

**Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення**

**КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»**

**РОЗРОБКА НАВЧАЛЬНОГО МОБІЛЬНОГО ЗАСТОСУНКУ З
ВИКОРИСТАННЯМ МЕТОДУ КАРТОК НА SDK FLUTTER**

**DEVELOPING AN EDUCATIONAL MOBILE APPLICATION
USING THE FLASHCARD METHOD ON THE FLUTTER SDK**

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-42
Мазяр Вадим Миколайович

Керівник:
Бойко Лев Степанович

Кваліфікаційну роботу
допущено до захисту
«__» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Галузь знань: 12 «Інформаційні технології»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

«__» _____ 20__ р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Мазяру Вадиму Миколайовичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка навчального мобільного застосунку з використанням методу карток на SDK Flutter»

Керівник роботи: доцент Бойко Л.С.

затверджені наказом закладу вищої освіти від «31» грудня 2025 р. № 541/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра «29» травня 2026р.

3. Вихідні дані до роботи: методичні матеріали з розробки мобільних застосунків, принципи побудови кросплатформних систем, методи організації локального збереження даних, підходи до створення інтерактивного освітнього контенту та забезпечення адаптивної взаємодії користувача із системою.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз і вибір засобів проектування, опис функціонального наповнення об'єкта проектування, розробка й обґрунтування системного наповнення, оцінка ергономічних та надійнісних параметрів проекрованої системи, функціонально-структурна схема роботи об'єкта проектування

5. Перелік графічного матеріалу: 5 рисунків, 8 лістингів, 3 таблиці.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	<i>Бойко Л. С.</i>		
Специфікація вимог до розробленої системи	<i>Бойко Л. С.</i>		
Розробка об'єкта проектування	<i>Бойко Л. С.</i>		
Нормоконтроль	<i>Суринович О. М.</i>		
Гарант ОП	<i>Ліщина Н. М.</i>		
Показник запозичень тексту	%		
Академічна доброчесність	<i>Бойко Л. С.</i>		

7. Дата видачі завдання «3» лютого 2026 р.

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 14.02.2026 р.	
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 03.03.2026 р.	
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 14.03.2026 р.	
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 14.04.2026 р.	
5	Практична реалізація об'єкта проектування та розробка бази даних	до 20.04.2026 р.	
6	Тестування та налагодження об'єкта проектування	до 04.05.2026 р.	
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедрі	до 29.05.2026 р.	

Здобувач вищої освіти _____

Вадим МАЗЯР

Керівник кваліфікаційної роботи _____

Лев БОЙКО

АНОТАЦІЯ

Мазяр В. М. Розробка навчального мобільного застосунку з використанням методу карток на SDK Flutter. Рукопис.

Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків та списку використаних джерел. У першому розділі проведено аналіз предметної області та розглянуто існуючі аналоги. У другому розділі виконано аналіз вимог до розроблюваного продукту, визначено основні функціональні та нефункціональні вимоги до системи та обґрунтовано вибір технологій і засобів розробки. У третьому розділі наведено високорівневий опис та обґрунтування архітектури мобільного застосунку, наведено особливості реалізації програмного продукту із використанням SDK Flutter та розглянуто приклади використання допоміжних сторонніх бібліотек (пакетів). У висновках узагальнено результати виконаної роботи та визначено перспективи подальшого розвитку програмного продукту.

Ключові слова: Flutter, Provider, SQLite, кросплатформна розробка , Android, iOS, застосунок, архітектура.

ANNOTATION

Maziar V. M. Developing an educational mobile application using the flashcard method on the Flutter SDK. Manuscript.

Bachelor's qualification thesis in the Educational Program “Software Engineering” in the specialty “Software Engineering”. Lutsk National Technical University. Lutsk, 2026.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions, and a list of references. The first chapter provides an analysis of the subject area and reviews existing analogues. The second chapter analyzes the requirements for the product under development, defines the main functional and non-functional requirements for the system, and justifies the choice of development technologies and tools. The third chapter presents a high-level description and rationale for the architecture of the mobile application, details the implementation features of the software product using the Flutter SDK, and discusses examples of utilizing auxiliary third-party libraries (packages). The conclusions summarize the results of the work and outline prospects for further development of the software product.

Keywords: Flutter, Provider, SQLite, Cross-Platform Development, Android, iOS, Application, Architecture.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАВНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ	9
1.1 Аналіз сучасного стану проблеми	9
1.2 Постановка завдання на кваліфікаційну роботу бакалавра.....	12
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	13
2.1 Аналіз, визначення вимог та проєктування програмного забезпечення	13
2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання	18
РОЗДІЛ 3 РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ «EDUCARDS»	22
3.1 Практична реалізація об'єкта проєктування	22
3.2 Тестування та налагодження інформаційно-комп'ютерної системи.....	28
3.3 Архітектура застосунку	30
ВИСНОВКИ.....	38
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	40

ВСТУП

В умовах стрімкого розвитку цифрових технологій та масового поширення мобільних пристроїв особливої актуальності набуває створення програмних засобів, що забезпечують доступність якісної освіти у будь-який час та у будь-якому місці. Мобільні застосунки для навчання перетворились на повноцінний інструмент освітнього процесу, охоплюючи широку аудиторію користувачів різних вікових категорій.

Одним із науково обґрунтованих підходів до організації навчання є метод карток, в основі якого лежать принципи інтервального повторення та активного відтворення інформації. Реалізація цього методу у мобільному форматі підсилює його педагогічний ефект завдяки інтерактивному інтерфейсу, елементам гейміфікації та можливості навчання у зручний для користувача час.

Актуальність теми кваліфікаційної роботи визначається поєднанням доведеної педагогічної ефективності методу карток як інструменту засвоєння навчального матеріалу та технологічної зрілості SDK Flutter як сучасної платформи для розробки мобільних застосунків.

Метою кваліфікаційної роботи є розробка кросплатформного навчального мобільного застосунку з використанням методу карток.

Об'єктом дослідження є процес розробки мобільних інформаційних систем освітнього призначення.

Предметом дослідження є архітектурні, функціональні та програмні рішення під час створення мобільного продукту засобами Flutter із використанням локальної бази даних SQLite та механізмів керування станом.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести аналіз предметної області та існуючих аналогів;
- визначити функціональні та нефункціональні вимоги до системи;
- спроектувати архітектуру мобільної системи;
- реалізувати систему локального збереження даних;
- реалізувати механізми управління станом застосунку;

- розробити інтерактивний користувацький інтерфейс;
- провести тестування та налагодження програмного продукту.

Практичне значення отриманих результатів полягає у створенні мобільного застосунку, який може використовуватися у навчальних цілях як допоміжний інструмент під час дистанційного або ж самостійного навчання.

Апробація результатів роботи здійснюється шляхом практичного використання програмного продукту на мобільних платформах Android та iOS.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАВНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Сучасний ринок цифрових рішень для навчання у 2025-2026 роках активно зростає: найбільший попит мають AI-інструменти, гібридне навчання, VR/AR, мікронавчання та системи персоналізації. В Україні цифровізація освіти теж прискорюється: високий рівень доступу до інтернету та цифрових навичок підтримує перехід до онлайн- та змішаних форматів [1].

Цифрові освітні рішення вже не сприймаються як допоміжні – вони стають базовою інфраструктурою для шкіл, університетів та корпоративного навчання. Зростання ринку підштовхують попит на гнучкість, автоматизацію перевірки, аналітику прогресу та індивідуальні траєкторії навчання [2].

