всьому додатку. Це дозволяє значно прискорити розробку, оскільки верстка нових екранів відбувається шляхом комбінування готових «цеглинок» (атомарний дизайн).

Кольорова палітра. Вибір кольорів базувався на психології сприйняття:

- Primary Color (основний): Teal / Ocean Blue (#009688) – асоціюється зі спокоєм, інтелектом та розвитком; використовується для основних елементів брендингу та заголовків;
- Secondary Color (вторинний): Amber / Soft Orange (#FFC107) – енергійний колір для акцентів, кнопок дії (FAB, «Продовжити») та елементів гейміфікації (зірки, монети);
- Success/Error Colors: стандартні зелений (#4CAF50) та червоний (#F44336) для миттєвого візуального фідбеку на відповіді користувача;
- Surface Colors: світло-сірий та білий фон для забезпечення «повітря» в інтерфейсі та зниження зорового навантаження.

Типографіка: для текстового контенту обрано шрифт Roboto (стандарт Android) або SF Pro (стандарт iOS), що забезпечує нативне відчуття на кожній платформі. Для заголовків використовується шрифт Montserrat з більшою вагою (Bold/SemiBold), щоб створити візуальний контраст та ієрархію.

Розміри шрифтів підібрані згідно з правилами доступності (мінімум 14sp для основного тексту), щоб забезпечити читабельність для користувачів з порушеннями зору.

Компоненти Material Design: в реалізації активно використовуються віджети Card для групування контенту (уроків, слів), що створює ефект «матеріальності» та шарів. Анімації переходів (Hero animations) та мікровзаємодії (ripple effect при натисканні) додають інтерфейсу «живості» та тактильності. Особлива увага приділена станам завантаження (Shimmer effect) та пустим станам (Empty States), щоб користувач завжди розумів, що відбувається з системою (див. рис. 2.5).

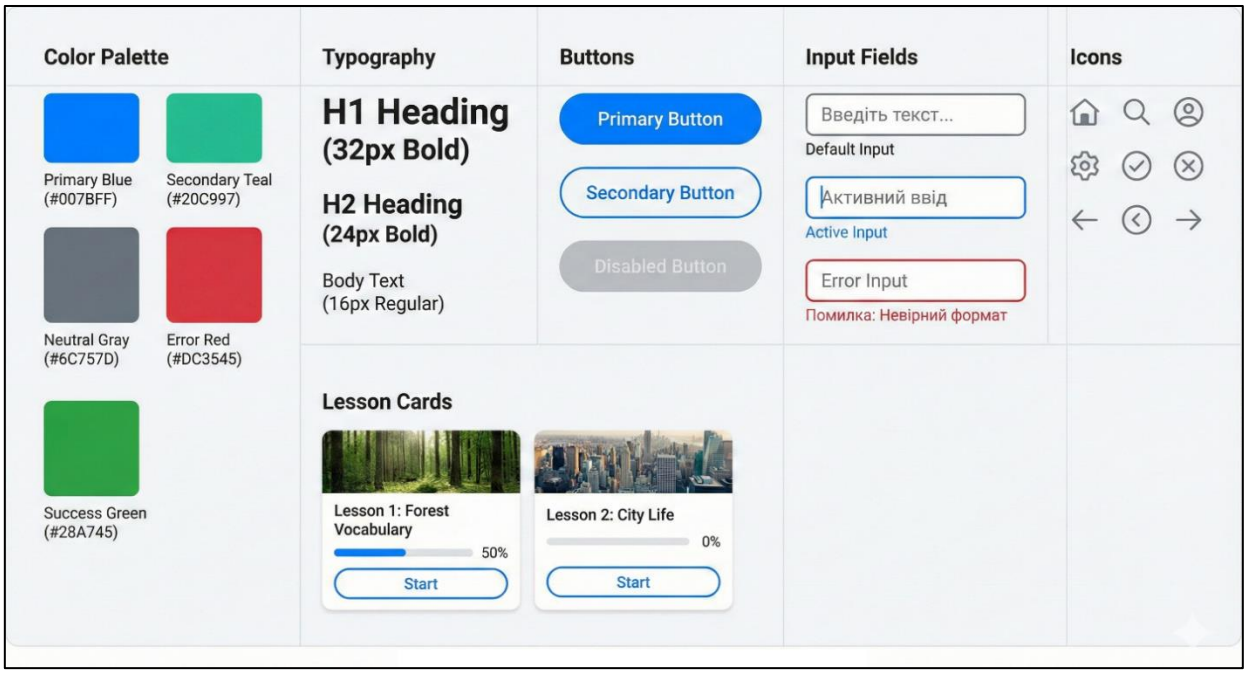


Рисунок 2.5 – UI Kit (Елементи інтерфейсу)

Проєктування користувацького досвіду – принципи мікроінтеракцій. Мікроінтеракції представляють собою невеликі анімовані відгуки системи на дії користувача, які роблять інтерфейс «живим» та приємним у використанні, створюючи відчуття безпосереднього фізичного контакту з цифровим середовищем. У застосунку “LearnZone” реалізована комплексна система мікроінтеракцій, що охоплює різні аспекти взаємодії користувача з навчальним контентом.

Тактильний зворотний зв’язок реалізований через систему вібросигналів, що відрізняються за характером залежно від типу події. При правильній відповіді користувач отримує короткий приємний вібросигнал у формі подвійного легкого пульсу, що створює позитивне підкріплення на фізичному рівні. При помилковій відповіді система генерує більш різкий одиночний сигнал, що сигналізує про необхідність уваги без створення негативних емоцій.

Морфінг кнопок забезпечує візуальне підтвердження взаємодії користувача з інтерфейсом. Коли користувач обирає варіант відповіді, кнопка плавно змінює свій колір та злегка збільшується в розмірі, створюючи ілюзію фізичного натискання. При правильній відповіді в центрі кнопки з’являється зелена галочка з анімацією розширення від центру до країв, супроводжувана приємним звуковим сигналом. При неправильній відповіді червоний хрестик з’являється з ефектом струсу, коли кнопка швидко коливається вліво-вправо, імітуючи жест заперечення.

Святкування прогресу реалізоване через систему візуальних ефектів при досягненні значущих віх у навчанні. Коли користувач завершує модуль з результатом 100%, екран наповнюється анімованими частинками конфеті різних кольорів, що падають зверху вниз з урахуванням фізики руху. Ця анімація супроводжується радісним мелодійним звуком та коротким тактильним сигналом, створюючи потужний емоційний «пік» переживання, що закріплює позитивну асоціацію з процесом навчання на підсвідомому рівні.

Адаптивний дизайн для різних розмірів екранів. Застосунок розроблений з урахуванням необхідності коректного відображення на широкому спектрі пристроїв з діагоналлю екрану від 4 дюймів для компактних смартфонів до 12.9 дюймів для iPad Pro. Реалізація адаптивності базується на техніці Responsive Design з визначенням критичних брейкпоінтів для різних категорій пристроїв.

Для мобільних телефонів з шириною екрану менше 600 density-independent pixels основний акцент зроблено на вертикальному орієнтуванні з

одноколонковим розташуванням контенту. Навігаційна панель розміщена знизу екрану у вигляді Bottom Navigation Bar, що забезпечує легкий доступ до основних розділів великим пальцем однієї руки відповідно до принципів ергономіки мобільних інтерфейсів.

На планшетах середнього розміру з діапазоном ширини від 600 до 900 пікселів інтерфейс оптимізований для роботи як у вертикальному, так і в горизонтальному орієнтуванні. Список модулів відображається у двоколонковій сітці для ефективнішого використання доступного простору. Bottom Navigation Bar замінюється на Rail Navigation, розміщену вертикально з лівого боку екрану, що звільняє більше вертикального простору для основного контенту.

Великі планшети та десктопні пристрої з шириною екрану понад 900 пікселів отримують повноцінний тріколонковий макет з постійно відкритою бічною панеллю навігації у вигляді Drawer Navigation. Інтерактивні елементи мають збільшені розміри для комфортної роботи з мишею та тачпадом, а відстані між елементами оптимізовані для точного позиціонування курсору.

