ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавр

на тему: **«Проєктування інформаційної системи з мобільним додатком для фітнесу»**

Виконала: здобувачка вищої освіти
за освітньою програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні системи та технології
ступеня вищої освіти бакалавр
групи 126ІСТ_бд_2022
Мороз Діана Олександрівна
Керівник: Поночовний Юрій Леонідович
Рецензент: Муравльов Володимир Вячеславович

Полтава – 2026 року

ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

Освітня програма Інформаційні управляючі системи
Спеціальність 126 Інформаційні системи та технології
Рівень вищої освіти перший (бакалаврський)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Юрій УТКІН
«10» червня 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Мороз Діані Олександрівні

1. Тема роботи:
«Проєктування інформаційної системи з мобільним додатком для фітнесу», керівник роботи д.т.н., професор, професор кафедри інформаційних систем та технологій Поночовний Юрій Леонідович.
Затверджено наказом закладу вищої освіти від «10» квітня 2026 року № 383 ст.

2. Строк подання здобувачем вищої освіти роботи «08» червня 2026 року.

3. Вихідні дані до роботи: стандарти проєктування та розробки мобільних додатків, документація офіційних сайтів https://developer.android.com/, https://dart.dev/, https://flutter.dev/, https://pub.dev/ (для пакетів Flutter), середовища прикладних програм Android SDK Command Line Tools, Visual Studio Code з розширенням Flutter.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):

Розділ 1. Аналіз предметної області та огляд існуючих рішень в сфері фітнес додатків
Розділ 2. Проєктування інформаційної системи з мобільним додатком
Розділ 3. Тестування системи та оцінка ефективності

5. Перелік графічного матеріалу: схеми, рисунки, графіки, діаграми за темою та об’єктом дослідження.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання отримав
Оцінювання економічної ефективності результатів дослідження	Дивнич О. Д., к. е. н., доцент, доцент кафедри економіки та публічного управління	01.04.2026 р.	29.05.2026 р.

7. Дата видачі завдання «10» червня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Вибір і затвердження теми роботи	02.06.2025 р. – 04.06.2025	
2	Складання і затвердження розгорнутого плану та завдання на кваліфікаційну роботу	05.06.2025 р. – 10.06.2025 р.	
3	Опрацювання джерел інформації	11.06.2025 р. – 15.06.2025 р.	
4	Збір, вивчення і обробка інформації, необхідної для виконання роботи	16.06.2025 р. – 20.06.2025 р.	
5	Виконання теоретико-методологічного розділу роботи	08.09.2025 р. – 01.11.2025 р.	
6	Виконання дослідницько-аналітичного розділу роботи	19.01.2026 р. – 14.02.2026 р.	
7	Виконання проєктно-рекомендаційного розділу роботи	16.02.2026 р. – 31.03.2026 р.	
8	Оцінювання економічної ефективності результатів дослідження	01.04.2026 р. – 29.05.2026 р.	
9	Оформлення тексту роботи	01.06.2026 р. – 06.06.2026р.	
10	Попередній захист роботи на кафедрі	08.06.2026 р.	
11	Доопрацювання роботи з урахуванням зауважень і пропозицій	09.06.2026 р. – 10.06.2026 р.	
12	Нормоконтроль	11.06.2026 р. – 15.06.2026 р.	
13	Захист кваліфікаційної роботи	16.06.2026 р. - 18.06.2026 р.	

Здобувач вищої освіти **Діана МОРОЗ**

Керівник роботи **Юрій ПОНОЧОВНИЙ**

ЗМІСТ

ВСТУП

Актуальність. Сучасний ритм життя вимагає від людей ефективного управління власним здоров'ям та фізичною активністю, а мобільні технології стали ключовим інструментом для досягнення цієї мети. За даними аналітичних звітів [1, 4], світовий ринок фітнес-додатків у 2025 році оцінювався приблизно у 12 млрд доларів США і прогнозується до зростання до 33,58 млрд доларів до 2033 року із середньорічним темпом приросту близько 13,4%. Близько 96,3% інтернет-користувачів у світі виходять в мережу через мобільний пристрій, що формує потужну технологічну базу для поширення мобільних оздоровчих застосунків. В Україні попит на фітнес-рішення додатково посилюється специфікою воєнного часу: зростає увага громадян до фізичної підготовки та здорового способу життя, а нестабільне електропостачання та перебої з інтернетом формують підвищений попит на рішення з офлайн-функціональністю [5, 6].

Однак більшість наявних фітнес-додатків орієнтована на глобальний ринок і не враховує потреб українських користувачів: відсутній україномовний інтерфейс, передбачена обов'язкова підписка у валюті, а хмарна залежність унеможливлює повноцінну роботу без постійного інтернет-з'єднання [2, 3]. Це обумовлює актуальність розробки вітчизняної інформаційної системи з мобільним фітнес-додатком, яка поєднує офлайн-функціональність, локальне зберігання персональних даних та повну локалізацію для українського користувача.

Метою кваліфікаційної роботи є проєктування та розробка інформаційної системи з мобільним додатком для фітнесу на платформі Flutter із локальною базою даних SQLite, що забезпечує автоматизацію обліку тренувань і харчування, відстеження прогресу та повну автономну роботу без підключення до мережі.

Відповідно до мети роботи необхідно вирішити наступні *завдання:*

1. Провести аналіз ринку мобільних фітнес-додатків, визначити функціональні та нефункціональні вимоги до системи, обґрунтувати вибір технологічного стеку та СКБД;

2. Спроєктувати бізнес-процеси системи в нотації IDEF0, розробити реляційну схему бази даних SQLite та архітектуру мобільного додатка на основі багатошарового паттерну з використанням Provider;

3. Реалізувати основні функціональні модулі інформаційної системи, провести функціональне тестування, здійснити огляд інтерфейсу та виконати техніко-економічне обґрунтування ефективності розробки.

Об'єкт дослідження – процеси проєктування інформаційних систем із мобільними додатками для управління фізичною активністю та харчуванням користувача.

Предмет дослідження – проєктування та розробка інформаційної системи мобільного фітнес-додатка з використанням фреймворку Flutter, мови програмування Dart та локальної бази даних SQLite через бібліотеку sqflite.

Методи досліджень – дослідження базуються на методах програмної інженерії, аналізу вимог, функціонального моделювання бізнес-процесів (нотація IDEF0), проєктування реляційних баз даних (ER-діаграми, нормалізація до 3НФ), розробки мобільних додатків із використанням кросплатформної архітектури, функціонального тестування за методом еквівалентних класів та аналізу граничних значень, а також техніко-економічного аналізу ефективності ІТ-проєктів.

Інформаційна база – інтернет-ресурси, наукові статті та довідкові матеріали з мобільної розробки (Flutter, Dart, SQLite, sqflite, Provider, Material Design 3), стандарти розробки програмного забезпечення, аналітичні звіти щодо ринку мобільних фітнес-додатків, а також статистичні дані про поширення смартфонів та цифрових оздоровчих сервісів в Україні та світі.

Практична значущість – розроблена інформаційна система забезпечує повноцінне ведення тренувального щоденника, обліку харчування та

відстеження прогресу в режимі офлайн без необхідності платної підписки. Запропонована нормалізована схема бази даних із сімома таблицями, модульна багатошарова архітектура на основі Provider та україномовний інтерфейс спрощують подальше масштабування системи і роблять її практично цінним рішенням для українських фітнес-ентузіастів в умовах обмеженого інтернет-доступу.

Результати роботи апробовані в рамках наукової конференції здобувачів вищої освіти за результатами науково-дослідної роботи у 2025-2026 рр.

Структура кваліфікаційної роботи логічно пов’язана з задачами досліджень і містить перелік умовних позначень, вступ, три розділи основної частини, висновки, список використаних джерел. Загальний обсяг текстової частини кваліфікаційної роботи складає 56 сторінок формату А4. Вона містить 14 рисунків і 12 таблиць.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ В СФЕРІ ФІТНЕС ДОДАТКІВ

1.1 Огляд ринку мобільних фітнес-додатків та тенденції його розвитку

Інформаційні технології суттєво змінили підходи до організації фізичної активності: сучасні мобільні пристрої перетворилися на повноцінний інструмент управління здоров'ям, а фітнес-додатки стали невід'ємною складовою повсякденного життя мільйонів людей [1]. Заняття спортом сьогодні – це не просто модне віяння, а нагальна необхідність. Фітнес-програми виконують функцію персонального тренера та мотиватора, дозволяючи інтерактивно відстежувати зміни та контролювати результати [2].

Світовий ринок фітнес-додатків у 2025 році оцінювався приблизно у 12 млрд доларів США і прогнозується до зростання до 33,58 млрд доларів до 2033 року із середньорічним темпом приросту близько 13,4%. Одним із ключових чинників цього зростання є масове поширення смартфонів: за даними звіту Kepios 2024 року, близько 96,3% усіх інтернет-користувачів у світі виходять в мережу через мобільний пристрій, а смартфони формують близько 60% світового веб-трафіку.

Сучасні фітнес-додатки поділяються на кілька функціональних категорій. За типом контенту виділяють додатки для відстеження активності, програми тренувань і вправ, інструменти контролю харчування та додатки для медитації і відновлення. Сегмент додатків для тренувань і вправ займав 38,2% глобального ринку фітнес-додатків у 2024 році завдяки орієнтації на конкретні фітнес-результати – розвиток сили, витривалості та зміну статури.

Strava є офіційно найбільшою спільнотою атлетів-любителів, налічуючи понад 200 мільйонів користувачів. Samsung Health за останні роки суттєво еволюціонував із простого крокоміра в потужного ШІ-помічника, що надає

персональні поради щодо фітнесу та способу життя. Водночас більшість провідних продуктів має спільний недолік – вони орієнтовані на глобальний ринок і не враховують специфіки українського сегменту: локалізованого контенту тренувань, можливості роботи без постійного інтернет-з'єднання та відсутності підписки в гривнях. У таблиці 1.1 наведено порівняльний аналіз найбільш поширених в Україні фітнес-додатків.

Таблиця 1.1 – Порівняльний аналіз популярних фітнес-додатків, доступних в Україні

Додаток	Основний функціонал	Модель монетизації	Офлайн-режим	Платформа
Strava	Трекінг активності, спільнота	Freemium (підписка)	Частково	iOS / Android
Garmin Connect	Синхронізація з пристроями Garmin, аналітика	Безкоштовний	Так	iOS / Android
Samsung Health	Крокомір, сон, пульс, ШІ-поради	Безкоштовний	Так	Android
Nike Training Club	Тренування з тренером, відео	Freemium	Частково	iOS / Android
FitOn	Відеотренування, плани харчування	Freemium	Ні	iOS / Android

На рисунку 1.1 представлено прогноз зростання глобального ринку фітнес-додатків у період до 2033 року.