Згідно результатів різноманітних досліджень найбільшого зростання у популярності за останній час досягли AI-помічники для персоналізації, створення контенту та автоматизації оцінювання [1]; платформи змішаного навчання та LMS-системи для організації курсу, завдань і комунікації [3]; VR/AR і симуляції для практичних навичок, особливо там, де важлива візуалізація або безпечне тренування [4]; мобільні рішення для навчання «в коротких сесіях», що зручні для повторення матеріалу та самостійної практики [1]. Проте активне впровадження та використання цифрових рішень не має на меті замінити або витіснити класичне навчання, а радше навпаки – доповнити його.

Класичне навчання сильніше там, де важливі жива взаємодія, дисципліна, миттєвий зворотний зв'язок та соціалізація. Воно краще працює для складних тем, лабораторних робіт та ситуацій, де потрібен нагляд вчителя або викладача [5]. Тоді як, наприклад, мобільні рішення мають перевагу у доступності, гнучкості та персоналізації, оскільки учень може працювати із навчальним матеріалом будь-де, у зручний час та у зручному темпі [6]. Порівняльний аналіз форматів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз форматів навчання

Критерій	Класичне навчання	Цифрові рішення
Гнучкість	Нижча, прив'язка до розкладу	Висока, доступ за допомогою цифрових пристроїв (телефон, планшет, комп'ютер)
Соціальна взаємодія	Дуже висока	Обмежена, але можлива через чати й відео
Персоналізація	Залежить від викладача	Висока завдяки адаптивним системам та AI
Контроль та дисципліна	Вищі в класі/аудиторії	Слабші без самоконтролю
Практичні симуляції	Обмежені ресурсами	Сильні у VR/AR, симуляторах, інтерактиві
Доступність	Потребує фізичної присутності	Працює майже всюди, де є інтернет

В контексті аналізу предметної області варто також розглянути питання доцільності застосування методу карток у процесі навчання. Власне ефективність даного методу ґрунтується на трьох взаємопов'язаних когнітивних механізмах, наукова обґрунтованість яких є детально доведеною в експериментальній психології: активному пригадуванню, інтервальному повторенню та негайному зворотному зв'язку [7].

Емпіричну ефективність методу карток підтверджують і прикладні дослідження в освіті. Зокрема, рандомізоване контрольоване дослідження серед студентів-стоматологів встановило, що навчання за допомогою мобільного застосунку з картками із інтервальним повторенням забезпечує значно вищий рівень утримання знань порівняно з традиційним лекційним методом [8].

Сучасний стан ринку мобільних застосунків демонструє зацікавленість розробників (а отже і користувачів) у реалізації навчальних продуктів на основі

або з використанням методу карток і найбільш популярними є такі застосунки як Anki, Quizlet та Memrise [9].

Anki – найпопулярніший застосунок інтервального повторення серед користувачів, які прагнуть глибокого засвоєння матеріалу, особливо студентів медичних спеціальностей та тих, хто вивчає іноземні мови (рис. 1.1). Він є безкоштовним, з відкритим вихідним кодом і вдосконалювався протягом двох десятиліть. Anki використовує складний алгоритм, побудований на основі SuperMemo 2 (SM-2), а на сьогодні також підтримує новіший алгоритм FSRS, що застосовує машинне навчання для підбору параметрів, які найкраще відповідають історії повторень користувача [9].

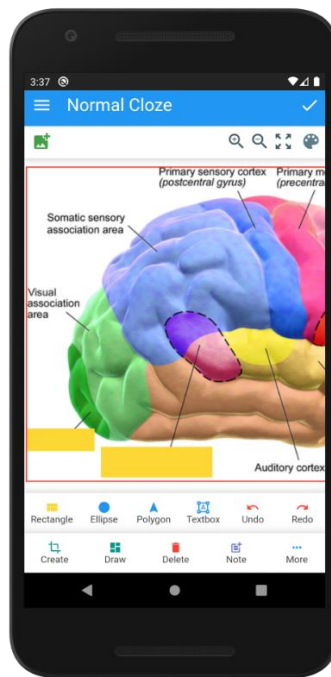


Рисунок 1.1 – Застосунок «Anki» [9]

Quizlet є найпопулярнішим застосунком флеш-карток загалом, з більш ніж 60 мільйонами активних користувачів. Однак він призначений для іншого сценарію використання, ніж Anki: є невимущеним, соціальним та оптимізованим для короткочасного навчання. Quizlet перевів значну частину найкращих функцій під платний доступ за останні роки, а його алгоритм інтервального повторення не є таким досконалим, як в Anki або FSRS [10].

Memrise спеціалізується на вивченні мов та поєднує інтервальне повторення з відеофрагментами носіїв мови та гейміфікацією. Даний застосунок не є універсальним рішенням флеш-карток, але він є сильним гравцем у своїй ніші [11].

Загалом порівняльний аналіз цих та інших застосунків виявляє низку обмежень. Наприклад складний інтерфейс Anki, обмежена функціональність безкоштовної версії Quizlet та вузька спеціалізація Memrise – усе це створює простір для створення нових рішень, які зможуть поєднати переваги різних підходів та вирішити існуючі проблеми.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою кваліфікаційної роботи є розробка кросплатформного навчального мобільного застосунку з використанням методу карток.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести аналіз предметної області та існуючих аналогів;
- визначити функціональні та нефункціональні вимоги до системи;
- спроектувати архітектуру мобільної системи;
- реалізувати систему локального збереження даних;
- реалізувати механізми управління станом застосунку;
- розробити інтерактивний користувацький інтерфейс;
- провести тестування та налагодження програмного продукту.

Послідовне виконання визначених вище завдань дозволить досягти основної мети кваліфікаційної роботи – створити навчальний мобільний застосунок на базі SDK Flutter, що реалізує метод карток як ефективний інструмент засвоєння та закріплення навчального матеріалу, а завдяки використанню інструментів кросплатформної розробки, програмний продукт буде доступним для широкого кола користувачів мобільних пристроїв під керуванням ОС Android та iOS.

РОЗДІЛ 2

СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз, визначення вимог та проєктування програмного забезпечення

На основі аналізу предметної області та визначеної концепції програмного продукту сформульовано перелік функціональних та нефункціональних вимог, яким повинно відповідати розроблюване програмне забезпечення.

Перша функціональна вимога (ФВ) полягає у можливості управління групами карток, що передбачає можливість створення, перегляду, редагування та видалення груп, кожна з яких об'єднує тематично пов'язаний набір навчальних карток. Друга ФВ полягає в керуванні картками у складі групи (карток): у межах кожної групи повинно бути забезпечено можливість створення, перегляду, редагування та видалення окремих карток (CRUD-операції), причому максимальна кількість карток в кожній групі не обмеженою і конкретний набір та послідовність карток визначається користувачем.

Наступні функціональні вимоги пов'язані з режимами взаємодії з картками та «режимами навчання». Так, третя функціональна вимога (ФВ-3) полягає у наданні режиму «перегляду» карток – послідовного відображення контенту обраної групи без перевірки знань, орієнтованого виключно на ознайомлення з матеріалом та\або його повторення. У свою чергу ФВ-4 передбачає наявність режиму перевірки знань, у якому користувачеві послідовно демонструються картки з обраної групи з можливістю вибору одного з двох варіантів: «Так» або «Ні». За кожну правильну відповідь користувачу нараховуватиметься один бал. Наступна функціональна вимога (ФВ-5) полягає в обліку балів за результатами проходження: застосунок повинен зберігати кількість балів, отриманих користувачем за результатами проходження режиму перевірки знань, та надавати можливість їх перегляду.