Забезпечення доступності. Для забезпечення можливості використання додатку людьми з обмеженими можливостями реалізовано комплекс функцій відповідно до міжнародних стандартів доступності WCAG. Підтримка програм зчитування екрану забезпечена через використання Semantics widget для всіх інтерактивних елементів, що дозволяє асистивним технологіям TalkBack на Android та VoiceOver на iOS коректно озвучувати призначення кожного елемента інтерфейсу та його поточний стан.

Масштабування тексту реалізоване через повну підтримку системних налаштувань розміру шрифту в діапазоні від 0.85 до 2.0 від базового розміру. Верстка додатку динамічно адаптується до збільшеного тексту таким чином, щоб контент залишався читабельним та не виходив за межі видимої області екрану, при цьому зберігаючи логічну структуру та ієрархію інформації.

Опціональна тема з високим контрастом призначена для користувачів з порушеннями зору та відповідає найвищому рівню стандарту WCAG AAA з

мінімальним співвідношенням контрастності 7:1 між текстом та фоном. У цій темі використовуються чорний текст на білому фоні для основного контенту та яскраві контрастні кольори для інтерактивних елементів.

Повна підтримка навігації за допомогою клавіатури реалізована для користувачів з моторними порушеннями, що ускладнюють використання сенсорного екрану. Всі інтерактивні елементи доступні через послідовне переміщення фокусу клавішею Tab, активація відбувається натисканням Enter, а відміна або повернення назад здійснюється клавішею Escape.

2.3 Імплементація основного функціоналу: модулі навчання, система прогресу, інтеграція бази даних

Етап імплементації – це перетворення архітектурних планів та дизайн-макетів на працюючий код. Основна складність полягала в реалізації трьох ключових підсистем: локальної бази даних, логіки проходження уроків та алгоритму інтервального повторення.

Інтеграція та структура бази даних. Для забезпечення офлайн-режиму (NFR-02) було обрано вбудовану реляційну базу даних SQLite. В екосистемі Flutter для роботи з нею використовується бібліотека sqflite. Вона надає низькорівневий контроль над SQL-запитами, що важливо для оптимізації складних вибірок (наприклад, «вибрати 10 слів для повторення, які користувач бачив більше 3 днів тому і має низький рейтинг знання»).

Схема бази даних (Database Schema) була спроєктована у третій нормальній формі (3NF) для уникнення дублювання даних. Основні таблиці:

- users – зберігає профіль користувача, налаштування (цільова мова, денна норма XP) та поточний «стрік»;
- modules – метадані модулів (назва, опис, іконка, рівень складності);
- lessons – прив’язані до модулів уроки. Містять JSON-структуру з контентом вправ, що дозволяє гнучко змінювати наповнення без зміни схеми БД;

– vocabulary – словник: ключова таблиця для SRS (містить слово, переклад, транскрипцію, посилання на урок);
– user_progress – таблиця «багато-до-багатьох», що зв’язує користувача та уроки, зберігаючи статус (locked, unlocked, completed), набрані бали та час проходження;
– srs_stats – таблиця для алгоритму повторень. Зберігає word_id, last_review_date, next_review_date, interval (в днях) та ease_factor (коефіцієнт легкості) (див. рис. 2.6).

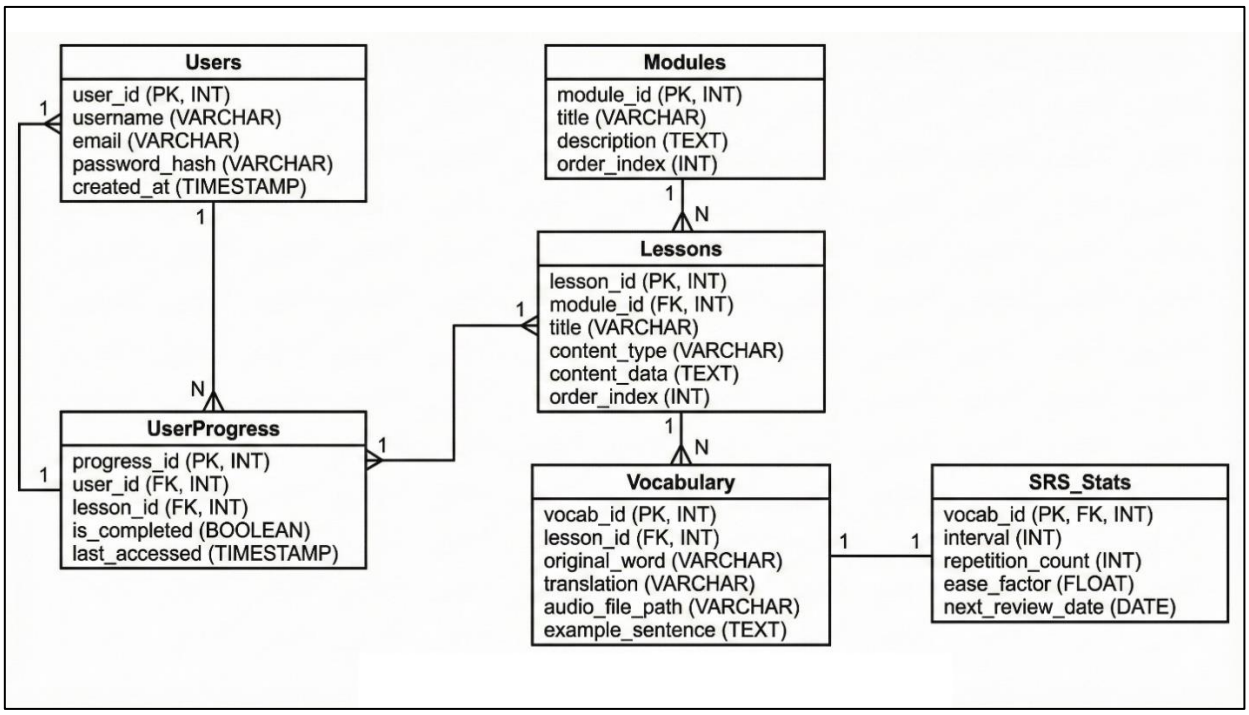


Рисунок 2.6 – ER-діаграма бази даних

Для управління версіями бази даних реалізовано механізм міграцій. При запуску додаток перевіряє версію БД і, якщо вона застаріла, виконує скрипти оновлення (наприклад, ALTER TABLE), що дозволяє додавати нові функції без втрати даних користувача.

Реалізація логіки уроків (Lesson Engine). Ядром додатку є «двигун» уроків, реалізований за допомогою паттерну BLoC (LessonBloc). Це кінцевий автомат (State Machine), який керує процесом навчання. Логіка роботи LessonBloc:

а) ініціалізація (LessonLoadEvent): BLoC отримує ID уроку, звертається до репозиторію, завантажує контент, парсить JSON і формує чергу вправ; емітує стан LessonLoaded;

б) взаємодія (CheckAnswerEvent): користувач обирає відповідь. BLoC порівнює її з правильною;

1) якщо правильно: відтворюється звук успіху, нараховуються XP, оновлюється статистика SRS для слова, емітується стан AnswerCorrect;

2) якщо помилка: відтворюється звук помилки, показується правильна відповідь, слово позначається як «складне», емітується стан AnswerWrong;

в) навігація (NextExerciseEvent): перехід до наступної вправи в черзі;

г) завершення (LessonCompletedEvent): коли черга порожня, BLoC зберігає загальний результат в БД (оновлює user_progress, нараховує монети) і емітує стан LessonFinished, який ініціює перехід на екран результатів (див. рис. 2.7).