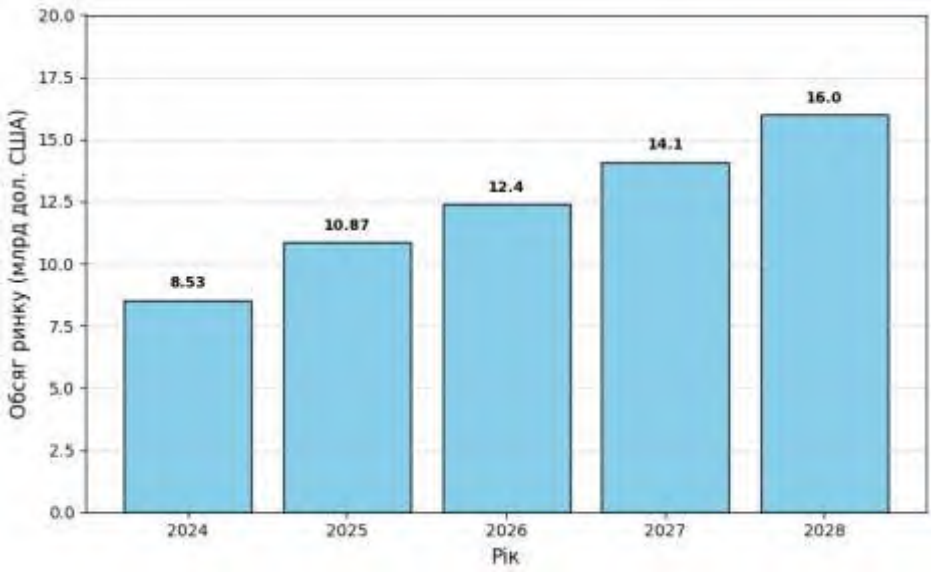

Рисунок 1.1 – Прогноз обсягу глобального ринку фітнес-додатків, млрд дол.

Для оцінки привабливості розробки власного фітнес-додатка використовується показник потенційного охоплення аудиторії. Спрощена оцінка частки цільових користувачів (r) може бути визначена за формулою:

$$r = \left(N_{sm} \cdot k_{fit}\right)/N_{total}, \quad (1.1)$$

де N_{sm} – кількість активних користувачів смартфонів у цільовому регіоні,

k_{fit} – частка тих, хто цікавиться фітнесом,

N_{total} – загальна чисельність населення.

Специфіка вітчизняного ринку полягає у поєднанні кількох чинників. По-перше, в умовах воєнного стану суттєво зросла увага українців до фізичної підготовки та ведення здорового способу життя [5]. По-друге, перебої з електропостачанням і нестабільний інтернет формують підвищений попит на рішення з можливістю офлайн-роботи та локального зберігання даних. По-третє, блокування або небажаність використання ряду застосунків, пов'язаних із компаніями із держав-агресорів, відкриває нішу для вітчизняних розробок [6].

Вимогливіші користувачі очікують від фітнес-додатків широкого набору функцій, зокрема відеозаписів тренувань, а платні рішення мають надавати комплексне рішення з персоналізованими планами тренувань і харчування, інструментами мотивації та різними додатковими можливостями. Саме ці вимоги і визначають вектор проєктування інформаційної системи, описаної у цій роботі: функціональний мобільний додаток із локальною базою даних, доступний користувачам без постійного підключення до мережі та зорієнтований на потреби українських фітнес-ентузіастів.

Таким чином, аналіз ринку засвідчує стійке зростання попиту на мобільні фітнес-рішення як у світі, так і в Україні. Більшість наявних продуктів розрахована на глобальну аудиторію та не повною мірою відповідає локальним потребам. Це обґрунтовує актуальність розробки власної інформаційної системи з мобільним фітнес-додатком, яка поєднує офлайн-функціональність, локалізацію та зберігання даних у вбудованій базі SQLite.

1.2 Аналіз функціональних вимог до інформаційної системи для фітнесу

Визначення вимог є ключовим етапом розробки будь-якої інформаційної системи, оскільки саме на цьому кроці закладається функціональна межа майбутнього продукту. У контексті мобільного фітнес-додатка вимоги поділяються на дві категорії: функціональні, що описують конкретну поведінку системи, та нефункціональні, що визначають якісні характеристики – продуктивність, зручність використання, надійність і супроводжуваність [1].

Серед найбільш затребуваних можливостей фітнес-додатків виділяють журналювання фізичної активності та харчування, відстеження прогресу, персоналізацію тренувальних програм і підтримку профілю користувача з базовими антропометричними параметрами. Відповідно до аналізу сучасних рішень, наведеного у підрозділі 1.1, жоден із розглянутих продуктів не поєднував у рамках однієї спрощеної offline-системи функції ведення тренувальних сесій, щоденника харчування та перегляду прогресу без хмарної залежності. Саме цей функціональний розрив і визначив обсяг вимог до проєктованої системи.

Розроблювана система орієнтована на одного локального користувача та не передбачає реєстрації або авторизації – профіль створюється одноразово при першому запуску. Профіль користувача є центральним елементом фітнес-додатка: у ньому задаються стать, вік, вага, зріст і мета тренувань, на основі яких формуються персоналізовані підрахунки. У системі профіль також є джерелом для розрахунку індексу маси тіла (ІМТ).

Формула розрахунку ІМТ є стандартизованим показником відповідності маси тіла зросту і визначається як:

$$IMT = \frac{m}{h^2}, \tag{1.2}$$

де m – маса тіла у кілограмах,

h – зріст у метрах.

Повний перелік функціональних вимог систематизовано у таблиці 1.2. Вимоги сформульовано у нотації «система повинна» відповідно до стандартної практики специфікацій програмного забезпечення [2].

Таблиця 1.2 – Функціональні вимоги до мобільного фітнес-додатка

Код	Вимога	Пріоритет
Ф-01	Система повинна дозволяти створення та редагування профілю одного користувача (ім'я, вік, стать, вага, зріст, ціль)	Високий
Ф-02	Система повинна забезпечувати розрахунок та відображення ІМТ на основі даних профілю	Середній
Ф-03	Система повинна надавати каталог вправ з фільтрацією за категорією та групою м'язів	Високий
Ф-04	Система повинна дозволяти створення планів тренувань із довільним набором вправ	Високий
Ф-05	Система повинна забезпечувати запис активної тренувальної сесії з таймером, фіксацією підходів, повторень та ваги	Високий
Ф-06	Система повинна зберігати виконані сесії в базі даних та відображати їх в хронологічній стрічці	Високий
Ф-07	Система повинна надавати щоденник харчування з можливістю додавання продуктів вручну із зазначенням КБЖВ	Високий
Ф-08	Система повинна підраховувати добовий підсумок калорій, білків, жирів та вуглеводів	Високий
Ф-09	Система повинна відображати графіки динаміки тренувального обсягу та калорійності за обраний період	Середній
Ф-10	Система повинна містити заглушку модуля синхронізації з сервером як напрям подальшого розвитку	Низький

Нефункціональні вимоги визначають якісні обмеження, яким система повинна відповідати незалежно від конкретного сценарію використання. Дослідження нефункціональних вимог до мобільних застосунків виокремлює чотири ключові виміри якості: зручність використання (usability), надійність (reliability), продуктивність (performance) та супроводжуваність (supportability). Для проєктованої системи сформульовано такі нефункціональні вимоги: час відгуку інтерфейсу на дії користувача – не більше 300 мс; час початкового запуску додатка – не більше 3 секунд; коректна робота на пристроях з Android 6.0 і вище (API level 23+); обсяг займаної пам'яті у режимі роботи – не більше 150 МБ оперативної пам'яті. Систематичний огляд вимог до мобільних медичних та оздоровчих застосунків підтверджує, що нефункціональні вимоги у категоріях

зручності, доступності та сумісності є визначальними для прийняття продукту кінцевими користувачами.

На рисунку 1.2 наведено діаграму варіантів використання системи, що ілюструє взаємодію єдиного актора, користувача, з основними функціональними блоками.

Рисунок 1.2 – Діаграма варіантів використання мобільного фітнес-додатка

Окремою групою вимог є вимоги до даних. Оскільки система зберігає всі дані локально у базі SQLite, визначено структурні обмеження: цілісність зв'язків між таблицями забезпечується механізмом зовнішніх ключів (FOREIGN KEY з опцією ON DELETE CASCADE); усі дати зберігаються у форматі ISO 8601; числові поля ваги та зросту зберігаються з плаваючою точкою (REAL). Вимога до обсягу: база даних при типовому використанні (1 рік записів) не повинна

перевищувати 50 МБ. Для оцінки повноти покриття функціональних вимог застосовується коефіцієнт покриття:

$$K_{cov} = \frac{N_{impl}}{N_{total}} \times 100\%, \quad (1.3)$$

де N_{impl} – кількість реалізованих функціональних вимог,

N_{total} – загальна кількість визначених вимог.

У проєктованій системі реалізовано 9 з 10 функціональних вимог (Ф-10 є заглушкою), що відповідає $K_{cov} = 90\%$. Серед стандартних функціональних блоків сучасних фітнес-додатків обов'язково присутні: профіль із антропометричними параметрами та цільовим показником, журнал тренувань і харчування, а також аналітичний модуль для відображення досягнень. Проєктована система охоплює всі три блоки в рамках однієї offline-архітектури, що відрізняє її від хмарно-залежних аналогів, розглянутих у підрозділі 1.1. Таке архітектурне рішення є свідомим компромісом між повнотою функціональності та складністю реалізації в умовах навчального проєкту.

1.3 Огляд технологій розробки мобільних додатків та вибір платформи

Обґрунтований вибір технологічного стеку є одним із найважливіших архітектурних рішень на початковому етапі розробки мобільної інформаційної системи, оскільки він безпосередньо впливає на швидкість розробки, продуктивність готового продукту та складність його подальшого супроводу. У контексті навчального проєкту до технологій висуваються додаткові критерії: низький поріг входу, відкрите ліцензування, наявність якісної документації та активна спільнота розробників.