Наступні функціональні вимоги описують критерії роботи з даними. Так, ФВ-6 визначає необхідність збереження даних, створених або змінених користувачем, між сеансами роботи із застосунком, тобто їх доступність після закриття та повторного відкриття програми. ФВ-7 встановлює що усі дані, які створюються та використовуються застосунком (групи карток, картки, результати проходження), повинні зберігатися виключно на пристрої користувача без передачі на зовнішні сервери.

Нефункціональні вимоги визначають характеристики якості розроблюваного програмного забезпечення та обмеження, у межах яких воно повинно функціонувати. Першою нефункціональною вимогою (НФВ-1) є платформна сумісність: застосунок повинен однаково функціонувати на мобільних пристроях під керуванням ОС Android та iOS, що забезпечується властивостями обраного фреймворку Flutter [12].

Окрім забезпечення кросплатформності, важливим аспектом роботи застосунку є його незалежність від зовнішньої інфраструктури. З огляду на це наступна нефункціональна вимога НФВ-2 полягає в автономності роботи – застосунок повинен функціонувати без підключення до мережі Інтернет, оскільки усі дані зберігаються локально та не передбачається жодної мережевої взаємодії. Це дозволить користуватися застосунком навіть за умови повної відсутності мережі, оскільки достатньо буде лише завантажити програмний і його подальша робота буде повністю автономна з точки зору взаємодії з мережею.

Враховуючи орієнтацію розроблюваного програмного продукту на щоденне використання у навчальному процесі, особливу увагу необхідно приділити зручності взаємодії користувача із системою. Тому нефункціональна вимога НФВ-3 визначає зрозумілість користувацького інтерфейсу, що передбачає логічне групування елементів керування, використання загальноприйнятих для мобільних застосунків принципів навігації та чітку візуальну ієрархію.

Також у контексті інтерфейсу користувача та користувацького досвіду (UI/UX) не менш важливим фактором є забезпечення комфортної роботи застосунку під час взаємодії з картками та переходів між екранами. У зв'язку вимога НФВ-4 встановлює, що час відгуку інтерфейсу на дії користувача (перехід між картками, перехід між екранами, відкриття груп) має бути мінімальним та не створювати відчуття затримки під час взаємодії із застосунком.

Оскільки застосунок використовує локальне зберігання інформації, важливим аспектом також є забезпечення надійності роботи з даними. П'ята нефункціональна вимога (НФВ-5) стосується цілісності даних: інформація, що зберігається у локальній базі даних, не повинна втрачатися або пошкоджуватися у разі коректного завершення роботи застосунку.

Це обмеження забезпечується транзакційністю обраної СУБД SQLite, яка підтримує властивості ACID [13]. Додатково важливим аспектом функціонування мобільного застосунку є забезпечення безпечного зберігання інформації користувача. У зв'язку з цим вимога НФВ-6 визначає конфіденційність даних користувача, яка забезпечується автоматично з огляду на локальне зберігання інформації, що не передається на зовнішні ресурси.

Під час проєктування програмного забезпечення також необхідно враховувати можливість його подальшого розвитку та модернізації. З цієї причини сьома нефункціональна вимога (НФВ-6) стосується розширюваності: архітектура застосунку повинна забезпечувати можливість подальшого додавання нових режимів роботи та функціональних можливостей без значної переробки існуючого коду.

Сформульовані функціональні та нефункціональні вимоги становлять основу для подальшого проєктування архітектури та реалізації програмного забезпечення. Для зручності подальшого використання та забезпечення однозначної ідентифікації окремих вимог їх перелік зведено у таблиці 2.1 і 2.2.

Таблиця 2.1 – Перелік функціональних вимог

Код	Назва	Короткий опис
ФВ-1	Управління групами карток	Створення, перегляд, редагування та видалення груп карток. Кожна група об'єднує тематично пов'язаний набір навчальних карток
ФВ-2	Управління картками у складі групи	CRUD-операції (створення, перегляд, редагування, видалення) над картками у межах обраної групи. Кількість карток у групі не обмежена; набір та послідовність карток визначаються користувачем
ФВ-3	Режим перегляду карток	Послідовне відображення карток обраної групи без перевірки знань. Орієнтований на ознайомлення з матеріалом та його повторення
ФВ-4	Режим перевірки знань	Послідовне демонстрування карток обраної групи з вибором одного з двох варіантів відповіді — «Так» або «Ні». За кожну правильну відповідь нараховується один бал
ФВ-5	Облік балів за результатами проходження	Збереження кількості балів, отриманих користувачем у режимі перевірки знань, із можливістю їх подальшого перегляду
ФВ-6	Збереження даних між сесіями роботи	Дані, створені або змінені користувачем, залишаються доступними після закриття та повторного відкриття застосунку
ФВ-7	Локальне зберігання даних	Усі дані застосунку (групи карток, картки, результати проходження) зберігаються виключно на пристрої користувача без передачі на зовнішні сервери

Таблиця 2.2 – Перелік нефункціональних вимог

Код	Назва	Короткий опис
НФВ-1	Кросплатформність	Однакове функціонування застосунку на мобільних пристроях під керуванням операційних систем Android та iOS, що забезпечується властивостями фреймворку Flutter
НФВ-2	Автономність роботи	Функціонування застосунку без підключення до мережі Інтернет. Усі дані зберігаються локально, мережева взаємодія не передбачена
НФВ-3	Зрозумілість користувацького інтерфейсу	Логічне групування елементів керування, використання загальноприйнятих для мобільних застосунків принципів навігації та чітка візуальна ієрархія
НФВ-4	Швидкодія взаємодії інтерфейсом	Мінімальний час відгуку інтерфейсу на дії користувача (перехід між картками, перехід між екранами, відкриття груп) без відчуття затримки під час типової взаємодії
НФВ-5	Цілісність даних	Інформація у локальній базі даних не повинна втрачатися або пошкоджуватися у разі коректного завершення роботи застосунку. Забезпечується транзакційністю СУБД SQLite та підтримкою властивостей ACID
НФВ-6	Конфіденційність даних користувача	Дані користувача не передаються на зовнішні ресурси та зберігаються виключно на пристрої
НФВ-7	Розширюваність	Архітектура застосунку забезпечує можливість додавання нових режимів роботи та функціональних можливостей без значної переробки існуючого коду

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

У якості основного інструменту розробки було обрано Flutter SDK – відкритий набір інструментів для кросплатформного програмування від Google. За допомогою Flutter створення застосунків для таких ОС як Android, iOS, Web, Windows, Linux та macOS з єдиної кодової бази мовою [12].

Архітектурна особливість Flutter SDK полягає у тому, що замість використання нативних компонентів цільової платформи фреймворк здійснює відображення інтерфейсу через власний рушій рендерингу, що забезпечує візуальну й поведінкову консистентність на усіх цільових платформах [12]. Код на Dart – мові програмування, яка використовується у Flutter SDK [12] – компілюється напряму в нативний бінарний код (AOT-компіляція), що дозволяє досягти продуктивності, наближеної до нативної [14].

Перевагами нативної розробки окрім максимальної продуктивності також є прямий доступ до усіх можливостей цільової платформи та найбільш узгоджений з конкретною ОС користувацький досвід. Водночас нативна розробка має суттєві недоліки в контексті поставленого завдання:

- дублювання кодової бази. Кожна платформа потребує окремої реалізації як логіки, так і інтерфейсу, що збільшує обсяг роботи приблизно вдвічі ускладнює підтримку застосунку;
- збільшення часових затрат. Реалізація однакового функціоналу з використанням різних технологічних стеків під кожен цільову платформу збільшує загальний час розробки.