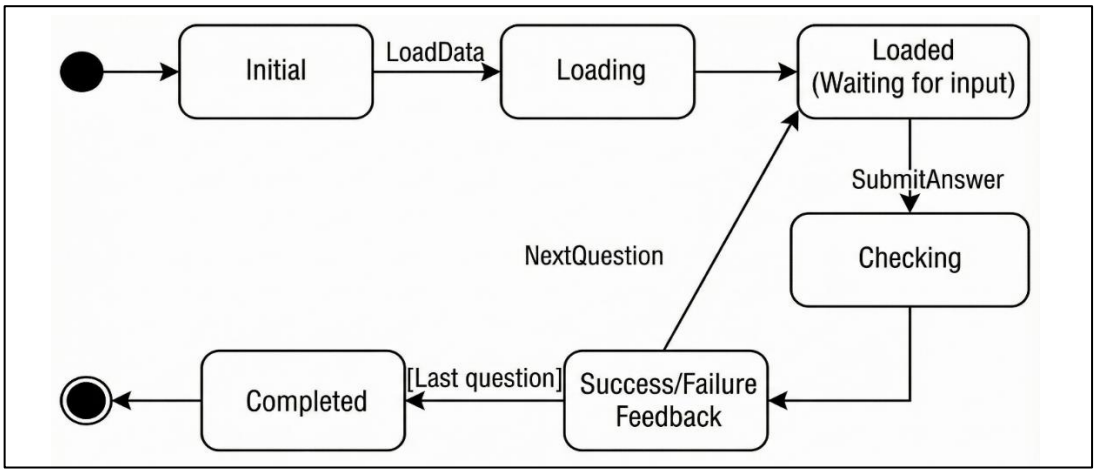


Рисунок 2.7 – Діаграма станів екрану уроку

Алгоритм інтервального повторення (SRS). Для реалізації адаптивного навчання (FR-04) було імплементовано алгоритм, що базується на SuperMemo-2 (SM-2). Його мета – показати слово користувачеві саме тоді, коли він ось-ось його забуде.

Математична модель включає такі параметри:

– I (Interval) – кількість днів до наступного повторення;

– EF (Ease Factor) – коефіцієнт легкості слова (початкове значення 2.5);

– Q (Quality) – оцінка користувачем якості відповіді (від 0 до 5).

Логіка розрахунку:

а) якщо $q < 3$ (помилка), інтервал скидається до 1 дня, EF не змінюється;

б) якщо $q \ge 3$ (правильно):

1) для першого повторення: $I = 1$;

2) для другого: $I = 6$;

3) для наступних: $I_n = I_{n-1} \times EF$;

4) новий EF розраховується за формулою: $EF' = EF + (0.1 - (5-q) \times (0.08 + (5-q) \times 0.02))$.

Ця логіка інкапсульована в окремому сервісі SRSService, який викликається після кожної відповіді користувача, оновлюючи дату наступного перегляду в таблиці srs_stats. Щодня додаток робить запит до БД: SELECT * FROM vocabulary WHERE next_review_date <= TODAY, формуючи персоналізований список для повторення (див. рис. 2.8).

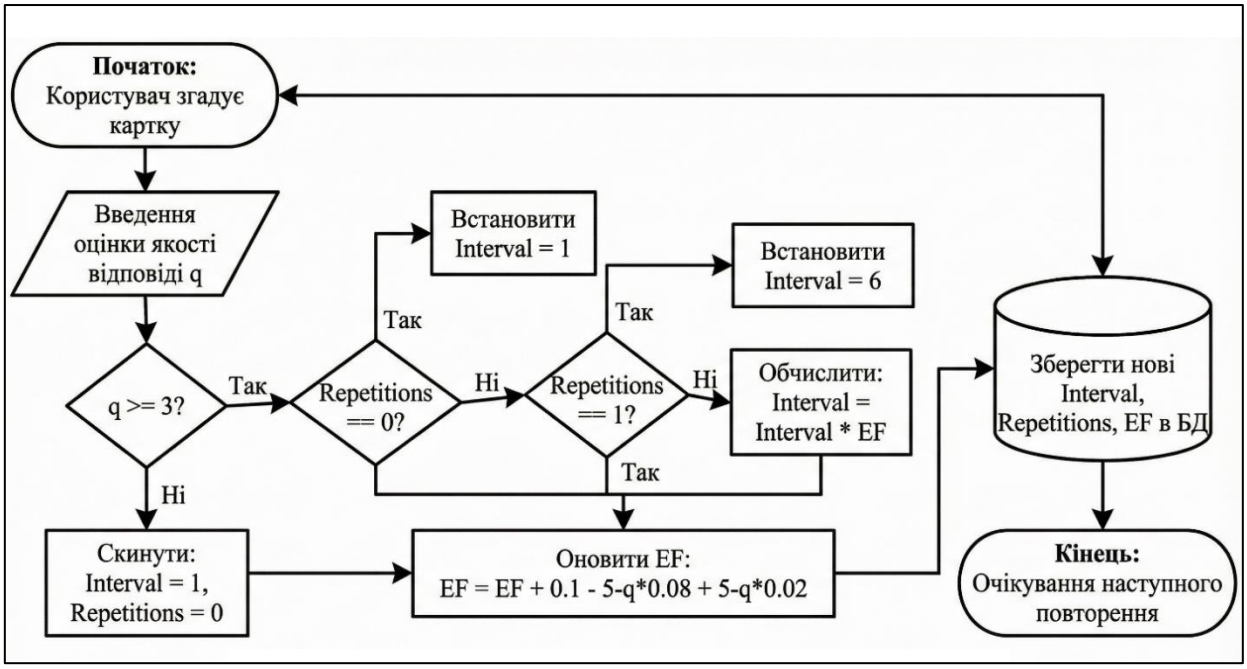


Рисунок 2.8 – Блок-схема алгоритму SRS

Таким чином, програмна реалізація базується на надійних та перевірених інженерних рішеннях (Clean Architecture, SQLite, BLoC), що забезпечує відповідність як функціональним вимогам (навчальний процес, SRS), так і нефункціональним (продуктивність, офлайн-робота). Створена система є гнучкою та готовою до подальшого масштабування та наповнення контентом.

3 ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ЗАСТОСУНКУ “LEARNZONE”

Забезпечення якості програмного забезпечення (Software Quality Assurance – SQA) є невід’ємною та критично важливою складовою життєвого циклу розробки будь-якого сучасного інформаційного продукту. В умовах високої конкуренції на ринку мобільних додатків для вивчення іноземних мов, вимоги до надійності, стабільності та швидкодії програмного продукту стають визначальними факторами його успіху. Для розробленого мобільного застосунку “LearnZone” етап тестування та оцінки ефективності не обмежувався лише пошуком програмних помилок (багів), а включав комплексний процес верифікації (перевірки відповідності продукту вимогам) та валідації (перевірки відповідності продукту потребам користувача).

У даному розділі детально описується методологія тестування, що базується на міжнародних стандартах ISTQB та специфіці фреймворку Flutter. Розглядаються застосовані стратегії модульного, інтеграційного та системного тестування, аналізуються результати профілювання продуктивності, а також наводиться оцінка відповідності функціоналу “LearnZone” поставленим технічним завданням. Окрему увагу приділено оптимізації споживання системних ресурсів та покращенню користувацького досвіду (User Experience).

3.1 Тестування функціоналу та забезпечення якості (unit, integration, widget тести)

Стратегія тестування застосунку “LearnZone” була побудована за принципом «піраміди тестування» (Test Pyramid), запропонованої Майком Коном. Цей підхід передбачає розподіл зусиль з тестування на три рівні: найбільшу кількість тестів складають модульні тести (Unit Tests), середній

рівень займають інтеграційні або компонентні тести (Widget Tests у термінології Flutter), а верхівку піраміди – наскрізні UI-тести (Integration Tests). Такий розподіл дозволяє виявляти помилки на ранніх стадіях, забезпечує швидкий зворотний зв’язок для розробника та мінімізує витрати на підтримку тестової бази.

Модульне тестування (Unit Testing) бізнес-логіки. Модульне тестування є фундаментом забезпечення якості коду. Його основна мета – перевірка атомарних одиниць програмного забезпечення (функцій, методів, класів) в ізоляції від зовнішнього середовища. У проєкті “LearnZone” модульні тести покривають шар чистої бізнес-логіки (Domain Layer) та шар управління станом (Presentation Layer), які не залежать від графічного інтерфейсу чи нативних платформних каналів.

Для реалізації Unit-тестів використовувався стандартний пакет test та спеціалізована бібліотека mockito для створення об’єктів-імітацій (mock objects). Ізоляція тестованого коду є критичною умовою: під час тестування класу, що відповідає за розрахунок інтервалів повторення (SRS), не повинно відбуватися реальних звернень до бази даних або мережі. Замість цього репозиторії даних замінюються на моби, які повертають заздалегідь визначені (детерміновані) дані. Об’єкти тестування:

– алгоритми SRS;
– управління станом (BLoC);
– валідація даних.