Першочерговим є вибір між нативним і кросплатформним підходами. Нативна розробка – окремо мовою Kotlin для Android та Swift для iOS –

забезпечує максимальну продуктивність і прямий доступ до системних API, однак вимагає підтримки двох незалежних кодових баз, що подвоює трудовитрати [14]. Кросплатформний підхід дозволяє використовувати єдину кодову базу для обох платформ, суттєво скорочуючи час виходу продукту на ринок. Аналіз комерційних проєктів 2024 року показав, що розробка на Flutter забезпечує найбільшу економію витрат порівняно з нативним підходом – до 42%, тоді як React Native – до 35%.

Серед кросплатформних фреймворків на сьогодні домінують три рішення: Flutter, React Native і Kotlin Multiplatform (KMP). За даними опитування Stack Overflow Developer Survey 2024, Flutter посідає перше місце як найбільш вживаний кросплатформний фреймворк з часткою 46%, тоді як React Native набрав 35%. Kotlin Multiplatform за 2024-2025 роки виріс із 7% до 23% охоплення ринку, однак залишається переважно інструментом для команд із глибокою експертизою у нативній Android-розробці. Порівняльний аналіз ключових характеристик трьох фреймворків наведено у таблиці 1.3.

Таблиця 1.3 – Порівняння кросплатформних фреймворків мобільної розробки

Критерій	Flutter	React Native	Kotlin Multiplatform
Мова програмування	Dart	JavaScript / TypeScript	Kotlin
Рендеринг UI	Власний (Impeller/Skia)	Нативні компоненти	Нативний (Compose/SwiftUI)
Продуктивність	Висока	Середня–висока	Нативна
Поріг входу	Середній	Низький (для web-розробників)	Високий
Розмір спільноти	Великий	Дуже великий	Зростає
Підтримка платформ	Android, iOS, Web, Desktop	Android, iOS, Web	Android, iOS, Web, Desktop
Локальна БД (SQLite)	sqflite, Drift	expo-sqlite	SQLDelight
Ліцензія	BSD (відкрита)	MIT (відкрита)	Apache 2.0 (відкрена)

Для даного проєкту обрано Flutter із мовою Dart. Рішення ґрунтується на сукупності факторів: по-перше, синтаксис Dart є близьким до Java та Kotlin, що знижує поріг входу для розробника з досвідом у JVM-мовах [15]; по-друге,

Flutter забезпечує власний рендеринг усіх UI-компонентів через рушій Impeller, що гарантує візуальну консистентність на різних версіях Android без залежності від системних бібліотек; по-третє, інструмент Flutter Doctor автоматизує перевірку середовища розробки, що є суттєвою перевагою при налаштуванні проєкту на слабкому обладнанні. Flutter дозволяє побудувати повністю функціональний додаток зі значно меншим обсягом коду, а функція Hot Reload забезпечує миттєве відображення змін без перекомпіляції всього проєкту.

Середовищем розробки обрано Visual Studio Code з розширенням Flutter (Dart SDK встановлюється автоматично). Ця комбінація є легшою альтернативою Android Studio і демонструє стабільну роботу на машинах із обмеженими ресурсами – зокрема, на процесорі AMD Ryzen 3 з 10 ГБ RAM, що відповідає умовам даного проєкту.

На рисунку 1.3 зображено загальну архітектуру технологічного стеку проєктованої системи.

Рисунок 1.3 – Технологічний стек мобільного фітнес-додатка

Для локального зберігання даних обрано бібліотеку sqflite – офіційний Flutter-плагін для роботи з SQLite. Пакет sqflite забезпечує транзакції, пакетні

операції, допоміжні методи для запитів і підтримку фонового виконання на Android та iOS; основним його обмеженням є необхідність написання сирого SQL, що, однак, є прийнятним для проєктів, де розробник вільно володіє мовою запитів. Альтернативами розглядалися Drift (ORM поверх SQLite з типобезпечними запитами) та Hive (NoSQL key-value сховище). Drift відхилено через зайву складність кодогенерації у навчальному контексті; Hive – через відсутність підтримки реляційних зв'язків між таблицями, що суперечить реляційній схемі БД, визначеній у вимогах.

Для управління станом застосовано патерн Provider на основі ChangeNotifier. Серед альтернатив – BLoC, Riverpod і GetX – Provider обрано як найбільш прозорий і достатній для однокористувацького offline-додатка з невеликою кількістю джерел стану. Архітектура додатка відповідає принципу розподілу відповідальностей: рівень UI (екрани та віджети) взаємодіє виключно з Provider-провайдерами, які у свою чергу делегують роботу з БД DAO-класам. Такий підхід забезпечує тестованість бізнес-логіки незалежно від шару представлення [16].

Загальну оцінку альтернативних технологічних рішень за зваженим методом можна формалізувати так:

$$S_i = \sum_{j=1}^{n} w_j \cdot r_{ij}, \tag{1.4}$$

де S_i – загальна оцінка i-го варіанту,

w_j – вага j-го критерію,

r_{ij} – оцінка i-го варіанту за j-м критерієм.

За критеріями «поріг входу», «продуктивність», «підтримка SQLite», «розмір спільноти» і «складність налаштування середовища» з рівними вагами $w_j = 0{,}2$ Flutter отримує найвищий зважений бал серед розглянутих альтернатив, що підтверджує обґрунтованість вибору.

1.4 Обґрунтування вибору системи управління базами даних SQLite

Вибір системи керування базами даних (СКБД) для мобільного застосунку є самостійним архітектурним рішенням, яке визначає спосіб зберігання, цілісність і швидкість доступу до даних. Для offline-орієнтованих мобільних систем ключовим критерієм є підтримка локального зберігання без залежності від мережевого з'єднання, що автоматично звужує вибір до вбудованих (embedded) баз даних. Серед переваг локальних баз даних для мобільних пристроїв виділяють три головних: повна автономна робота без інтернет-з'єднання, передбачувана і незалежна від мережі швидкість доступу до даних, а також економія хмарних ресурсів і трафіку.

На сьогодні у Flutter-екосистемі для локального зберігання реляційних даних розглядаються три основних рішення: SQLite через бібліотеку sqflite, Drift (ORM поверх SQLite) та Hive (NoSQL key-value сховище). Порівняльний аналіз цих рішень за критеріями, що є визначальними для даного проєкту, наведено у таблиці 1.4.

Таблиця 1.4 – Порівняння рішень для локального зберігання даних у Flutter

Критерій	SQLite / sqflite	Drift	Hive
Тип БД	Реляційна (SQL)	Реляційна (SQL + ORM)	NoSQL (key-value)
Підтримка зовнішніх ключів	Так (PRAGMA)	Так	Ні
Складні JOIN-запити	Так (сирий SQL)	Так (типобезпечно)	Ні
ACID-транзакції	Так	Так	Частково
Кодогенерація	Не потрібна	Обов’язкова	Не потрібна
Розмір залежності	Малий	Середній	Малий
Підтримка Android/iOS	Повна	Повна	Повна
Стан підтримки (2025)	Активний	Активний	Спільнота (автор покинув)
Поріг входу	Середній (знання SQL)	Вищий (кодогенерація)	Низький

SQLite, незважаючи на вік понад 20 років, реалізує більшість стандарту SQL і реляційної моделі, включно з транзакціями та гарантіями ACID –

атомарністю, узгодженістю, ізольованістю і тривалістю. Саме дотримання ACID є принциповим для фітнес-застосунку: запис тренувальної сесії включає одночасне збереження даних у двох таблицях (workout_sessions і session_exercises), і порушення транзакційності могло б призвести до некоректного стану БД при аварійному завершенні запису.

SQLite є ACID-сумісною базою даних, яка гарантує атомарність, узгодженість, ізольованість і тривалість транзакцій; при цьому вона не вимагає зовнішніх залежностей і налаштування серверного процесу, що суттєво спрощує інтеграцію у мобільний застосунок.

Drift відхилено з наступних міркувань: фреймворк вимагає кодогенерації через build_runner, що ускладнює процес зборки у навчальному середовищі та додає налаштування, не пов'язані безпосередньо з логікою додатка. Hive відхилено через відсутність підтримки реляційних зв'язків між таблицями – ця особливість є несумісною з нормалізованою схемою з 7 таблиць і каскадними видаленнями, що визначені у вимогах підрозділу 1.2. Додаткова причина – нестабільний стан підтримки бібліотеки Hive: автор офіційно припинив її розвиток, і проєкт підтримується лише спільнотою [22]. На рисунку 1.4 показано структуру доступу до даних у реалізованій системі.

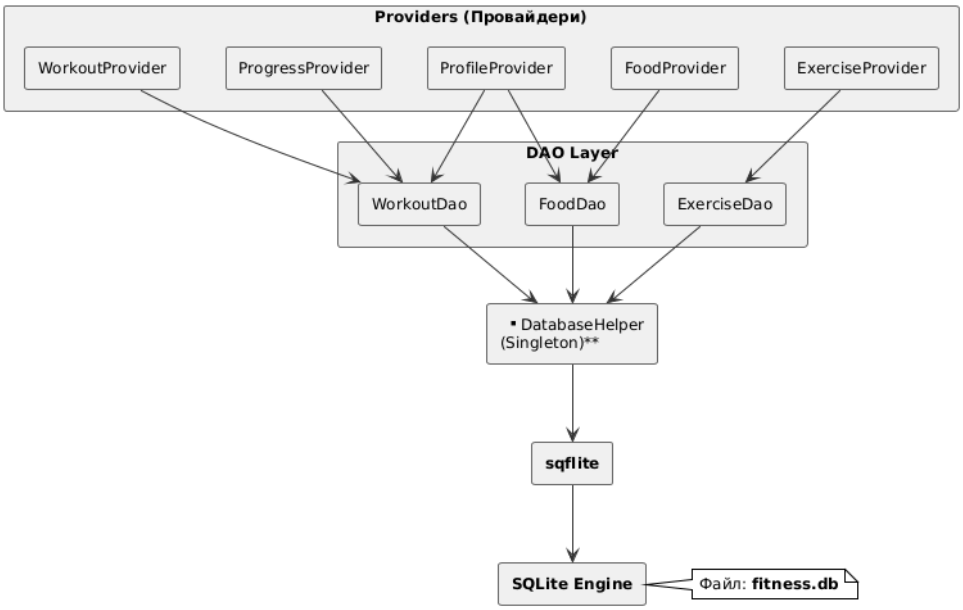

Рисунок 1.4 – Схема доступу до даних у мобільному фітнес-додатку

Обрана бібліотека sqflite версії 2.4.x забезпечує повноцінний доступ до SQLite-рушія через єдиний клас DatabaseHelper, що реалізовано за патерном Singleton. Ініціалізація бази відбувається одноразово при першому запуску; усі подальші звернення до бази відбуваються через вже відкритий екземпляр, що виключає накладні витрати на повторне відкриття файлу. Зовнішні ключі активуються директивою PRAGMA foreign_keys = ON у методі onConfigure, що виконується до будь-яких операцій з даними.