Flutter, у свою чергою, дає змогу повторно використовувати близько 95% коду між Android та iOS, оскільки бізнес-логіка та елементи інтерфейсу описуються одним кодом [15]. Це робить його раціональним вибором для застосунків, де глибока інтеграція зі специфічними платформними API не є визначальною.

Звісно ж, окрім Flutter, на ринку існує низка інших, альтернативних крос-платформних інструментів.

React Native – фреймворк від компанії Meta, що використовує мову JavaScript та працює за принципом трансформації JavaScript-компонентів у нативні елементи через так званий «міст» (bridge) [16].

Хоча останні версії React Native значно покращили продуктивність завдяки впровадженню нової архітектури, він поступається Flutter у плані візуальної консистентності між платформами, оскільки покладається на нативні компоненти, які різняться за поведінкою на Android та iOS [17]. Крім того, Flutter випереджає React Native за активністю розробки та темпами розвитку: за даними на початок травня 2026 року, репозиторій Flutter мав понад 175 тисяч зірок на GitHub [18] проти 125 тис. у React Native [19].

Для застосунку з картками, де важливою є передбачувана поведінка анімацій і єдиний вигляд на всіх пристроях, власний рушій рендерингу Flutter є перевагою.

Kotlin Multiplatform (KMP) – це технологія від компанії JetBrains, що дозволяє розділяти бізнес-логіку між платформами, проте потребує реалізації нативного інтерфейсу окремо для Android та iOS [20]. Такий підхід зменшує дублювання бізнес-логіки, проте не вирішує проблему з необхідністю реалізації інтерфейсу користувача окремо під кожен із цільових платформ.

Ionic – гібридне рішення на основі веб-технологій (HTML, CSS, JavaScript), у якому застосунок фактично виконується у вбудованому веб-перегляді [21]. Незважаючи на відносну простоту у вивченні для веб-розробників, його продуктивність та користувацький досвід (User experience, UX) істотно поступаються Flutter.

Таким чином, Flutter оптимально поєднує крос-платформну розробку, високу продуктивність рендерингу, єдиний вигляд інтерфейсу на усіх цільових платформах та активну підтримку спільнотою. Окрім того, Flutter має значний каталог готових компонентів – віджетів, що суттєво зменшує час розробки інтерфейсу; а функція Hot Reload, який дозволяє миттєво бачити результат змін

у кодi без перезапуску застосунку, суттєво пришвидшує розробку застосунків загалом [12].

У Flutter існує декілька підходів до керування станом: від найпростішого `setState` до спеціалізованих бібліотек на кшталт `Provider`, `Riverpod`, `Bloc`, `GetX` тощо [12], проте для розробки цільового застосунку було обрано саме пакет `Provider` [22].

`Provider` являє собою «обгортку» над низькорівневим механізмом `InheritedWidget` у Flutter і спрощує передачу даних та реалізує механізм реактивного оновлення даних у дереві віджетів [22]. Перевагами цього пакету є простота та невелика кількість шаблонного коду, тіста інтеграція з Flutter, стабільність та якісна документація.

Рішення розробляти застосунок без серверної (back-end) частини не було компромісом, а свідомим вибором, який впливає з характеру предметної області та забезпечує низку технічних та користувацьких переваг.

Метод карток за своєю суттю не потребує взаємодії з віддаленим сервером під час навчального процесу, оскільки користувачеві достатньо мати локальний набір карток для тренування. Реалізація збереження карток та прогресу на пристрої користувача гарантує, що повну функціональність застосунку за будь-яких умов зв'язку: у транспорті, на природі, у регіонах із нестабільним інтернетом або в умовах роумінгу.

Для навчальних застосунків саме офлайн-доступність є однією з ключових переваг, що забезпечує безперервність процесу засвоєння матеріалу й підвищує ефективність запам'ятовування.

У якості локального сховища було обрано `SQLite` – реляційну вбудовану базу даних, підтримка якої у Flutter забезпечується пакетом `sqflite` [23]. Обґрунтуванням такого вибору стало те, що на відміну від деяких інших альтернатив (наприклад, `Hive` чи `Isar`), `SQLite` має десятиліття експлуатації у промислових системах та добре задокументована. Також ця база даних є стандартним вбудованим сховищем як на Android, так і на iOS, а пакет `sqflite`

пропонує єдиний API для обох платформ із підтримкою фонового виконання запитів.

Важливим є те, що SQLite забезпечує ACID-транзакції, підтримує JOIN-запити, агрегатні функції та індекси, що дозволяє ефективно реалізовувати вибірки карток за різними критеріями (наприклад, картки, які підлягають повторенню сьогодні) без повного перебору усіх даних.

РОЗДІЛ 3

РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ «EDUCARDS»

3.1 Практична реалізація об'єкта проєктування

Екран з картками реалізовано із використанням модульного підходу: він складається з кількох незалежних компонентів, кожен з яких відповідає за окремий аспект функціональності. До основних компонентів належать QuizScreen (основний батьківський контейнер екрану, що створює локальний екземпляр QuizProvider та ініціює завантаження питань); QuizBody (забезпечує компоновку «карткової» області та панель кнопок); TextCard (компонент окремої картки запитання, що відображає контент у режимі, придатному для прокручування); DescriptionDialog (модальний діалог із поясненням, що з'являється при неправильній відповіді у разі увімкненої опції підказок); QuizController (клас, що оркеструє події інтерфейсу та делегує виконання операцій провайдеру та відповідним сервісам); QuizEndDialog (діалогове вікно із підсумковим результатом, що відображається після проходження усіх запитань). Екран із запитаннями наведено на рисунку 3.1.



Рисунок 3.1 – Екран з картками на тему «Інформатика»

Центральну частину екрану займає область карткового свайпера, у якій одночасно відображається одна картка з текстом поточного питання. Нижче розташовано панель з двома кнопками відповіді – «Ні» і «Так», що дублюють функціональність свайп-жестів (рис. 3.2) та забезпечують альтернативний спосіб введення відповіді. Після завершення сесії панель кнопок автоматично приховується.

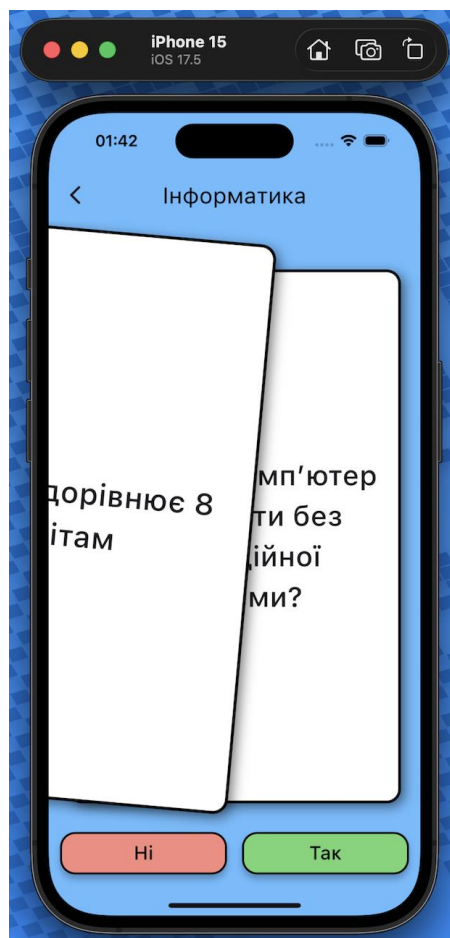


Рисунок 3.2 – Демонстрація можливостей використання свайп-жестів для взаємодії з картками

Завдяки використанню віджета CardSwiper, було реалізовано відображення стеку карток з ефектом тривимірної глибини, обробку горизонтальних свайп-жестів у двох напрямках (ліворуч – заперечна відповідь та праворуч – стверджувальна), анімовану зміну карток при переході до наступного

питання та програмне керування картками завдяки екземпляру класу `CardSwiperController`.