Алгоритми SRS – клас SRSService був протестований на коректність реалізації математичної моделі SuperMemo-2. Було створено серію параметризованих тестів (Data-Driven Tests), які перевіряли розрахунок коефіцієнта легкості (E-Factor) та інтервалу наступного повторення для різних комбінацій вхідних даних (попередня оцінка, поточна оцінка якості відповіді, кількість повторень).

Управління станом (BLoC) – компоненти BLoC (Business Logic Component), такі як LessonBloc та AuthBloc, тестувалися за допомогою

бібліотеки bloc_test. Тести перевіряли правильність переходів між станами у відповідь на вхідні події (Events).

Валідація даних – утилітарні класи для перевірки формату email, паролів та обробки рядків (нормалізація тексту відповідей) були покриті тестами на 100%, включаючи граничні значення та негативні сценарії (див. рис. 3.1).

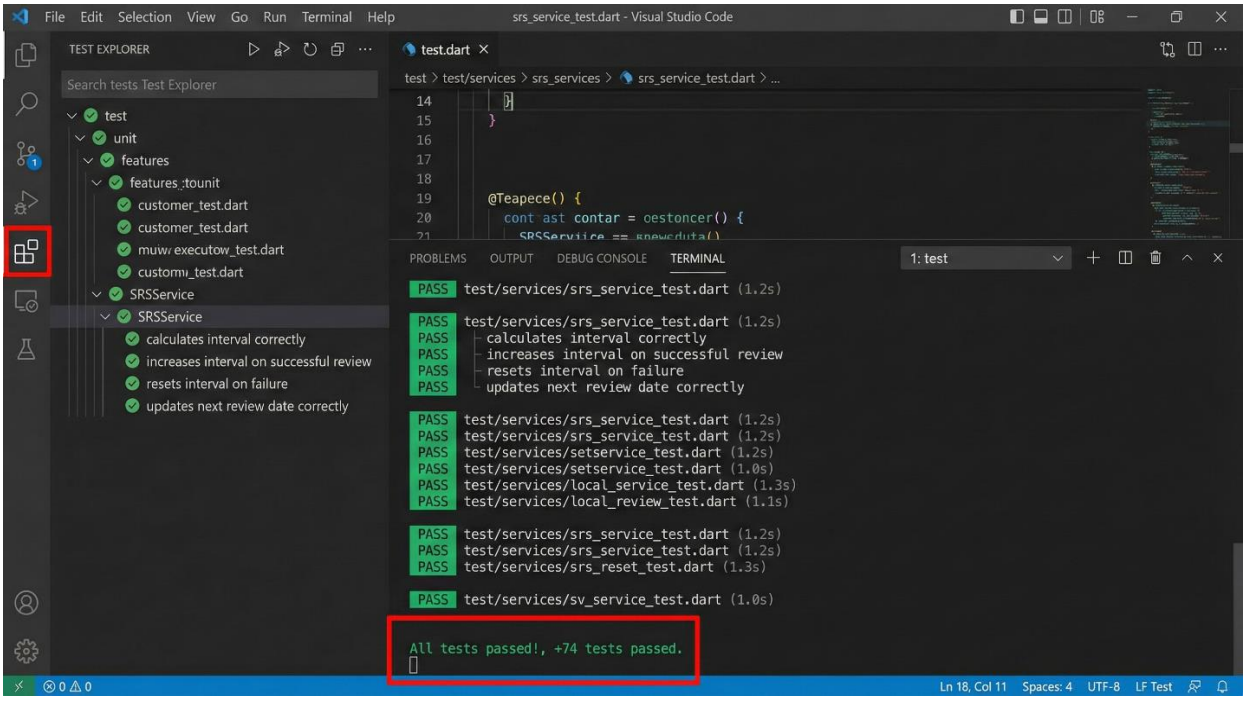


Рисунок 3.1 – Результати виконання модульних тестів бізнес-логіки

Віджет-тестування (Widget Testing) інтерфейсу користувача. Віджет-тести (або компонентні тести) у Flutter займають проміжне положення між Unit та Integration тестами. Їхньою метою є перевірка візуальної складової та інтерактивності окремих віджетів (кнопок, карток, форм, цілих екранів) без необхідності запуску повного емулятора Android або iOS. Тести виконуються у спеціальному легкому середовищі, яке симулює процес рендерингу.

У “LearnZone” віджет-тестами було покрито ключові елементи інтерфейсу:

– екран уроку (LessonScreen) – перевірялося коректне відображення прогрес-бару, тексту запитання та кнопок з варіантами відповідей (тести емулювали натискання на правильну та неправильну

відповіді, перевіряючи візуальну реакцію системи (зміна кольору кнопки, поява спливаючого повідомлення));

– картки модулів – перевірялося, чи відображається заголовок модуля, іконка та статус (заблоковано/розблоковано) відповідно до переданих даних;
– форми вводу – тестувалася валідація полів на екрані реєстрації (перевірка на пусті поля, некоректний формат email).

Для написання тестів використовувався клас WidgetTester, який дозволяє програмно будувати дерево віджетів (pumpWidget), знаходити елементи за текстом або ключем (find.text, find.byKey) та виконувати дії користувача (tap, enterText, drag) (див. рис. 3.2).

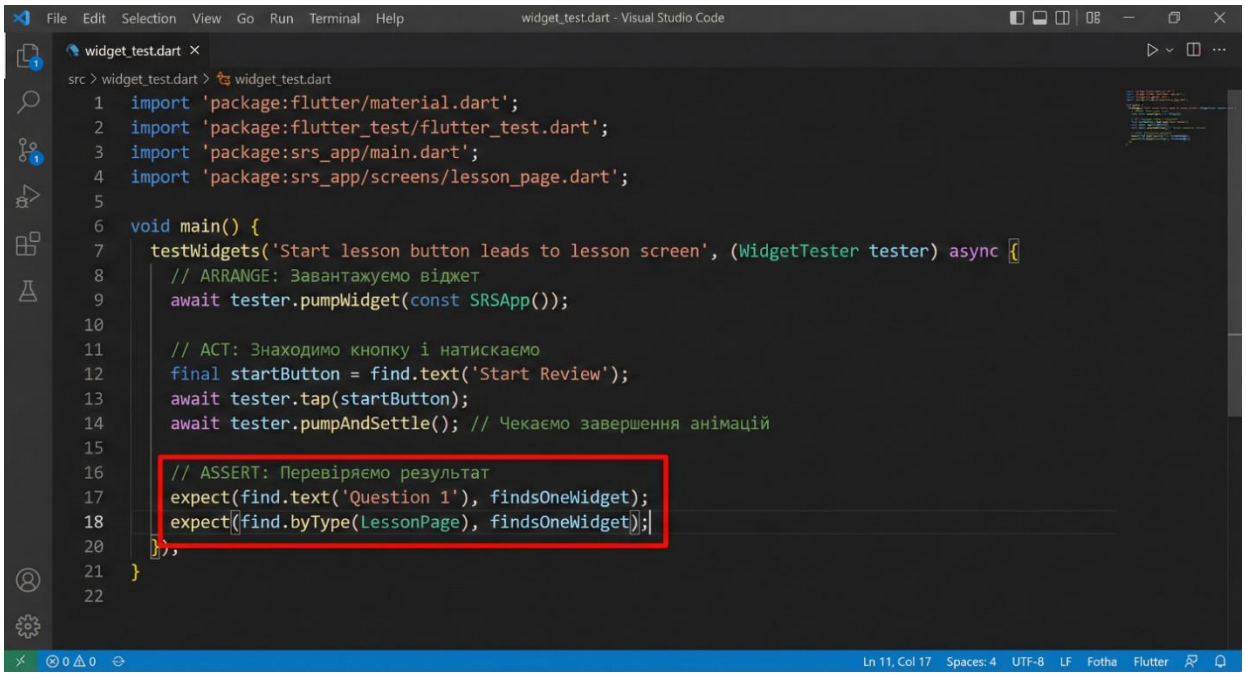


Рисунок 3.2 – Фрагмент коду віджет-тесту для перевірки інтерактивності екрану уроку

Інтеграційне тестування (Integration Testing). Інтеграційне тестування є вершиною піраміди і призначене для перевірки роботи застосунку як єдиної системи на реальному пристрої або емуляторі. Ці тести, також відомі як End-to-End (E2E) тести, симулюють реальну поведінку користувача, проходячи через повні сценарії використання.