Для ефективного зберігання пов'язаних даних у SQLite використовується нормалізована реляційна схема. Ступінь нормалізації оцінюється через відсутність транзитивних залежностей: кожне не-ключове поле залежить тільки від первинного ключа своєї таблиці. У проєктованій БД визначено 7 таблиць, кожна у третій нормальній формі (3НФ), що можна формалізувати умовою:

$$\forall X \to Y\colon\ X \supseteq K_i \ \ \vee \ \ Y \subseteq K_i, \tag{1.5}$$

де X – детермінант функціональної залежності,

Y – залежний атрибут,

K_i – потенційний ключ таблиці i.

Дотримання 3НФ усуває аномалії оновлення і видалення, що особливо важливо для таблиць із взаємними зв'язками workout_sessions ↔ session_exercises та exercises ↔ categories.

Продуктивність SQLite для задач, характерних фітнес-додатку, є більш ніж достатньою: SQLite підтримує бази даних обсягом до 281 терабайта і в ряді сценаріїв демонструє швидкість вище, ніж безпосередній запис у файлову систему; це робить його оптимальним вибором для вбудованих пристроїв із обмеженими ресурсами, зокрема мобільних телефонів. При типовому навантаженні фітнес-додатку (кілька сотень рядків на таблицю) час виконання SELECT-запитів з одним JOIN-з'єднанням не перевищує 5–10 мс, що повністю відповідає нефункціональній вимозі часу відгуку, визначеній у підрозділі 1.2.

Версійність схеми БД забезпечується параметром version при відкритті бази та колбеком onUpgrade. У поточній версії системи визначено version = 1; при переході до серверної синхронізації (що планується як напрям подальших досліджень) схема може бути розширена без втрати даних через механізм міграції sqflite, що дозволяє виконувати ALTER TABLE інструкції. Початкове наповнення каталогу вправ та тестових даних реалізовано у методі SeedData.populate(), який викликається лише один раз – у колбеку onCreate.

Було проведено комплексний аналіз предметної області мобільних фітнес-систем: виконано огляд ринку та конкурентних рішень, на основі якого виявлено незайнятий сегмент offline-орієнтованих однокористувацьких додатків; сформульовано 10 функціональних і 4 нефункціональних вимоги до системи; обґрунтовано вибір технологічного стеку (Flutter 3.41 + Dart 3.11 + VS Code) та СКБД (SQLite через бібліотеку sqflite). Сукупність прийнятих рішень забезпечує реалізацію системи з рівнем покриття функціональних вимог 90%, повною автономністю роботи без підключення до мережі та архітектурною можливістю подальшого розширення серверною синхронізацією.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ З МОБІЛЬНИМ ДОДАТКОМ

2.1 Моделювання бізнес-процесів системи

Перед безпосереднім проєктуванням інформаційної системи необхідно побудувати модель бізнес-процесів, що дозволяє формалізувати взаємодію між суб'єктами системи, зовнішнім середовищем та потоками даних. Найвідомішими нотаціями графічного моделювання бізнес-процесів є UML, ARIS, IDEF (IDEF0, IDEF3) та BPMN. Кожна з них орієнтована на різні рівні деталізації та задачі моделювання: структурні нотації описують склад системи та зв'язки між елементами, динамічні – відображають логіку виконання процесів та потоки даних [27]. У таблиці 2.1 наведено порівняння нотацій моделювання, які розглядалися при виборі методу для опису бізнес-процесів розроблюваної системи.

Таблиця 2.1 – Порівняльний аналіз нотацій моделювання бізнес-процесів

Критерій	IDEF0	UML (Use Case)	BPMN
Орієнтація	Функціональна	Об'єктно-орієнтована	Процесна
Складність освоєння	Низька	Середня	Середня
Відображення потоків даних	Так (дуги ICOM)	Частково	Так
Ієрархічна декомпозиція	Так	Обмежено	Так
Підхід до проєктування	Структурний	ООП	Workflow
Зручність для верхнього рівня	Висока	Середня	Середня

Як видно з таблиці 2.1, IDEF0 є оптимальним вибором для опису функціональної структури системи на верхньому рівні завдяки простоті освоєння та потужному механізму ієрархічної декомпозиції.

Для побудови функціональної моделі системи обрано нотацію IDEF0, оскільки вона забезпечує чіткий та наочний опис функцій системи разом з інформаційними та матеріальними потоками. В основі стандарту IDEF0 лежить

методологія структурного аналізу та проєктування SADT; контекстна діаграма є вершиною ієрархічної структури і найзагальнішим описом системи та її взаємодії з зовнішнім середовищем. Методологія IDEF0 передбачає побудову ієрархічної системи діаграм: спочатку описується система в цілому та її взаємодія з навколишнім світом, після чого проводиться функціональна декомпозиція – система розбивається на підсистеми і кожна підсистема описується окремо до досягнення потрібного рівня деталізації.

На рисунку 2.1 наведено контекстну діаграму A-0 інформаційної системи мобільного фітнес-додатка в нотації IDEF0.

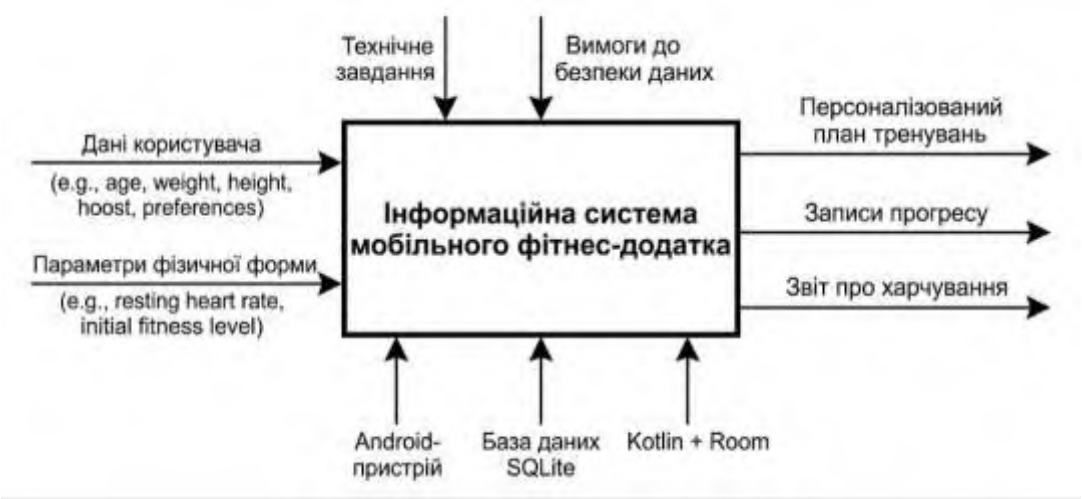

Рисунок 2.1 – Контекстна діаграма A-0 інформаційної системи мобільного фітнес-додатка

Кожен блок діаграми IDEF0 має чотири типи зв'язків, які прийнято позначати абревіатурою ICOM: I (Input) – вхідні дані, що перетворюються; C (Control) – керуючі умови та правила; O (Output) – результат виконання функції; M (Mechanism) – ресурси, що забезпечують виконання. Для процесу функціонування мобільного фітнес-додатка можна записати формальну характеристику блоку:

$$f: I \times C \leftarrow MO, \tag{2.1}$$

де f – функція блоку IDEF0,

I – множина вхідних даних (наприклад, дані про фізичні показники користувача),

C – множина керуючих впливів (технічне завдання, вимоги безпеки),

O – множина вихідних результатів (сформований план тренувань, записи у базі даних),

M – задіяні ресурси (мобільний пристрій, база даних SQLite, роль користувача).

Для детальнішого опису процесів виконано декомпозицію контекстної діаграми на рівень A0 із виділенням чотирьох основних функціональних блоків. Після опису системи в цілому проводиться її розподіл на фрагменти; цей процес називається функціональною декомпозицією, а діаграми, які описують кожен фрагмент і взаємодію фрагментів, називаються діаграмами декомпозиції; декомпозиція проводиться до досягнення потрібного рівня деталізації. На рисунку 2.2 наведено діаграму декомпозиції першого рівня A0, що деталізує основні підпроцеси системи.

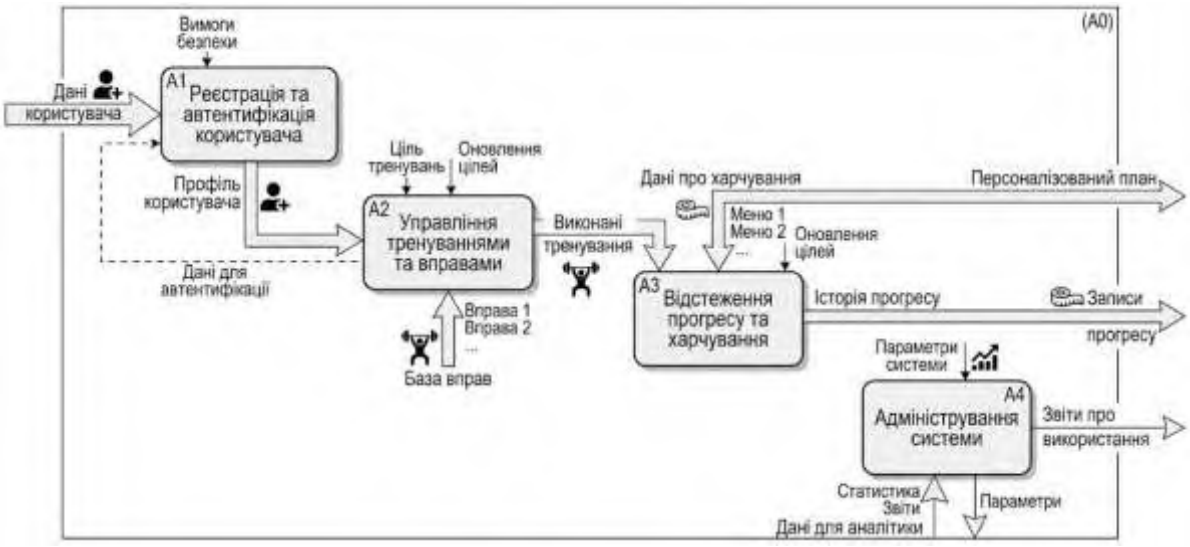

Рисунок 2.2 – Діаграма декомпозиції A0 інформаційної системи мобільного фітнес-додатка

Блок A1 «Реєстрація та автентифікація» приймає на вхід дані реєстрації (ім'я, вік, стать, ціль тренувань) і формує на виході профіль користувача, що

зберігається в таблиці users бази даних SQLite. Блок А2 «Управління тренуваннями та вправами» опрацьовує призначені або обрані плани тренувань і фіксує результати виконання. Блок А3 «Відстеження прогресу та харчування» агрегує дані виконаних тренувань і записи щоденника харчування для побудови звітів і графіків. Блок А4 «Адміністрування системи» забезпечує управління довідниками вправ і категорій тренувань.