Обробка відповіді користувача (як отриманої через свайп-жест, так і через натискання кнопки) було реалізовано класі-контролері `QuizController`. Цей клас містить два публічні методи-обробники: `onSwipe()`, що викликається віджетом `CardSwiper` при завершенні свайп-жесту, та `onTapAnswerButton()`, що викликається при натисканні кнопки «Так» або «Ні». Обидва обробники делегують виконання основної логіки приватному методу `_processPick()`, що уніфікує обробку незалежно від способу введення (ліст. 3.1).

Лістинг 3.1 – Код методу `_processPick`

```
void _processPick(int index, bool answer) {
    final question = provider.questions[index];
    final isCorrect = question.correctAnswer == answer;

    provider.pickAnswer(index, answer);

    if (isCorrect) {
        SoundService.instance.playCorrectSound();
    } else {
        SoundService.instance.playErrorSound();
        VibrationService.instance.vibrateError();
        if (settings.showTipsOnWrongAnswers) {
            _showDescriptionDialog(question.comments);
        }
    }

    if (provider.isFinished) {
        onEnd();
    }
}
```

кінець лістингу 3.1

Варто зазначити, що особливістю реалізації методу карток у застосунку `EduCards` є «багатоканальний» зворотний зв'язок, що супроводжує кожну

відповідь користувача. Так, було реалізовано три «канали» зворотного зв'язку: звуковий (через SoundService), тактильний (через VibrationService) та візуальний (через DescriptionDialog).

Звуковий зворотний зв'язок залежить від правильності відповіді. Так, якщо було надано правильну відповідь, відтворюється файл sound/correct.mp3, у випадку помилки – sound/error.mp3. Для уникнення взаємного перекриття звуків при швидких відповідях у сервісі SoundService було реалізовано використання окремих екземплярів AudioPlayer.

Тактильний зворотний зв'язок реалізовано лише для неправильних відповідей у вигляді м'якого вібраційного імпульсу – за умови, що пристрій підтримує вібрацію та користувач увімкнув відповідну опцію в налаштуваннях.

Візуальний зворотний зв'язок у вигляді DescriptionDialog відображається лише при неправильній відповіді та за умови увімкненої опції відображення підказок (рис. 3.3).

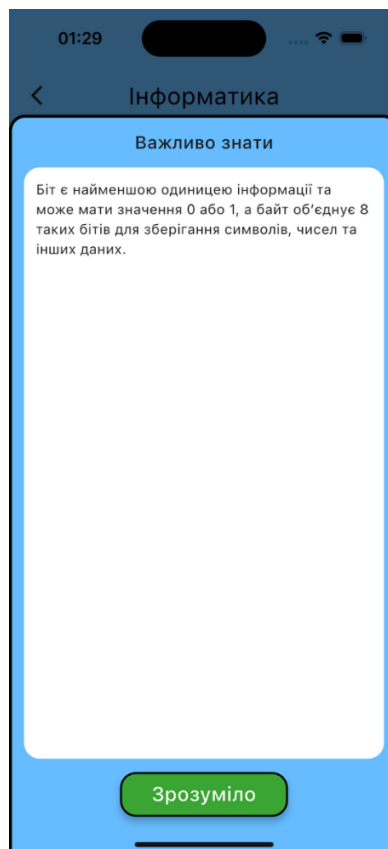


Рисунок 3.3 – Демонстрація візуального зворотного зв'язку у випадку неправильної відповіді

Локалізація мобільних застосунків є важливим аспектом їх розробки, оскільки забезпечує адаптацію інтерфейсу до мовного та культурного середовища кінцевого користувача.

Хоча на поточному етапі застосунок EduCards підтримує лише українську мову, його архітектура передбачає можливість додавання інших мов без потреби рефакторингу основного коду.

Механізм локалізації було реалізовано з використанням пакету intl та згенерованих файлів, розміщених у директорії lib/generated/intl.

До основних артефактів локалізації належать: L10n – єдина точка доступу до перекладених рядків з коду застосунку; AppLocalizationDelegate – клас, що інтегрує локалізацію зі стандартним механізмом Flutter; файл messages_uk.dart – згенерований модуль з українськими перекладами; файл messages_all.dart – згенерований реєстр усіх доступних мов застосунку.

При використанні пакету intl підключення локалізації виконується у кореневому віджеті – MaterialApp – шляхом передавання масиву делегатів локалізації та переліку підтримуваних мов.

Лістинг 3.2 демонструє підключення механізму локалізованої стрічки.

Лістинг 3.2 – Підключення механізму локалізації у застосунку

```
MaterialApp(  
  ....  
  localizationsDelegates: const [  
    S.delegate,  
    GlobalMaterialLocalizations.delegate,  
    GlobalWidgetsLocalizations.delegate,  
    GlobalCupertinoLocalizations.delegate,  
  ],  
  supportedLocales: [  
    Locale('uk'), // Ukrainian
```

```
],
....
)
```

кінець лістингу 3.2

Звертання до локалізованих рядків з коду віджетів реалізується через виклик `L10n.of(context)`, що повертає екземпляр класу `L10n` для поточної локалі. Приклад використання наведено у лістингу 3.3.

Лістинг 3.3 – Приклад отримання локалізованої стрічки

```
....
Text(L10n.of(context).buttonOk)
....
```

кінець лістингу 3.3

Закладена в архітектурі застосунку модель локалізації забезпечує мінімальні витрати на додавання нових мов. Для введення підтримки нової мови (наприклад, англійської) необхідно виконати наступні дії: створити новий ARB-файл (`intl_en.arb`) з перекладами всіх наявних ключів; повторно запустити генератор коду, який автоматично створить файл `messages_en.dart`; додати рядок `Locale('en')` до переліку підтримуваних мов у віджеті `MaterialApp`. При цьому зміни в решті коду застосунку не потрібні, оскільки усі звертання до перекладів через клас `L10n` залишаються незмінними, а вибір конкретного перекладу здійснюється автоматично на основі системних налаштувань пристрою.

Таким чином, обраний підхід до локалізації забезпечує одночасно прозорість поточної реалізації та достатню гнучкість для майбутнього масштабування застосунку на нові мови та регіони.

3.2 Тестування та налагодження інформаційно-комп'ютерної системи

Завершальним етапом розробки EduCards стало його тестування та налагодження з метою яких була перевірка відповідності реалізованого рішення висунутим функціональним та нефункціональним вимогам, виявлення та усунення дефектів, а також підтвердження стабільності роботи. Оскільки застосунок було розроблено із застосуванням кросплатформного SDK Flutter і призначено для функціонування на пристроях під керуванням ОС Android та iOS, особливої уваги потребувало тестування сумісності: перевірка коректності роботи програмного рішення на пристроях із різними апаратними характеристиками, розмірами екрану та версіями операційних систем.

Однією із переваг використання Flutter SDK є те, що цей фреймворк надає набір інструментів для забезпечення трьох рівнів автоматизованого тестування програмного забезпечення: модульного (unit), віджетного (widget) та інтеграційного (integration). Модульне тестування спрямоване на перевірку коректності роботи окремих функцій, методів або класів ізольовано від решти системи. Перевірка віджетів надає змогу перевірити правильність відображення та поведінки окремих елементів інтерфейсу користувача. Інтеграційне тестування забезпечує перевірку застосунку або його значної частини в цілому, відтворюючи реальні сценарії взаємодії.