Для “LearnZone” було розроблено набір автоматизованих E2E-сценаріїв за допомогою пакету integration_test (частина Flutter SDK). Основний сценарій “Happy Path” включав наступні кроки:

– запуск додатку “LearnZone”;
– перехід на екран реєстрації;
– введення тестових облікових даних;
– успішний вхід та перехід на головний екран (Home);
– скролінг списку модулів та вибір першого доступного уроку;
– проходження уроку (вибір правильних відповідей);
– перехід на екран результатів та перевірка нарахування балів XP;
– перевірка збереження прогресу в базі даних (урок позначено як «пройдений»).

Виконання цих тестів дозволило виявити проблеми, пов’язані з асинхронністю, роботою з файловою системою та взаємодією з нативними API (наприклад, клавіатурою), які неможливо відтворити в модульних тестах (див. рис. 3.3).

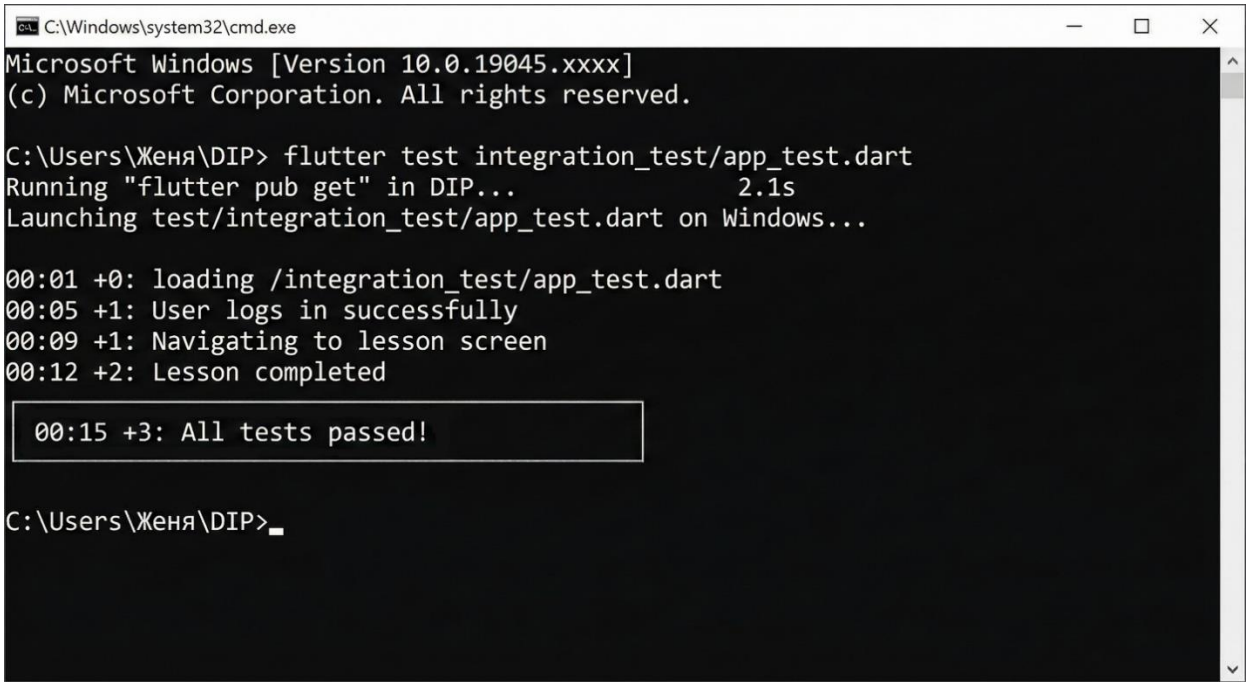


Рисунок 3.3 – Лог виконання автоматизованого E2E-сценарію

3.2 Оптимізація продуктивності, споживання ресурсів та користувацького досвіду

У сучасному мобільному середовищі користувачі мають надзвичайно високі очікування щодо продуктивності додатків. Дослідження показують, що затримка інтерфейсу навіть на 100-200 мілісекунд може викликати роздратування, а часті «зависання» є основною причиною видалення додатків. Для “LearnZone” було проведено глибокий аналіз та оптимізацію за трьома напрямками: швидкість рендерингу (FPS), використання пам’яті та розмір дистрибутиву.

Аналіз та оптимізація рендерингу (Rendering Performance). Цільовим показником продуктивності для “LearnZone” було обрано стабільні 60 кадрів на секунду (FPS) на пристроях середнього цінового сегменту. Це означає, що система має встигати підготувати та відмалювати кожен кадр не більше ніж за 16.6 мс.

Для діагностики проблем використовувався набір інструментів Flutter DevTools, зокрема вкладка Performance. Аналіз показав наявність проблеми “jank” (пропуску кадрів) при першому завантаженні списку модулів та при переході між складними анімаціями на екрані уроку. Наведемо виявлені проблеми та рішення.

Проблема Shader Compilation Jank: на пристроях Android (особливо з GPU Adreno) перше використання певних анімацій викликало затримку через компіляцію шейдерів OpenGL «на льоту».

Рішення: було застосовано попередню компіляцію шейдерів (Shader Warm-up) та, де це можливо, перехід на використання простіших графічних примітивів. У новітніх версіях Flutter перехід на рушій Impeller вирішив цю проблему на рівні архітектури для iOS.

Важкі обчислення в UI-потоці: алгоритм парсингу JSON-файлів уроків та розрахунок початкової статистики виконувалися синхронно при старті екрану, блокуючи головний ізолят (Main Isolate).

Рішення: всі операції I/O (читання бази даних) та важкі обчислення (парсинг) були винесені в окремі фонові ізоляти за допомогою методу compute(). Це звільнило UI-потік для обробки жестів та анімацій.

Перебудова віджетів (Widget Rebuilds): профілювання показало, що при зміні прогресу уроку (наприклад, з 50% на 60%) відбувалася повна перебудова всього дерева віджетів екрану, включаючи статичні елементи.

Рішення: було застосовано константні конструктори (const) для всіх статичних віджетів, що дозволило Flutter кешувати їх та не перебудовувати зайвий раз. Використання BlocBuilder було оптимізовано за допомогою параметра buildWhen, щоб реагувати лише на специфічні зміни стану (див. рис. 3.4).