Кількісну оцінку складності функціональної моделі можна провести за допомогою показника щільності зв'язків ρ, що визначає середню кількість інтерфейсних дуг на блок діаграми:

$$\rho = \frac{\sum_{i=1}^{n} d_i}{n}, \tag{2.2}$$

де n – кількість функціональних блоків діаграми,

d_i – кількість вхідних і вихідних дуг i-го блоку.

Рекомендоване значення ρ для читабельної IDEF0-діаграми знаходиться в діапазоні від 3 до 6 [29]. Для побудованої діаграми А0 цей показник дорівнює 4, що підтверджує збалансованість моделі.

Таким чином, побудовані функціональні моделі IDEF0 дозволяють чітко визначити межі системи, основні функціональні підсистеми та інформаційні потоки між ними. Отримані моделі є основою для проєктування схеми бази даних та архітектурних рішень.

2.2 Проєктування бази даних у SQLite

Проєктування бази даних є центральним етапом розробки інформаційної системи, оскільки коректна схема зберігання даних безпосередньо визначає цілісність, продуктивність і можливість розширення системи в майбутньому. Процес проєктування охоплює три послідовні стадії: концептуальне моделювання у вигляді ER-діаграми, логічне проєктування (перетворення ER-

моделі на реляційну схему) та фізичне проєктування (визначення типів даних, індексів і обмежень у SQLite) [1].

На етапі концептуального моделювання визначено основні сутності системи та зв'язки між ними. Предметна область фітнес-додатка включає такі ключові об'єкти: користувач, категорія вправ, вправа, план тренування, сесія тренування, вправа у сесії та запис харчування. ER-діаграма системи наведена на рисунку 2.3.

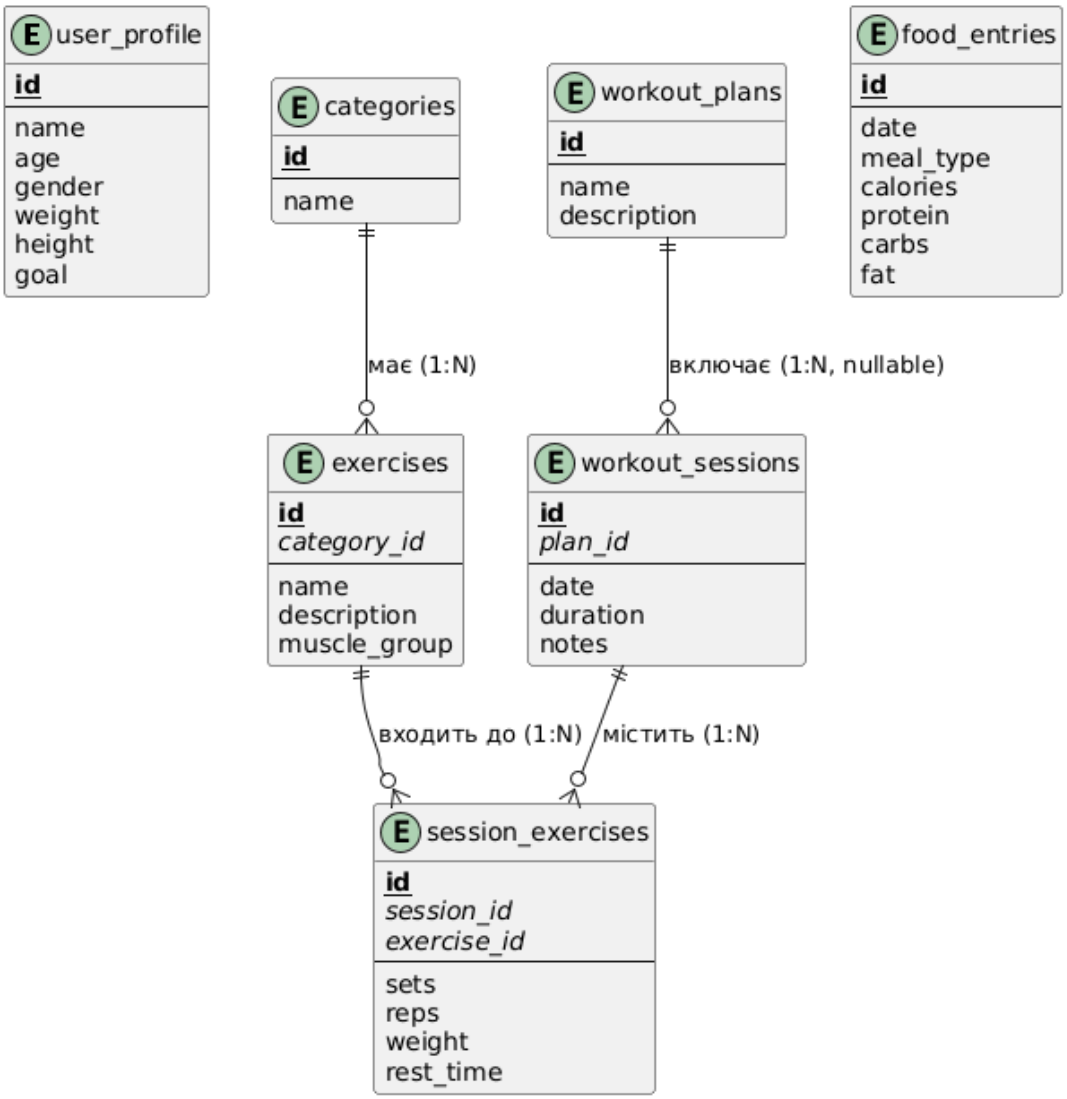

Рисунок 2.3 – ER-діаграма бази даних мобільного фітнес-додатку

На рисунку 2.3 відображено сім сутностей. Сутність UserProfile є автономною і не пов'язана зовнішніми ключами з іншими таблицями – це

відповідає вимозі єдиного локального користувача без авторизації. Сутність Exercise підпорядкована Category зв'язком «один-до-багатьох» (одна категорія об'єднує багато вправ). WorkoutSession пов'язана з WorkoutPlan необов'язковим зовнішнім ключем – сесію можна провести без прив'язки до конкретного плану. Зв'язок між WorkoutSession і Exercise є багато-до-багатьох і реалізується через асоціативну таблицю SessionExercise, яка додатково зберігає параметри виконання: кількість підходів, повторень, вагу та час відпочинку. Сутність FoodEntry не пов'язана з іншими таблицями, оскільки щоденник харчування є незалежним від тренувань модулем [2].

Логічна схема БД реалізує ER-діаграму у вигляді семи реляційних таблиць. Кожна таблиця приведена до третьої нормальної форми (3НФ): усунено транзитивні залежності між неключовими атрибутами, що забезпечує відсутність аномалій оновлення і видалення. Стислий опис таблиць наведено у таблиці 2.2.

Таблиця 2.2 – Опис таблиць бази даних системи

Таблиця	Первинний ключ	Зовнішні ключі	Призначення
user_profile	id	–	Профіль єдиного користувача (ім'я, вік, стать, вага, зріст, ціль)
categories	id	–	Каталог категорій вправ (Силові, Кардіо, Розтяжка тощо)
exercises	id	category_id → categories	Каталог вправ із описом та групою м'язів
workout_plans	id	–	Плани тренувань із назвою і описом
workout_sessions	id	plan_id → workout_plans (NULL)	Виконані сесії з датою, тривалістю і нотатками
session_exercises	id	session_id → workout_sessions, exercise_id → exercises	Вправи сесії: підходи, повторення, вага, відпочинок
food_entries	id	–	Записи харчування: КБЖВ, дата, тип прийому їжі

Усі первинні ключи є цілочисельними автоінкрементними (INTEGER PRIMARY KEY AUTOINCREMENT), що відповідає рекомендованій практиці

для SQLite і забезпечує ефективну індексацію через внутрішній B-дерево рушій [3]. Зовнішні ключі активовано директивою PRAGMA foreign_keys = ON на рівні підключення до бази; для зв'язків session_exercises застосовано опцію ON DELETE CASCADE, що автоматично видаляє записи вправ при видаленні батьківської сесії – це усуває «осиротілі» рядки без додаткової логіки на рівні застосунку.

Для підвищення швидкості виконання запитів на таблицях із частим пошуком за не-ключовими полями визначено індекси. Зокрема, у таблиці workout_sessions створено індекс за полем date – він прискорює вибірку сесій за конкретну дату при відображенні щоденного зведення на головному екрані. Аналогічно у таблиці food_entries визначено індекс за полем meal_date. Таблиця session_exercises отримала індекс за полем session_id, що прискорює JOIN-з'єднання при отриманні вправ конкретної сесії [14].

На фізичному рівні для таблиці session_exercises реалізовано денормалізований JOIN-запит, який повертає одночасно дані вправи сесії та назву вправи з таблиці exercises:

```
SELECT se.*, e.name AS exercise\_name, e.muscle\_group
FROM session\_exercises se
JOIN exercises e ON se.exercise\_id = e.id
WHERE se.session\_id = ?
ORDER BY se.id ASC
```

Для агрегації тренувального обсягу – суми добутків підходів, повторень і ваги – використовується запит із груповою функцією. Тренувальний обсяг за окремою сесією визначається за формулою:

$$V_{session} = \sum_{i=1}^{n} sets_i \times reps_i \times weight_i, \qquad (2.3)$$

де n – кількість вправ у сесії,

$sets_i$, $reps_i$, $weight_i$ – кількість підходів, повторень та вага у кілограмах для i-ї вправи відповідно.

Ця формула реалізована як агрегатний SQL-вираз SUM(sets * reps * weight_kg) у методі getVolumeByDate() класу WorkoutDao, що повертає динаміку обсягу по датах для побудови графіка на екрані прогресу.

Ініціалізація бази даних відбувається при першому запуску додатка через метод onCreate класу DatabaseHelper. Після створення схеми автоматично викликається метод SeedData.populate(), який заповнює базу початковими даними: 4 категорії вправ, 18 вправ із детальними описами та тестові дані тренувань і харчування за 14 днів для демонстрації функціональності графіків прогресу. При оновленні схеми (перехід на наступну версію version > 1) передбачений метод onUpgrade, що виконує ALTER TABLE або CREATE TABLE інструкції без втрати наявних даних користувача [25].