Особливу увагу було приділено тестуванню сумісності та продуктивності застосунку на реальних мобільних пристроях, що дало змогу оцінити поведінку EduCards в умовах експлуатації на реальних фізичних пристроях, а не на емуляторах. Для цього застосунок було встановлено та протестовано на п'яти мобільних пристроях, які охоп на

Для перевірки сумісності застосунку встановлено та протестовано на п'яти мобільних пристроях, які охоплюють усі основні цінові та апаратні категорії – від преміальних до бюджетних флагманських моделей до бюджетних, а також обидві цільові операційні системи: Android та iOS. Такий добір пристроїв

дозволяє оцінити роботу програмного рішення в умовах суттєвої різниці в обчислюваних потужностях, обсягу оперативної пам'яті та роздільної здатності екрану. Характеристики обраних пристроїв наведено в таблиці 3.1.

Таблиця 3.1. – Характеристики пристроїв, використаних для тестування

Пристрій	Категорія	ОС	Процесор	ОЗП	Екран
Apple iPhone 17 Pro Max	Преміальний (флагман)	iOS	Apple A19 Pro	12 ГБ	6,9" OLED
Apple iPhone XS	Флагман попереднього покоління	iOS	Apple A12 Bionic	4 ГБ	5,8" OLED
Google Pixel 7a	Середній клас	Android	Google Tensor G2	8 ГБ	6,1" OLED
Xiaomi Mi 9	Флагман попереднього покоління	Android	Qualcomm Snapdragon 855	6 ГБ	6,39" AMOLED
Motorola Moto G24	Бюджетний	Android	MediaTek Helio G85	4 ГБ	6,56" IPS LCD

Наведений перелік пристроїв для тестування є цілком репрезентативним. Так, Apple iPhone 17 Pro Max представляє сучасні високопродуктивні флагманські пристрої, тоді як Apple iPhone XS дає змогу оцінити роботу застосунку на старішій моделі екосистеми iOS.

У сегменті Android Google Pixel 7a відповідає пристроям середнього цінового діапазону, у той час як Xiaomi Mi 9 – це флагманська модель попереднього покоління. Motorola Moto G24 – це сучасний бюджетний пристрій з обмеженими апаратними ресурсами. Такий набір пристроїв відповідає реальному розподілу смартфонів серед цільової аудиторії навчального застосунку.

За результатами тестування встановлено, що на всіх перелічених пристроях застосунок запускався коректно, інтерфейс користувача відображався без графічних спотворень та адаптувався до різних розмірів і співвідношень сторін екрана, а основні функціональні можливості працювали відповідно до висунутих вимог. Швидкодія застосунку на усіх пристроях, зокрема й на бюджетних та старіших моделях, була оцінена як задовільна: переходи між екранами, прокручування списків та робота з картками відбувалися плавно, без помітних затримок або підвисань. Це підтверджує, що обраний фреймворк забезпечує прийнятний рівень продуктивності на широкому спектрі апаратного забезпечення.

Налагодження застосунку здійснювалося паралельно з процесами розроблення та тестування. Для виявлення й усунення дефектів використано штатні інструменти екосистеми Flutter: інтегрований у середовище розробки налагоджувач(debugger) з можливістю встановлення точок зупинки та покрокового виконання коду, механізм гарячого перезавантаження (hot reload), що дає змогу оперативно вносити та перевіряти зміни без повного перезапуску застосунку та також набір інструментів Flutter DevTools для аналізу продуктивності, інспектування дерева віджетів та контролю використання оперативної пам'яті. Виявлені під час тестування невідповідності та помилки фіксувалися й усувалися шляхом внесення відповідних змін до програмного коду з подальшою повторною перевіркою відповідних сценаріїв.

Проведене тестування та налагодження дозволило переконатися у працездатності, стабільності та достатній продуктивності розробленої інформаційно-комп'ютерної системи на пристроях різних категорій, що підтверджує її готовність до практичного використання.

3.3 Архітектура застосунку

Якість програмного забезпечення значною мірою визначається обґрунтованістю його архітектурного дизайну. В основі архітектури застосунку

EduCards лежить поєднання шаблону Provider + Repository, який забезпечує чіткий розподіл «обов'язків» між програмними компонентами та сприяє дотриманню принципу єдиної відповідальності – Single Responsibility Principle.

Архітектура застосунку має багат шарову структуру, у якій кожен рівень (шар) виконує строго визначені функції та взаємодіє лише з безпосередньо суміжними шарами. Такий підхід забезпечує низький рівень зв'язності між компонентами та високий рівень внутрішньої узгодженості у межах кожного шару. Загальна схема архітектури застосунку наведена на рисунку 3.4.

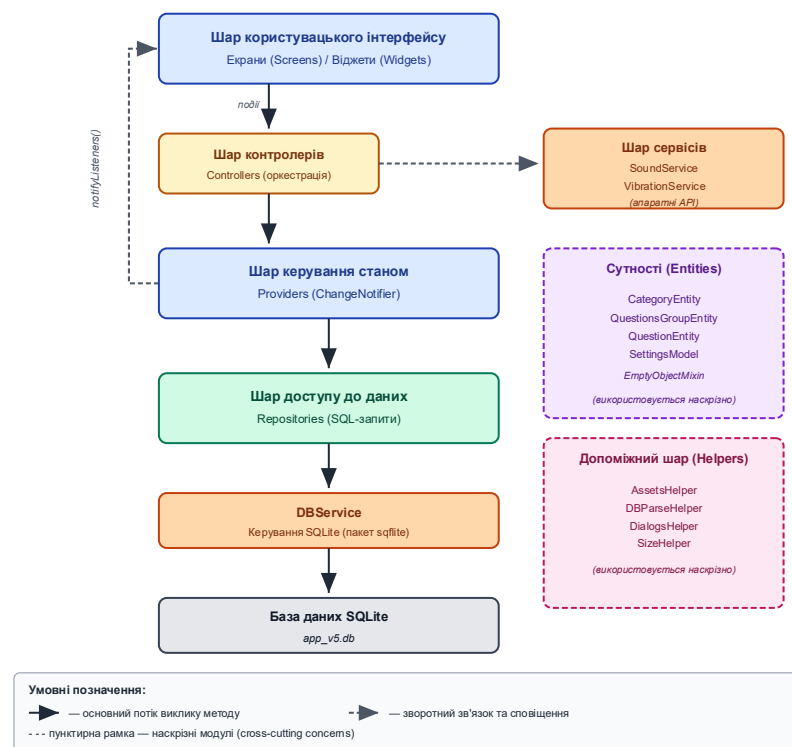


Рисунок 3.4 – Архітектурна схема застосунку EduCards

Рух даних у застосунку відбувається у двох напрямках: прямому та зворотному. Прямий потік – виклик методів, має низхідну спрямованість: від користувацького інтерфейсу через шар управління станом до бази даних. Зворотний потік – сповіщення про зміну стану, має висхідну спрямованість і реалізується через механізм `notifyListeners()` пакету `provider`. Розглянемо призначення кожного шару архітектури докладніше.

Інтерфейс користувача – верхній шар архітектури, який складається з екранів (screens) та віджетів (widgets). Цей архітектурний рівень відповідає виключно за відображення даних та обробку дій користувача. У ньому не реалізовано елементів бізнес-логіки, і компоненти інтерфейсу не мають прямого доступу до бази даних оскільки уся взаємодія з даними здійснюється виключно через шар керування станом.

Віджети «підписуються» на зміни стану провайдерів: при надходженні сповіщення `notifyListeners()` відповідні віджети автоматично перебудовуються, що забезпечує реактивний характер інтерфейсу.