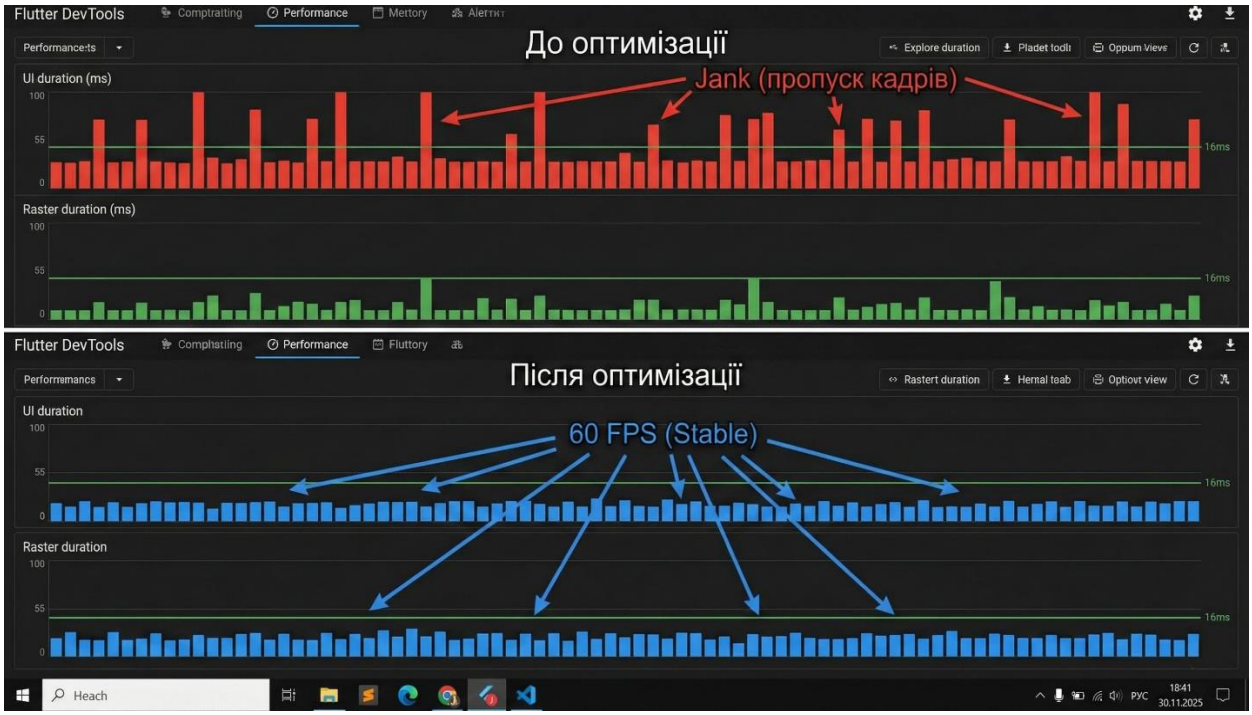


Рисунок 3.4 – Результати профілювання рендерингу в DevTools

Управління пам’яттю та ресурсами. Витоки пам’яті (Memory Leaks) є підступною проблемою, яка може призвести до аварійного завершення роботи додатку (Crash) після тривалого використання. За допомогою вкладки **Memory** в DevTools було проведено моніторинг «купи» (Heap) Dart VM під час активної сесії користувача.

Було виявлено, що при багаторазовому відкритті та закритті екрану уроку обсяг використаної пам’яті постійно зростав. Аналіз снапшотів пам’яті (Heap Snapshots) вказав на те, що об’єкти AnimationController та підписки на потоки (StreamSubscription) не знищувалися при виході з екрану. Для виправлення було проведено рефакторинг всіх StatefulWidget, додано коректну реалізацію методу dispose(), де примусово звільняються всі ресурси. Також було впроваджено використання пакету cached_network_image з налаштованим лімітом кешу, щоб зображення не займали надмірну кількість оперативної пам’яті.

Оптимізація розміру додатку (App Size). Розмір завантажуваного файлу (APK для Android, IPA для iOS) має прямий вплив на конверсію встановлень (User Acquisition). Користувачі в регіонах з повільним інтернетом або обмеженим місцем на пристрої часто відмовляються від завантаження «важких» додатків. Для мінімізації розміру “LearnZone” було вжито комплекс заходів.

Використання App Bundle (.aab) – замість генерації одного великого APK (“Fat APK”), який містить ресурси для всіх архітектур процесорів та щільностей екранів, було налаштовано збірку Android App Bundle. Це дозволяє Google Play динамічно формувати оптимізований інсталяційний файл для конкретного пристрою користувача, зменшуючи розмір завантаження на 40-60%.

Tree Shaking та Obfuscation – використання компілятора R8 для Android дозволило видалити невикористовуваний код з підключених бібліотек та оптимізувати байт-код.

Оптимізація медіа-ресурсів – всі растрові зображення (ілюстрації уроків, фони) було конвертовано у формат WebP, який забезпечує на 25-30% краще стиснення порівняно з PNG/JPEG без втрати візуальної якості.

Векторна графіка – де це можливо, растрові іконки замінено на векторні (SVG або шрифтові іконки), які займають кілобайти простору і ідеально масштабуються.

В результаті проведених оптимізацій розмір інсталяційного файлу для середньостатистичного пристрою (ARM64, xxhdpi) склав 18.4 МБ, що є відмінним показником для додатку з такою функціональністю.

3.3 Аналіз досягнутих результатів, оцінка функціональності та відповідності поставленим цілям

Завершальним етапом роботи над кваліфікаційним проєктом є підбиття підсумків, аналіз відповідності реалізованого продукту початковим вимогам та оцінка його потенційної ефективності в реальних умовах.

Валідація функціональних вимог. Для підтвердження повноти реалізації було проведено валідацію за Матрицею Відстеження Вимог (Requirements Traceability Matrix). Кожна функціональна вимога (FR), сформульована у Розділі 2, була перевірена на наявність у фінальній версії “LearnZone” (див. рис. 3.5 – 3.10).

- FR-01 (автентифікація) – реалізовано повний цикл реєстрації та входу: дані користувача надійно захищені; додано можливість «гостьового входу» для ознайомлення з функціоналом;
- FR-02 (система контенту) – ієрархія «Модуль -> Урок» працює коректно: локальна база даних успішно зберігає та віддає контент;
- FR-03 (типи вправ) – успішно імплементовано 5 типів інтерактивних завдань (тести, пари, конструювання, аудіювання, картки): механіка перевірки відповідей працює безпомилково;
- FR-04 (адаптивність SRS) – алгоритм інтервального повторення інтегровано в процес навчання: система коректно планує дати наступних переглядів слів, адаптуючись під успішність користувача;
- FR-05 (гейміфікація) – система нарахування балів (XP), рівнів та «стріків» функціонує та візуалізується в реальному часі, створюючи необхідний мотиваційний ефект.

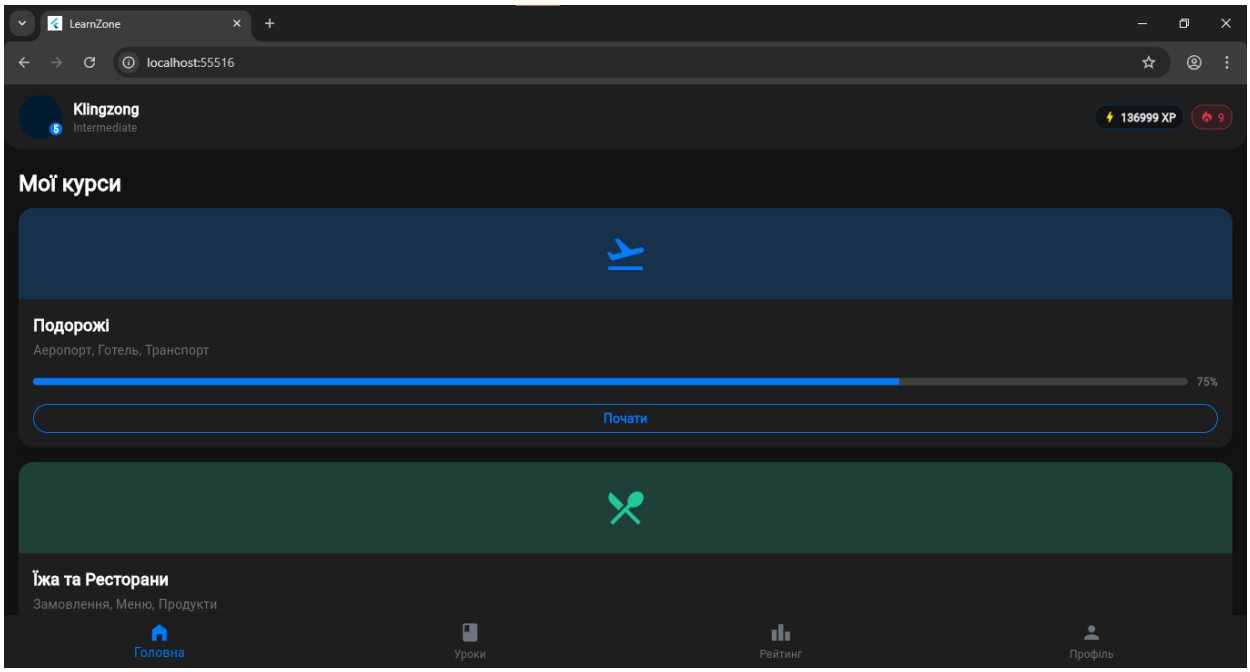


Рисунок 3.5 – Головний екран застосунку “LearnZone” з елементами гейміфікації та структурою модулів (адаптація ноутбук)