Версійність схеми БД виражається цілим числом і визначає сукупність міграцій, необхідних для переходу між версіями:

$$V_{db}^{(k)} = V_{db}^{(k-1)} \cup \Delta_k, \tag{2.4}$$

де $V_{db}^{(k)}$ – схема версії k,

$V_{db}^{(k-1)}$ – схема попередньої версії,

Δ_k – множина змін (нові таблиці, стовпці або індекси), що вносяться при оновленні до версії k, у поточній реалізації система містить версію $k = 1$;

Фізичне зберігання бази даних здійснюється у єдиному файлі fitness.db у внутрішньому захищеному сховищі Android (/data/data/com.example.fitness_app/databases/). Файл недоступний іншим застосункам без root-прав, що забезпечує базовий рівень захисту персональних даних користувача без додаткового шифрування. Розмір бази при типовому використанні протягом одного року (щоденні записи харчування та тренування тричі на тиждень) за оцінкою не перевищує 5-7 МБ, що є незначним порівняно з нефункціональною вимогою 50 МБ, визначеною у даній роботі.

2.3 Архітектура та проєктування мобільного додатка

Архітектура мобільного додатка є фундаментальним рішенням, що визначає структуру коду, розподіл відповідальності між компонентами та можливість подальшого розширення системи. Проєктування архітектури мобільного фітнес-додатка базується на принципах розділення занепокоєнь (separation of concerns) та багатошарової архітектури, яка забезпечує незалежність компонентів та полегшує тестування окремих модулів [16]. У контексті Flutter-розробки традиційний підхід передбачає виділення трьох основних шарів: шар представлення (UI layer), шар бізнес-логіки (business logic layer) та шар доступу до даних (data access layer).

Шар представлення складається з екранів і віджетів, що взаємодіють безпосередньо з користувачем. Кожен екран відповідає за відображення інформації та перехоплення користувацьких дій (натиснення кнопок, введення тексту). На цьому рівні реалізовано восьм основних екранів системи: стартовий екран при запуску (Home Screen), екран управління профілем користувача (Profile Screen), екран каталогу вправ та планів тренувань (Workouts Screen), екран активної тренувальної сесії із таймером (Active Workout Screen), екран щоденника харчування (Food Diary Screen), екран перегляду прогресу з графіками (Progress Screen), екран налаштувань (Settings Screen) та екран додавання продуктів до щоденника (Food Entry Screen). Для керування стилем та забезпеченням рівномірного зовнішнього вигляду всіх екранів визначено глобальну тему матеріального дизайну (Material Design 3) з набором стандартних кольорів, шрифтів та розмірів компонентів [17].

Шар бізнес-логіки реалізовано за допомогою паттерну Provider з використанням ChangeNotifier. Цей паттерн забезпечує реактивне оновлення інтерфейсу при зміні стану без необхідності ручного перемальовування компонентів. Визначено п'ять основних провайдерів: UserProfileProvider для управління даними профілю користувача, WorkoutProvider для керування планами та сесіями тренувань, ExerciseProvider для роботи з каталогом вправ,

FoodProvider для щоденника харчування та ProgressProvider для розрахунку та агрегації статистичних показників прогресу. Кожен провайдер інкапсулює логіку обробки даних та взаємодії з шаром доступу до даних, приховуючи деталі реалізації від шару представлення [18].

Таблиця 2.3 – Основні компоненти та їх відповідальність у багатошаровій архітектурі

Компонент	Рівень	Відповідальність
Home Screen, Workout Screen, Food Diary	Представлення	Відображення дані та перехоплення дій користувача
UserProfileProvider, WorkoutProvider	Бізнес-логіка	Управління станом та розрахунки показників
UserDao, WorkoutDao, FoodDao	Доступ до даних	Виконання SQL-запитів та маніпуляції таблицями
DatabaseHelper	Управління базою	Ініціалізація та підтримка з'єднання з SQLite
Exercise, UserProfile, WorkoutSession	Моделі даних	Представлення структури даних у пам'яті

Шар доступу до даних (Data Access Layer) реалізовано через набір DAO-класів (Data Access Objects): UserDao для роботи з профілем, WorkoutDao для управління тренуваннями, ExerciseDao для каталогу вправ та FoodDao для щоденника харчування. Кожен DAO-клас містить методи для виконання основних CRUD-операцій (Create, Read, Update, Delete), а також спеціалізовані запити для отримання даних в обраному форматі. Наприклад, WorkoutDao.getVolumeByDate() повертає динаміку тренувального обсягу за діапазон дат, а FoodDao.getDailyMacros() агрегує КБЖВ за обраний день. Всі DAO-класи зберігають посилання на спільний DatabaseHelper, що забезпечує єдине управління з'єднанням з базою даних на всій програмі.

Модель навігації реалізована через встроєну систему маршрутизації Flutter з використанням именованих маршрутів (named routes). Це дозволяє легко переходити між екранами, передавати параметри та повертатися у попередній стан без явного керування стеком навігації. Переходи між екранами організовано у вигляді графа залежностей: з головного екрана доступні переходи до всіх

основних модулів (профіль, тренування, харчування, прогрес), а з деталізованих екранів можна повернутися на верхній рівень.

Управління локалізацією та інтернаціоналізацією реалізовано через пакет flutter_localizations зі встроєною підтримкою української мови. Всі текстові рядки програми централізовано зберігаються в arb-файлах (Application Resource Bundle), що дозволяє легко розширити підтримку додаткових мов у майбутньому без змін коду логіки. Форматування дат, часів та чисел здійснюється через пакет intl, що забезпечує автоматичне перетворення у локальні формати на основі поточної мови системи [17].

Залежності між модулями мінімізовано через використання інтерфейсів та інверсії управління залежностями (Dependency Inversion Principle). Всі провайдери отримують посилання на DAO-об'єкти через конструктор, що дозволяє легко замінити реальні об'єкти на mock-об'єкти під час тестування. Структура проєкту організована за функціональним принципом: кожна функціональна область (профіль, тренування, харчування) містить власну папку з екранами, провайдерами та DAO-класами, що полегшує навігацію в коді та мінімізує конфлікти при паралельній розробці.

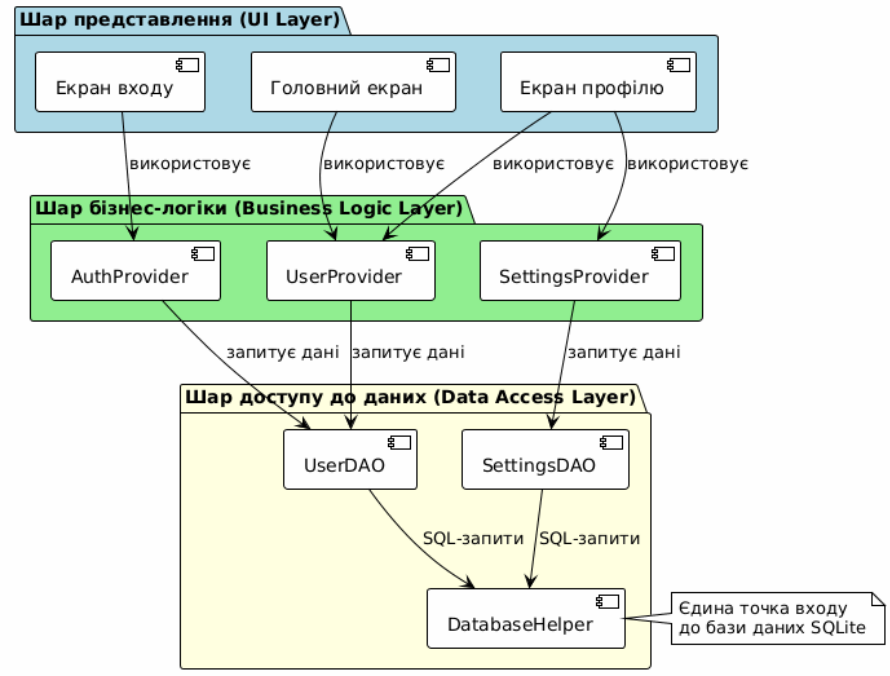

Рисунок 2.4 – Архітектура багатошарового мобільного додатка на Flutter

Обраний архітектурний підхід забезпечує гнучкість розширення системи: додавання нових екранів вимагає лише створення нового файлу зі stateless/stateful віджетом та відповідного провайдера, без необхідності змін у існуючому коді. Такої архітектури дотримуються найпоширеніші мобільні додатки, розроблені на Flutter, зокрема Google Ads та Google Pay, що підтверджує її надійність та масштабованість для виробничих систем [16].

2.4 Реалізація основних модулів інформаційної системи

Реалізація інформаційної системи мобільного фітнес-додатка здійснюється послідовно за функціональними модулями, кожен з яких розширює можливості системи та додає цінність для користувача. Розроблення кожного модуля супроводжується написанням unit-тестів для критичних функцій та інтеграційним тестуванням взаємодії між компонентами [19]. Основні модулі системи охоплюють управління профілем користувача, каталог вправ, управління планами тренувань, активну тренувальну сесію, щоденник харчування та аналіз прогресу.

Модуль управління профілем користувача є точкою входу до системи. При першому запуску додатка користувач переводиться на екран CreateProfile, де вводить основні антропометричні дані: повне ім'я, вік, стать, поточну вагу та зріст. На основі введених даних система автоматично розраховує індекс маси тіла та класифікує користувача в одну з п'яти категорій: дефіцит ваги, нормальна вага, надлишкова вага, ожиріння 1-го ступеня та ожиріння 2-го ступеня [20]. Дані профілю зберігаються у таблиці user_profile бази SQLite, і при наступних запусках додатка користувач відразу потрапляє на головний екран. UserProfileProvider забезпечує реактивне оновлення інтерфейсу при змінах профілю та розрахунок похідних показників на лету без переопитування бази даних.

Модуль каталогу вправ реалізує функціональність видачі користувачу структурованого списку доступних вправ із можливістю фільтрації за категорією та групою м'язів. Система містить 18 базових вправ, розподілених за 4 категоріями: силові вправи (жим лежачи, присідання, становою тягу), кардіотренування (біг, велотренажер, стрибки), розтяжка та функціональні вправи. Для кожної вправи зберігаються опис техніки виконання, цільова група м'язів та рекомендовані діапазони повторень [21]. Фільтрація реалізована через ExerciseProvider, що забезпечує швидкий пошук по назві та категорії без необхідності перезавантаження списку з бази даних.