Шар контролерів. Контролери є допоміжним шаром, що відповідає за «оркестрацію» подій інтерфейсу та навігацію між екранами. На цьому рівні не відбувається реалізації бізнес-логіки у чистому вигляді. Цей архітектурний шар виступає посередником між діями користувача (наприклад, натисканням кнопки або свайп-жестом) та викликами методів провайдерів або переходами на інші екрани.

Винесення цієї логіки в окремий шар дає змогу зберегти декларативний характер коду віджетів та не ускладнювати вихідний код, який відповідальний лише за відображення даних.

Керування станом – цей шар є центральним елементом архітектури та реалізує шаблон `Observer` через за допомогою класу `ChangeNotifier` з пакету `provider`. Завданням цього архітектурного рівня є зберігання поточного стану застосунку, координація запитів до шару доступу до даних, а також сповіщення підписаних віджетів про зміни стану. Прикладом такого провайдеру є клас `CategoryProvider`, який призначений для доступу до переліку груп (категорій) запитань (ліст. 3.4).

Лістинг 3.4 – Клас `CategoryProvider`

```
class CategoryProvider extends ChangeNotifier {  
  final List<QuestionsGroupEntity> _questionGroups = [];  
  
  var _category = CategoryEntity.buildEmpty();
```

```
/// Getters & Setters
```

```
List<QuestionsGroupEntity> get questionGroups => _questionGroups;
```

```
CategoryEntity get category => _category;
```

```
set category(CategoryEntity value) {
    _category = value;
    notifyListeners();
}
```

```
/// Methods
```

```
QuestionsGroupEntity groupByIndex(int index) {
    if ((index < 0) || (index > (_questionGroups.length - 1))) {
        return QuestionsGroupEntity.buildEmpty();
    }
}
```

```
    return _questionGroups[index];
}
```

```
Future<void> fetchGroups() async {
    final result = await CategoryRepository.fetch(_category.id);

    _questionGroups
        ..clear()
        ..addAll(result);
    notifyListeners();
}
}
```

кінець лістингу 3.4

Шар доступу до даних – репозиторії. Репозиторії інкапсулюють логіку взаємодії з базою даних. Кожен репозиторій реалізує набір статичних методів, що виконують SQL-запити до відповідних таблиць та повертають результати у вигляді окремих об'єктів або списків екземплярів відповідних класів. Такий підхід дозволяє ізолювати специфіку зберігання даних від решти застосунку: у разі потреби заміни джерела даних (наприклад, переходу від локальної СУБД до віддаленого REST-сервісу) модифікації потребуватиме лише даний

архітектурний рівень, тоді як інші шари залишаються без змін. Приклад, репозиторію для отримання переліку категорій запитань наведено у лістингу 3.5.

Лістинг 3.5 – Клас репозиторію CategoryRepository

```
class CategoryRepository {
    static Future<List<QuestionsGroupEntity>> fetch(int categoryId) async {
        final db = DBService().db;
        if (db == null) {
            return [];
        }

        final results = await db.rawQuery(
            'SELECT
                id, object_key, category_id, title, logo, answered, total FROM
                question_groups WHERE category_id = ?',
            [categoryId],
        );
        if (results.isEmpty) {
            return [];
        }

        final result = <QuestionsGroupEntity>[];
        for (var i = 0; i < results.length; i++) {
            result.add(
                QuestionsGroupEntity.buildFromDB(results.elementAt(i))
            );
        }

        return result;
    }
}
```

кінець лістингу 3.5

Шар сервісів. Сервіси є обгортками над зовнішніми інтеграціями та системними можливостями пристрою. На відміну від репозиторіїв, що працюють із даними застосунку, сервіси здійснюють взаємодію з апаратним забезпеченням або системними API. У застосунку реалізовано три сервіси: DBService (для роботи з БД SQLite), SoundService (відтворення звукового зворотного зв'язку засобами пакета audioplayers) та VibrationService (керування тактильним

відгуком засобами пакета vibration) (ліст. 3.6). Варто зазначити, що з метою спрощення використання сервісів, в межах даної роботи усі сервіси реалізовані із використання шаблону Singleton.

Лістинг 3.6 – Сервіс VibrationService

```
class VibrationService {
  bool _hasVibrator = false;

  static final _instance = VibrationService._internal();

  late VibrationService _player;

  factory VibrationService() {
    return _instance;
  }

  /// Methods

  VibrationService._internal() {
    Vibration.hasVibrator().then((v) => _hasVibrator = v);
  }

  vibrateError() async {
    if (!_hasVibrator) {
      return;
    }

    Vibration.vibrate(preset: VibrationPreset.softPulse);
  }
}
```

кінець лістингу 3.6

Шар сутностей. Сутності являють собою звичайні класи, що моделюють предметну область застосунку та використовуються наскрізно – тобто доступні на усіх рівнях. Такі класи не містять складної логіки, окрім фабричних методів для створення об'єктів із мап (наприклад, Map<String, dynamic>), отриманих із бази даних. До основних сутностей належать CategoryEntity (категорія тем

верхнього рівня), QuestionsGroupEntity (група питань у межах категорії), QuestionEntity (окреме питання вікторини) (ліст. 3.7) та SettingsModel (модель користувацьких налаштувань). Для безпечної роботи з порожніми об'єктами використано міксин EmptyObjectMixin.

Лістинг 3.7 – Клас сутності QuestionEntity

```
class QuestionEntity {
    final int id;
    final String objectKey;
    final int questionGroupId;
    final String value;
    final QuestionType type;
    final bool correctAnswer;
    final String comments;

    QuestionEntity({
        required this.id,
        required this.objectKey,
        required this.questionGroupId,
        required this.value,
        required this.type,
        required this.correctAnswer,
        this.comments = "",
    });

    factory QuestionEntity.buildFromDB(Map<String, Object?> data) {
        final questionTypeRaw = DBParseHelper.stringValue(data, 'type');

        return QuestionEntity(
            id: DBParseHelper.intValue(data, 'id'),
            objectKey: DBParseHelper.stringValue(data, 'object_key'),
            questionGroupId: DBParseHelper.intValue(
                data, 'questions_group_id'),
            value: DBParseHelper.stringValue(data, 'question'),
            type: questionTypeRaw == 'image' ? QuestionType.image : QuestionType.text,
            correctAnswer: DBParseHelper.boolValue(data, 'answer', false),
            comments: DBParseHelper.stringValue(data, 'comments'),
        );
    }
}
```

кінець лістингу 3.7

Допоміжний шар – хелпери. Даний шар містить набір функцій-утиліт загального призначення, які використовуються у різних шарах застосунку: формування шляхів до медіаресурсів (AssetsHelper) (ліст. 3.8), відображення діалогових вікон (DialogsHelper) та ін.

Лістинг 3.8 – Клас SizeHelper

```
class SizeHelper {
  static const minBottomPadding = 15.0;

  static double getMediaBottomPadding(BuildContext context) {
    final mediaPaddingBottom = MediaQuery.
                                     of(context).
                                     padding.bottom;
    return mediaPaddingBottom < minBottomPadding
           ? minBottomPadding
           : mediaPaddingBottom;
  }

  static double getMediaTopPadding(BuildContext context) {
    return MediaQuery.of(context).padding.top;
  }
}
```

кінець лістингу 3.8

Обраний архітектурний підхід є оптимальним для застосунку даного класу з огляду на низку чинників. По-перше, комбінація Provider + Repository є офіційно рекомендованим підходом для Flutter-застосунків середньої складності. По-друге, чіткий розподіл шарів забезпечує тестованість компонентів незалежно один від одного, що є важливою передумовою якісного програмного забезпечення. По-третє, ізоляція шару доступу до даних дає змогу в перспективі розширити застосунок підтримкою синхронізації з віддаленим сервером або іншим джерелом контенту без значних змін у решті коду.