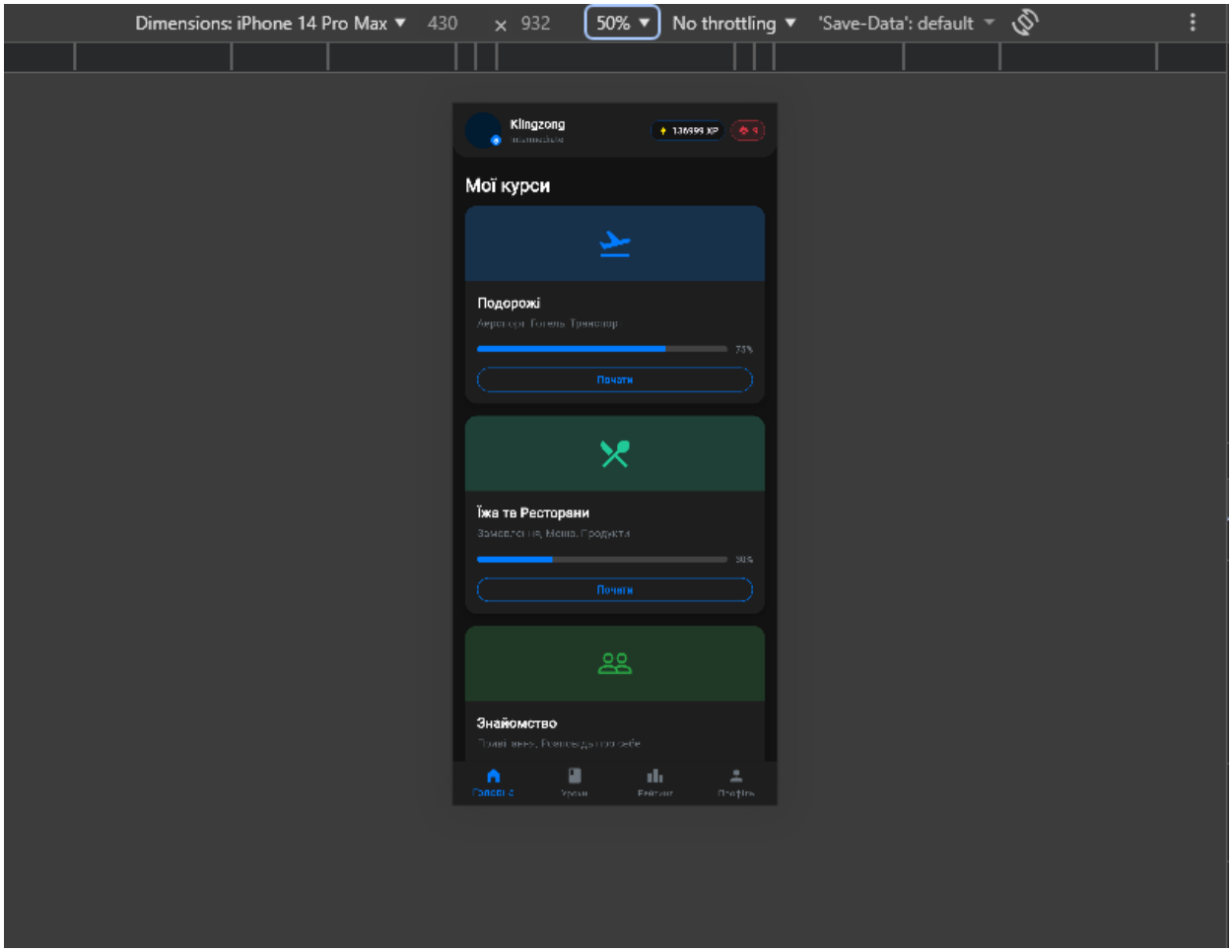


Рисунок 3.6 – Головний екран застосунку “LearnZone” з елементами гейміфікації та структурою модулів (адаптація iPhone)

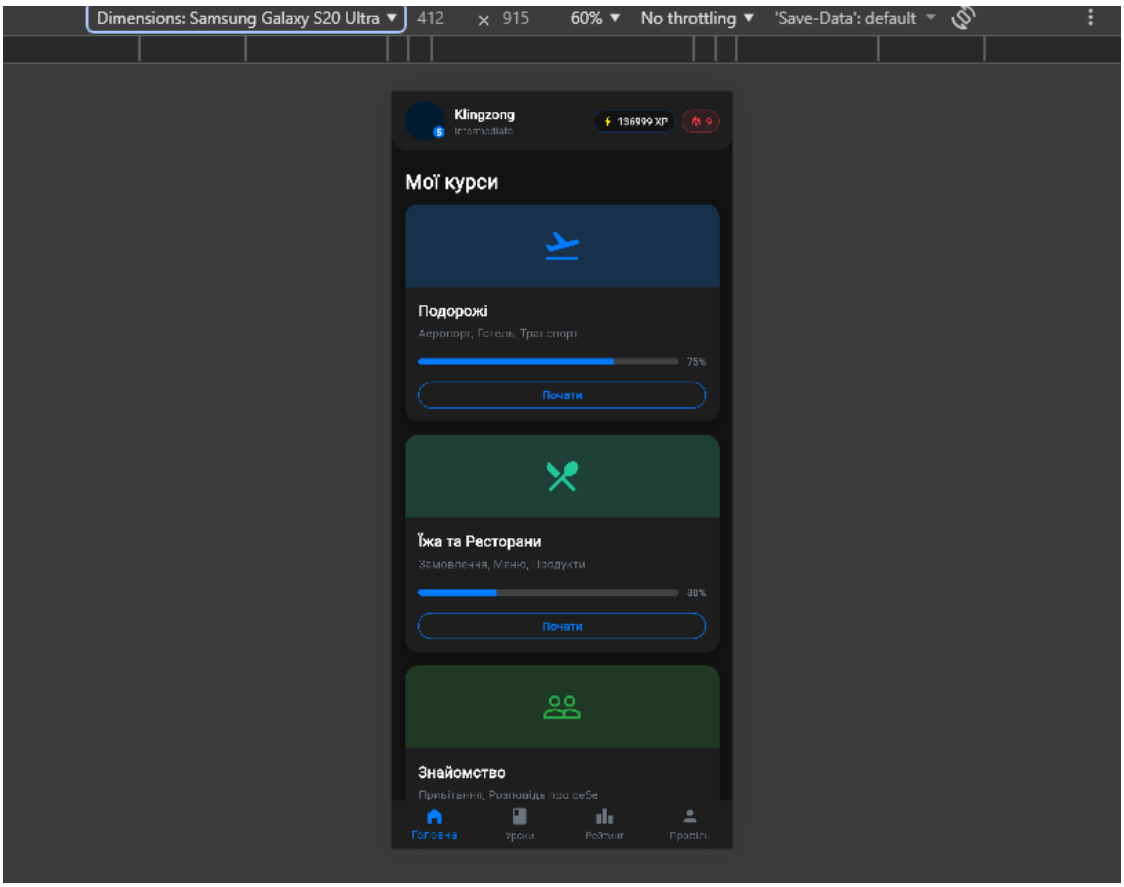


Рисунок 3.7 – Головний екран застосунку “LearnZone” з елементами гейміфікації та структурою модулів (адаптація Android)