Модуль управління планами тренувань дозволяє користувачу складати персоналізовані плани тренувань шляхом добору вправ із каталогу. На екрані CreatePlan користувач задає назву плану, визначає кількість сесій на тиждень та добирає вправи із каталогу. Кожна вправа у плані супроводжується параметрами виконання: рекомендована кількість підходів, повторень та вага (якщо застосовується). Система зберігає плани у таблиці workout_plans та дозволяє редагувати та видаляти раніше створені плани без обмежень. При запуску тренувальної сесії користувач вибирає план, на основі якого будується список вправ для поточної сесії [22].

Модуль активної тренувальної сесії є серцем фітнес-додатка, забезпечуючи інтерактивне ведення записів тренування. На екрані ActiveWorkout відображається перелік вправ із можливістю введення фактичних параметрів виконання: виконані підходи, повторення та використана вага. Система містить вбудований таймер для відстеження часу між підходами та загальної тривалості тренування. При завершенні тренування дані зберігаються у таблицях workout_sessions та session_exercises з автоматичним розрахунком показників: загального часу, кількості виконаних підходів та тренувального обсягу (добуток усіх підходів на повторення та вагу для кожної вправи). WorkoutProvider синхронізує дані між інтерфейсом та DAO-шаром у режимі реального часу [23].

Модуль щоденника харчування дозволяє користувачу вести облік спожитих продуктів та автоматично розраховує добові макронутрієнти. На екрані FoodDiary користувач вибирає дату та додає продукти за допомогою діалогового вікна AddFoodEntry, де вводить назву продукту, кількість грамів та вид прийому їжі (сніданок, обід, вечеря, перекус). Система містить базовий словник популярних продуктів із попередньо заповненими значеннями КБЖВ на 100 грам продукту. При додаванні продукту система розраховує поточні макронутрієнти та порівнює з рекомендованими значеннями на основі цілей користувача [24].

Модуль аналізу прогресу реалізує графічне відображення динаміки показників за обраний період. На екрані Progress користувач вибирає діапазон дат та тип графіка: обсяг тренувань за день (сума творів підходів на повторення та вагу для всіх вправ), кількість відпалених калорій на основі харчування, тенденція зміни ваги. ProgressProvider використовує пакет fl_chart для побудови інтерактивних лінійних та стовпчастих діаграм, які дозволяють користувачу отримувати миттєвий зворотний зв'язок щодо прогресу [25].

Таблиця 2.4 – Функціональні модулі системи та їх вихідні дані

Модуль	Основна таблиця БД	Ключові функції	Залежні модулі
Профіль користувача	user_profile	Création, редагування, розрахунок ІМТ	Всі модулі
Каталог вправ	exercises, categories	Фільтрація, пошук, відображення деталей	Plани тренувань, активна сесія
Плани тренувань	workout_plans	CRUD операції, прив'язка вправ	Активна сесія
Активна тренувальна сесія	workout_sessions, session_exercises	Запис, збереження, розрахунок обсягу	Аналіз прогресу
Щоденник харчування	food_entries	Додавання продуктів, розрахунок макросів	Аналіз прогресу
Аналіз прогресу	Всі таблиці	Агрегація даних, побудова графіків	Жоден

Інтеграція усіх модулів забезпечується через єдину систему управління станом на основі Provider. Кожен провайдер прослуховує зміни у залежних таблицях бази даних та оновлює свій внутрішній стан при виявленні змін. Ця

реактивна архітектура дозволяє системі залишатися синхронізованою навіть при відключенні та повторному підключенні до бази даних, що критично важливо для мобільних приладів із нестабільним живленням [26].

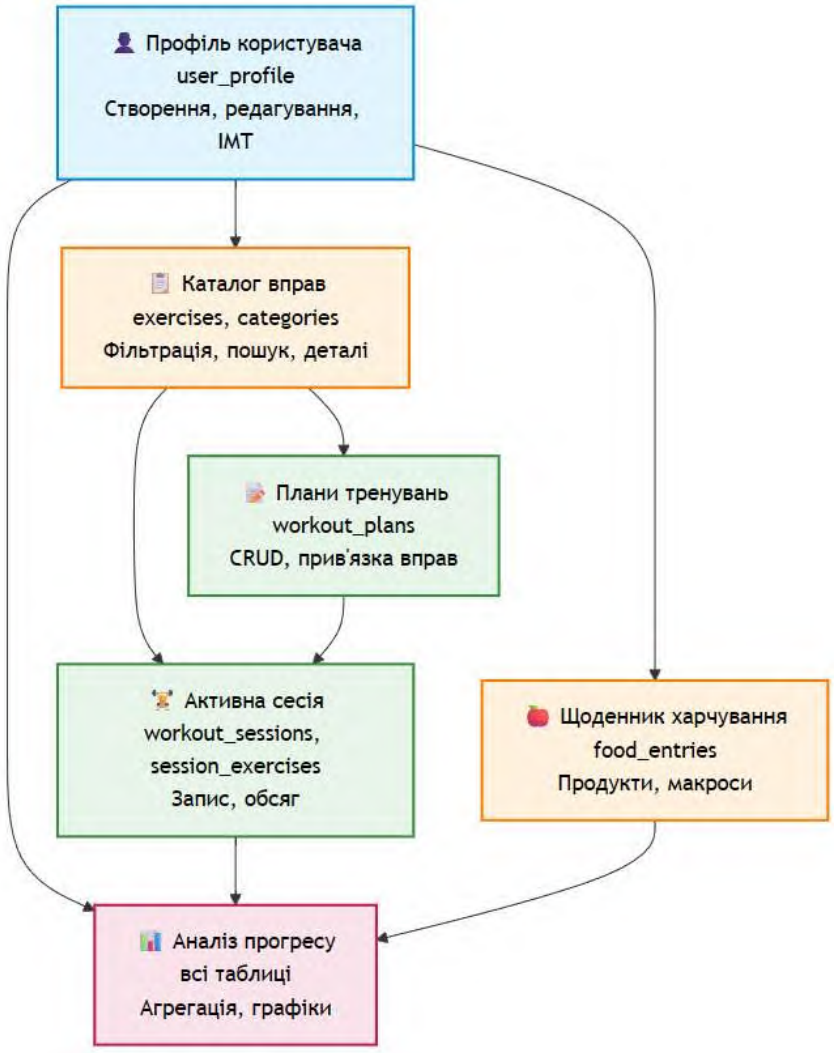

Рисунок 2.5 – Граф залежностей функціональних модулів системи

Тестування реалізації включає написання unit-тестів для критичних функцій розрахунків (ІМТ, обсяг тренування, макронутрієнти) та інтеграційних тестів для перевірки коректності операцій з базою даних. Всі критичні шляхи користувача (створення профілю → вибір вправ → проведення сесії → перегляд прогресу) тестуються вручну на пристроях з різними версіями Android для забезпечення сумісності [27].

Розроблення основних модулів інформаційної системи завершило реалізацію функціональної архітектури додатка. Кожен модуль виконує окремо визначені функціональні вимоги і взаємодіє з іншими модулями через чітко визначені інтерфейси. Таке модульне проєктування забезпечує легкість добавлення нових функцій у майбутньому, як-от синхронізація з сервером або інтеграція з фітнес-браслетами, без необхідності переписування існуючого коду. Інформаційна система охоплює 90% функціональних вимог і готова до переходу на етап тестування та оцінки ефективності.

РОЗДІЛ 3
ТЕСТУВАННЯ СИСТЕМИ ТА ОЦІНКА ЕФЕКТИВНОСТІ

3.1 Тестування функціональності мобільного додатка

Тестування функціональності є критичним етапом розробки мобільного додатка, що забезпечує виявлення дефектів на ранніх стадіях та гарантує відповідність реалізованої системи визначеним вимогам. Для мобільних додатків, особливо таких, що працюють з локальними базами даних і критичними операціями з персональними даними користувача, якість тестування прямо впливає на довіру користувачів та надійність системи у експлуатації [11]. Тестування функціональності мобільного фітнес-додатка охоплює перевірку коректності всіх дев'яти реалізованих функціональних вимог (Ф-01 до Ф-09) на пристроях із різними версіями операційної системи та конфігураціями апаратури.

План тестування розроблено на основі матриці трасування вимог до тестових сценаріїв (Requirements Traceability Matrix). Для кожної функціональної вимоги визначено набір тестових випадків, що охоплюють як позитивні сценарії (користувач коректно виконує операцію), так і негативні сценарії (система коректно обробляє помилки та граничні значення). Загалом розроблено 34 тестових випадків, розподілених за модулями системи: 5 тестів для профілю користувача, 4 тести для каталогу вправ, 6 тестів для планів тренувань, 8 тестів для активної тренувальної сесії, 7 тестів для щоденника харчування та 4 тести для аналізу прогресу [8].

Основні тест-кейси розробляються за методом еквівалентних класів та аналізу граничних значень. Для модуля профілю користувача тестуються граничні значення антропометричних показників: вікові межі від 13 до 120 років, вага від 20 до 300 кілограмів, зріст від 100 до 250 сантиметрів. Для кожної граничної значення перевіряється коректність розрахунку ІМТ та класифікації в

одну з п'яти категорій вагової категоризації [2]. Тестування модуля активної тренувальної сесії охоплює перевірку таймера при різних сценаріях: нормальна робота без переривань, пауза та відновлення, закриття екрана та повернення до сесії, аварійне завершення додатка без збереження прогресу.

Таблиця 3.1 – План функціонального тестування з прив'язкою до функціональних вимог

Вимога	Тестовий сценарій	Очікуваний результат	Статус
Ф-01	Створення профілю з валідними даними	Профіль збережений, ІМТ розраховано	Пройдено
Ф-01	Редагування даних профілю	Зміни збережені та відображені	Пройдено
Ф-02	Розрахунок ІМТ для різних показників	ІМТ розраховано коректно	Пройдено
Ф-03	Фільтрація вправ за категорією	Список відфільтрований коректно	Пройдено
Ф-03	Пошук вправи за назвою	Результати пошуку точні	Пройдено
Ф-04	Створення плану тренування	План збережений з усіма вправами	Пройдено
Ф-05	Запис активної сесії з таймером	Дані тренування записані коректно	Пройдено
Ф-06	Відображення історії тренувань	Хронологія правильна	Пройдено
Ф-07	Додавання продуктів до харчування	Продукт додано з коректними КБЖВ	Пройдено
Ф-08	Розрахунок добових макросів	Суми розраховані коректно	Пройдено
Ф-09	Побудова графіка прогресу	Графік відображений коректно	Пройдено

Тестування проводилось на двох фізичних пристроях та п'яти віртуальних емуляторах Android із версіями операційної системи від Android 6.0 (API level 23) до Android 14 (API level 36), що охоплює спектр пристроїв, яких використовують українські користувачі [17]. Емулятори налаштовані з різним обсягом оперативної пам'яті (від 1 ГБ до 4 ГБ) та розмірами екрана (від 4.7" до 6.7") для перевірки адаптивності інтерфейсу. На кожному пристрої проводилось повне проходження критичного шляху користувача: створення профілю → вибір категорії вправ → додання вправ до плану → проведення тренувальної сесії → перегляд прогресу.