ВИСНОВКИ

На основі виконаної кваліфікаційної роботи було досягнуто поставлену мету – розроблено кросплатформний навчальний мобільний застосунок EduCards з використанням методу карток на базі SDK Flutter.

У ході виконання роботи проведено аналіз предметної області та сучасних навчальних мобільних застосунків, зокрема Anki, Quizlet і Memrise. Аналіз дозволив визначити їхні переваги та недоліки, що стало основою для формування концепції власного програмного продукту, орієнтованого на простоту використання, автономність роботи та ефективне засвоєння навчального матеріалу.

На основі результатів аналізу було визначено функціональні та нефункціональні вимоги до системи. Зокрема, передбачено можливості управління групами карток і окремими картками, реалізовано режими перегляду та перевірки знань, облік результатів навчання, локальне збереження даних, а також забезпечено вимоги щодо кросплатформності, автономності, швидкодії, цілісності та конфіденційності даних.

У процесі проєктування було розроблено багатoshарову архітектуру застосунку на основі підходу Provider + Repository, що забезпечує чітке розмежування відповідальностей між компонентами системи, спрощує супровід програмного коду та створює умови для подальшого розширення функціональності застосунку.

Для забезпечення автономної роботи програмного продукту реалізовано систему локального збереження даних на базі SQLite із використанням пакета sqflite. Усі навчальні матеріали, налаштування та результати користувача зберігаються безпосередньо на пристрої, що дозволяє використовувати застосунок без підключення до мережі Інтернет.

У роботі реалізовано механізми управління станом застосунку за допомогою пакета Provider, що забезпечує реактивне оновлення інтерфейсу та

ефективну взаємодію між користувацьким інтерфейсом, бізнес-логікою і сховищем даних. Такий підхід дозволив спростити підтримку коду та підвищити стабільність роботи системи.

Розроблено інтерактивний користувацький інтерфейс із підтримкою жестів свайпу, кнопок швидкої відповіді, візуального, звукового та тактильного зворотного зв'язку. Інтерфейс забезпечує зручну взаємодію користувача з навчальним матеріалом та сприяє підвищенню ефективності процесу запам'ятовування інформації.

На завершальному етапі виконано тестування та налагодження програмного продукту на мобільних пристроях під керуванням Android та iOS різних цінових категорій. Результати тестування підтвердили коректність роботи реалізованих функцій, стабільність функціонування, адаптивність інтерфейсу та достатній рівень продуктивності застосунку для практичного використання.

Таким чином, усі поставлені завдання кваліфікаційної роботи були успішно виконані, а розроблений мобільний застосунок може використовуватися як ефективний інструмент для самостійного навчання, повторення та закріплення навчального матеріалу на мобільних пристроях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Освітні тренди: як учні навчаються в цифрову епоху. URL: <https://osvitoria.media/experience/osvitni-trendy-2025-yak-uchni-navchayutsya-v-tsyfrovu-epohu/> (дата звернення: 14.02.2026).
2. Шипуліна Юлія, Семенютін Артем. Стратегічні тенденції цифрової трансформації освітньої діяльності та розвитку ринку EDTECH. Економіка та суспільство, вип. 80 (Жовтень), 2025, doi: <https://doi.org/10.32782/2524-0072/2025-80-119>.
3. Цифровізація українських закладів освіти: готовність, інструменти, III та бар'єри. URL: <https://www.bdo.ua/uk-ua/insights-2/information-materials/2025/digital-shift-in-ukrainian-education-preparedness-technologies-ai-and-hurdles> (дата звернення: 16.02.2026).
4. Ісаков Р. І., Герасименко Л.О. Сучасні цифрові технології та інноваційні методи навчання у сфері ментального здоров'я. Сучасні цифрові технології та інноваційні методи навчання медичних кадрів. Матеріали науково-практичної конференції з міжнародною участю. Полтава, 2026. С. 92-93.
5. Онлайн та оффлайн навчання: різниця, плюси та мінуси. URL: <https://dan-it.com.ua/uk/blog/onlajn-ta-offlajn-navchannja-riznicja-pljusi-ta-minusi/> (дата звернення: 12.04.2026).
6. Онлайн-освіта vs традиційна: що обирають студенти у 2025 році. URL: <https://www.educationcenter.cz/ua/blog-ua/onlain-osvita-vs-tradytsiina-shcho-obyraiut-studentn-u-2025-rotsi/> (дата звернення: 17.04.2026).
7. Exploring the Effectiveness of Flashcards for Learning and Retention. URL: <https://simplyputpsych.co.uk/psych-101-1/exploring-the-effectiveness-of-flashcards-for-learning-and-retention> (дата звернення: 22.04.2026).
8. Santhosh V. N., Coutinho D., Ankola A.V., Parimala Y.K., Shankkari S., Ragu K. Effectiveness of spaced repetition learning using a mobile flashcard application among dental students: A randomized controlled trial. J Dent Educ. 2024 Sep;88(9):1267-1276, doi: <https://doi.org/10.1002/jdd.13561>.

9. Best Flashcard Apps for Students: Anki vs. RemNote vs. Quizlet (2025). URL <https://notigo.ai/blog/best-flashcard-apps-students-anki-remnote-quizlet-2025> (дата звернення: 28.04.2026).
10. Best Anki Alternatives 2026 URL: <https://goodoff.co/blog/best-anki-alternatives-2026-flashcard-apps> (дата звернення: 01.05.2026).
11. 7 Best Anki Alternatives: Modern Flashcard Apps Compared. URL: <https://scholarly.so/blog/best-anki-alternatives> (дата звернення: 11.05.2026).
12. Flutter documentation. URL: <https://docs.flutter.dev/> (дата звернення: 14.05.2026).
13. About SQLite. URL: <https://sqlite.org/about.html> (дата звернення: 15.05.2026).
14. Flutter vs Kotlin 2026: Which Framework to Choose for Android Apps. The Droids On Roids. URL: <https://www.thedroidsonroids.com/blog/flutter-vs-kotlin-comparison> (дата звернення: 16.05.2026).
15. Flutter vs React Native in 2025: Complete Performance, Cost & Feature Comparison. URL: <https://www.alimertgulec.com/en/blog/flutter-vs-react-native-2025> (дата звернення: 17.05.2026).
16. React Native Docs. URL: <https://reactnative.dev/docs/getting-started> (дата звернення: 17.05.2026).
17. Flutter vs React Native: App Framework Comparison 2026. URL: <https://www.knguru.de/en/blog/flutter-vs-react-native> (дата звернення: 18.05.2026).
18. Flutter. URL: <https://github.com/flutter/flutter> (дата звернення: 22.05.2026).
19. React-native. URL: <https://github.com/facebook/react-native> (дата звернення: 22.05.2026).
20. Flutter vs Kotlin Multiplatform: The 2026 Guide. Luciq. URL: <https://www.luciq.ai/blog/flutter-vs-kotlin-multiplatform-guide> (дата звернення: 23.05.2026).

21. A detailed comparison of cross platform app development frameworks. Sendbird. URL: <https://sendbird.com/developer/tutorials/choosing-a-cross-platform-app-development-framework-a-detailed-comparison> (дата звернення: 24.05.2026).

22. Provider. URL: <https://pub.dev/packages/provider> (дата звернення: 25.05.2026).

23. Sqflite. URL: <https://pub.dev/packages/sqflite> (дата звернення: 27.05.2026).