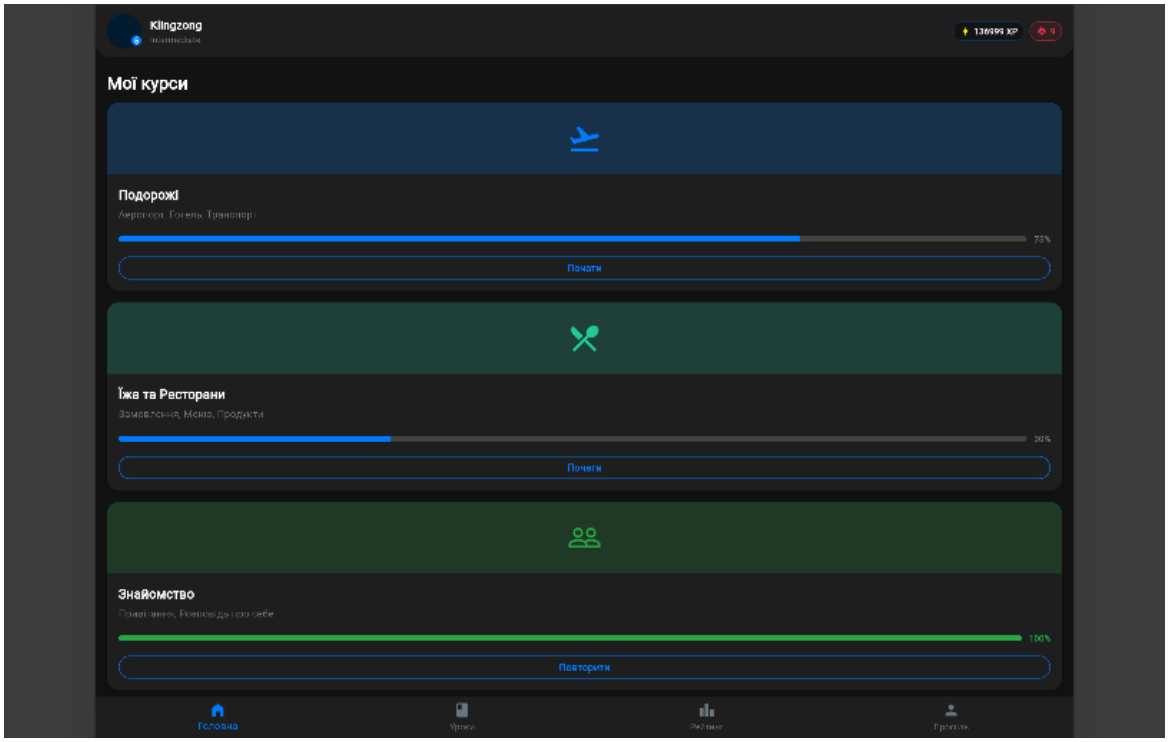


Рисунок 3.8 – Головний екран застосунку “LearnZone” з елементами гейміфікації та структурою модулів (адаптація планшет)

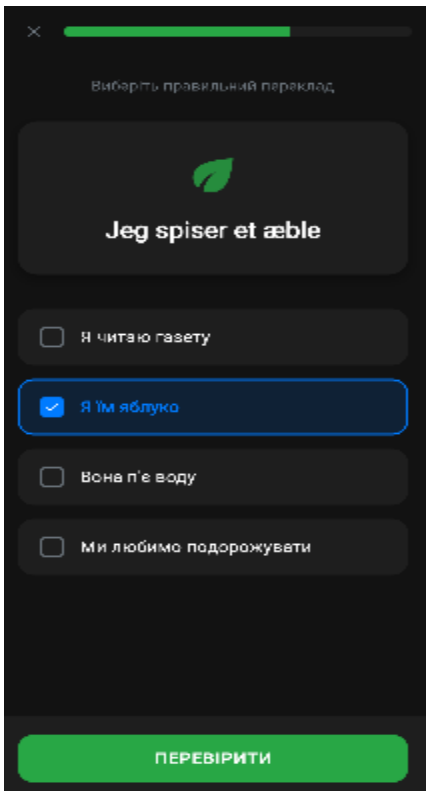


Рисунок 3.9 – Інтерфейс інтерактивного уроку в режимі виконання вправи

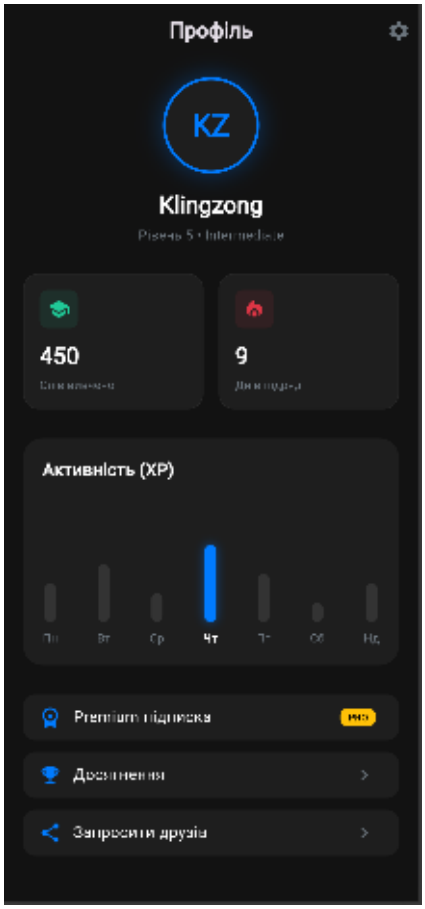


Рисунок 3.10 – Екран профілю користувача з візуалізацією навчальної статистики

Оцінка користувацького досвіду (UX Evaluation). Для попередньої оцінки зручності використання (Usability) було проведено евристичний аналіз інтерфейсу за методикою Якоба Нільсена (10 евристик юзабіліті) [7].

Видимість стану системи: додаток завжди інформує користувача про те, що відбувається (індикатори завантаження, прогрес-бари, повідомлення про успіх/помилку).

Зв'язок з реальним світом: використовуються зрозумілі іконки та термінологія, знайома користувачам (метафора «карток», «уроків»).

Контроль та свобода: користувач може в будь-який момент перервати урок і повернутися назад без фатальних наслідків.

Естетичний та мінімалістичний дизайн: інтерфейс не перевантажений зайвою інформацією, дотримано принципів «повітря» та візуальної ієрархії.

Суб'єктивне тестування на малій фокус-групі (5 осіб) показало, що користувачі легко орієнтуються в навігації та розуміють логіку виконання вправ без додаткових інструкцій. Особливо позитивно було оцінено темну тему інтерфейсу, яка зменшує навантаження на очі у вечірній час.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи магістра було успішно досягнуто поставленої мети – розроблено кросплатформений мобільний застосунок “LearnZone” для вивчення іноземних мов з використанням фреймворку Flutter. Для цього було вирішено всі визначені завдання.

У першому розділі роботи досліджено теоретичні основи розробки мобільних застосунків для навчання. Проведено аналіз сучасних методик вивчення іноземних мов та їх адаптації до мобільних платформ. Виконано порівняльний аналіз провідних додатків-конкурентів (Duolingo, Memrise, Babbel), що дозволило виявити їхні переваги та недоліки. На основі порівняння технологій обґрунтовано вибір фреймворку Flutter.

Другий розділ присвячено проєктуванню застосунку. Сформульовано функціональні та нефункціональні вимоги до системи. Розроблено архітектуру на основі принципів Clean Architecture з паттерном BLoC. Спроєктовано користувацький інтерфейс згідно з Material Design 3 та структуру локальної бази даних SQLite для офлайн-режиму.

У третьому розділі описано реалізацію, тестування та оцінку ефективності застосунку. Імплементовано систему автентифікації, навчальні модулі, п’ять типів інтерактивних вправ, алгоритм інтервального повторення SRS та систему гейміфікації. Проведено комплексне тестування (модульне, віджет-тести, інтеграційне). Виконано оптимізацію продуктивності (60 FPS, 18.4 МБ розмір додатку). Евристичний аналіз підтвердив високу якість UX.

В результаті отримано функціональний застосунок з інтуїтивним інтерфейсом, який повністю відповідає поставленим вимогам. Практична цінність полягає в можливості безпосереднього використання для самостійного вивчення мов та потенціалі комерціалізації в галузі EdTech.

Підсумовуючи, Flutter є потужною платформою для створення освітніх мобільних застосунків. Поєднання сучасних архітектурних патернів з науково

обґрунтованими педагогічними методиками створює основу для високоякісних продуктів. Отриманий досвід може бути застосований для подальшого розвитку “LearnZone” та реалізації інших освітніх проєктів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Krashen S. D. Principles and practice in second language acquisition. Oxford : Pergamon, 1982. 202 p.

2. Dörnyei Z. Motivational strategies in the language classroom. Cambridge University Press, 2009. 100 p.

3. Deterding S., others. From game design elements to gamefulness. *The 15th international academic mindtrek conference*. Tampere, Finland, 2011. URL: https://doi.org/10.1145/2181037.2181040 (дата звернення: 20.10.2025).

4. Woźniak P., Gorzelańczyk E. Optimization of repetition spacing in the practice of learning. *Acta neurobiologiae experimentalis*. 1994. Vol. 54, no. 1. P. 59-62. URL: https://doi.org/10.55782/ane-1994-1003 (дата звернення: 16.10.2025).

5. Clean architecture: a craftsman’s guide to software structure and design. Prentice Hall, 2017. 432 p.

6. Windmill E. Flutter in action. Manning Publications, 2019. 368 p.

7. Nielsen. J. Usability engineering. Boston : Academic Press, 1993. 358 p.

8. Gamma E. Design patterns: elements of reusable object-oriented software. Reading, Mass : Addison-Wesley, 1995. 395 p.

9. Google Material Design Guidelines. URL: https://material.io/design (дата звернення: 15.10.2025).