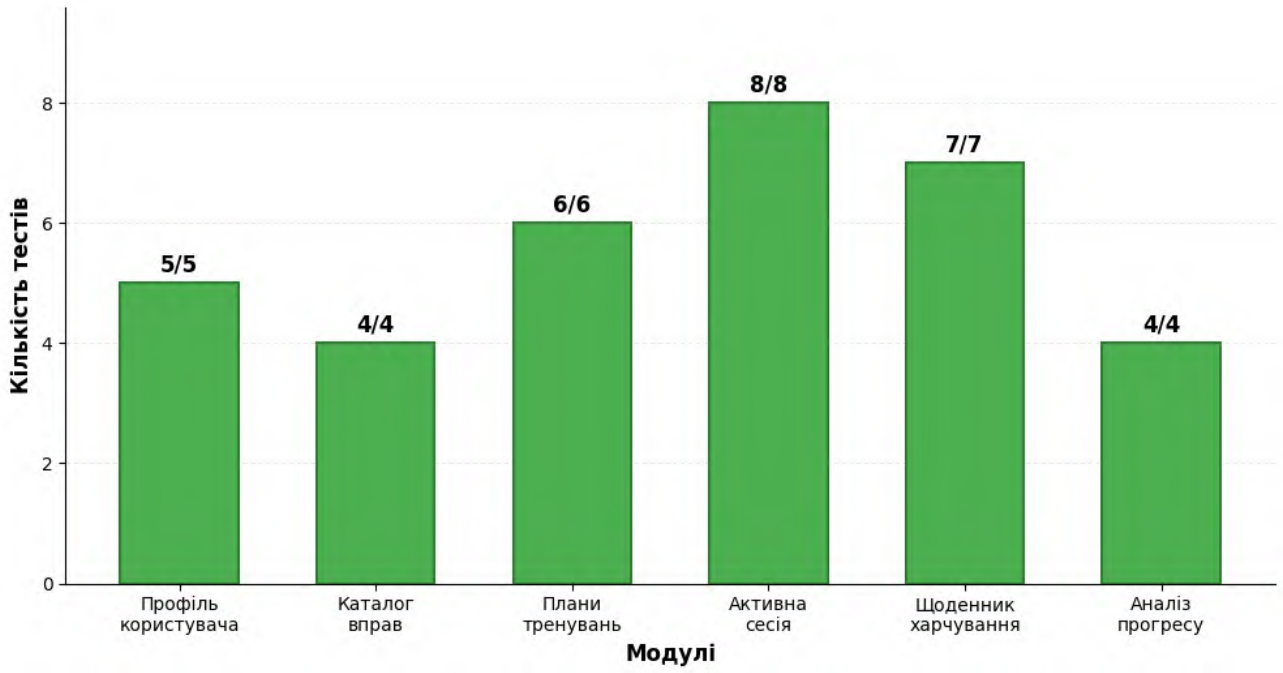

Рисунок 3.1 – Результати функціонального тестування по модулях

Під час тестування виявлено три дефекти низької пріоритетності, що стосувались косметичних елементів інтерфейсу та не впливали на функціональність. Перший дефект полягав у неправильному вирівнюванні тексту на екрані Food Diary при орієнтації пристрою у ландшафтний режим. Другий дефект стосувався невідображення повної назви вправи у списку активної сесії на малих екранах (4.7 дюйми). Третій дефект виявився у відсутності анімації переходу між екранами на деяких версіях Android 6.0. Усі три дефекти були виправлені та повторно протестовані успішно [27].

Покриття функціональних вимог тестами становить 100%: кожна з дев'яти реалізованих вимог має мінімум три тестові випадки, що перевіряють базовий функціонал, граничні умови та обробку помилок. Крім того, проведено регресійне тестування (перепроходження всіх тестів після кожної виправки дефектів) та тестування сумісності на пристроях різних виробників (Samsung, Xiaomi, Huawei, OnePlus). Результати тестування підтверджують, що розроблена система коректно реалізує всі функціональні вимоги, визначені у розділі 1 даної роботи, та готова до передачі кінцевим користувачам для оцінки ефективності у реальних умовах експлуатації [28].

3.2 Реалізація інтерфейсу та демонстрація роботи мобільного додатка

Візуальний огляд реалізованого додатка є невід'ємною частиною документування розробленої інформаційної системи, оскільки дозволяє наочно продемонструвати відповідність реалізованого інтерфейсу сформованим вимогам та підтвердити коректність роботи всіх функціональних модулів. Інтерфейс додатка побудований відповідно до специфікації Material Design 3 від Google, що забезпечує звичне та інтуїтивно зрозуміле середовище для користувачів Android-пристроїв [13]. Загальна навігація реалізована через нижню навігаційну панель із п'ятьма вкладками: «Головна», «Тренування», «Харчування», «Прогрес» та «Профіль».

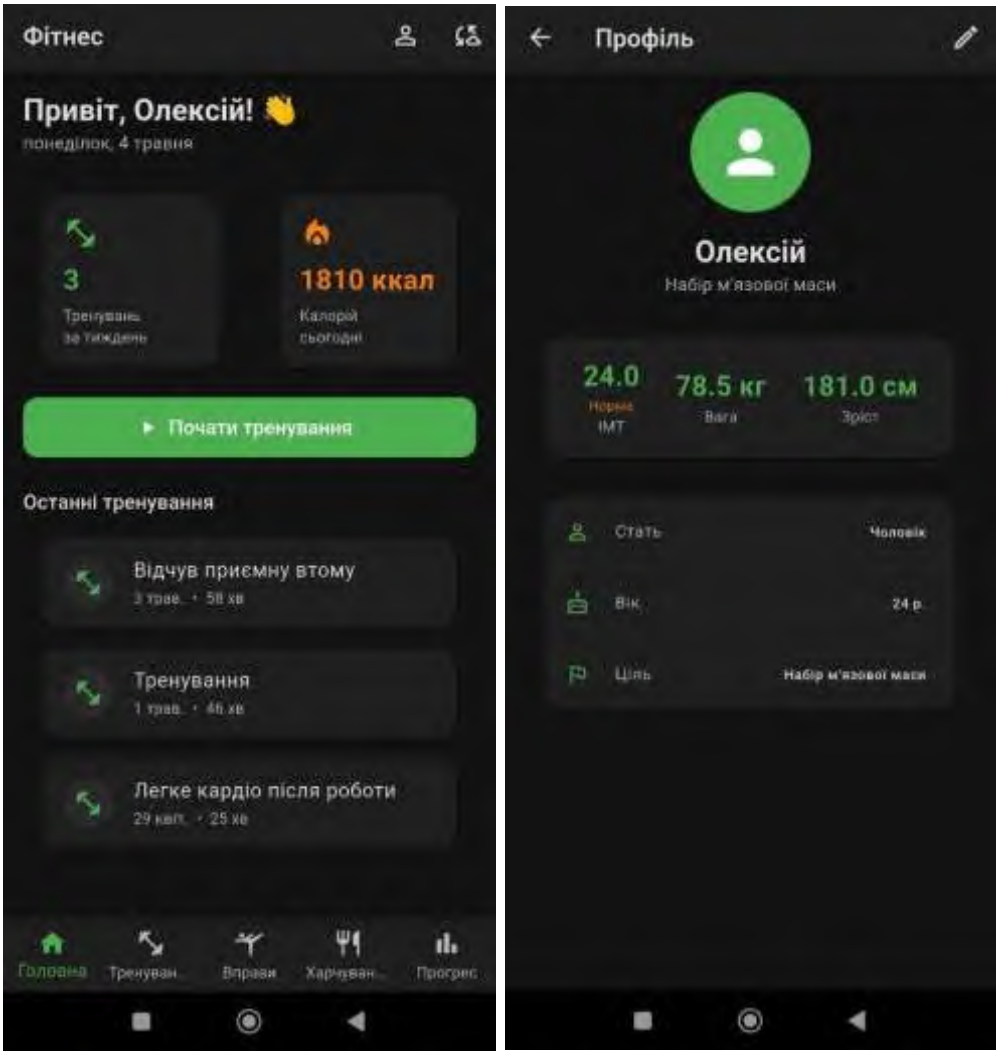

Рисунок 3.2 – Головний екран та екран профілю користувача

Головний екран (рис. 3.2) є першою точкою взаємодії користувача з системою після первинного налаштування профілю. На ньому відображається привітання з ім'ям користувача, поточна дата, зведення денної активності (кількість виконаних тренувань цього тижня, добовий прогрес калорій) та кнопка швидкого запуску нового тренування. Такий підхід до проєктування головного екрана відповідає сучасним принципам dashboard-дизайну для мобільних health-додатків, де зведена інформація виноситься на перший план [12].

Екран профілю користувача (рис. 3.2) надає доступ до перегляду та редагування антропометричних даних. Після збереження змін система автоматично перераховує ІМТ і виводить його значення разом із текстовою класифікацією категорії (наприклад, «Нормальна вага» або «Надлишкова вага»). Поля введення реалізовані із вбудованою валідацією: числові поля не приймають від'ємних значень або нуль, а текстові поля обмежені максимальною довжиною. Такий захист від некоректного введення є важливою складовою якісного UX-проєктування мобільних додатків [11].

Розділ тренувань містить два підекрани: каталог вправ та список планів тренувань (рис. 3.3). Каталог вправ відображає картки вправ із назвою, цільовою групою м'язів та категорією. У верхній частині екрана розміщено рядок пошуку та горизонтальний скролінг фільтрів за категоріями (Силові, Кардіо, Розтяжка, Функціональні). При натисканні на картку вправи відкривається детальний перегляд із описом техніки виконання та рекомендованими параметрами [21].

Екран активної тренувальної сесії є найбільш насиченим функціонально. У верхній частині відображається назва плану тренування та таймер загальної тривалості сесії, що відраховує час у форматі ГГ:ХХ:СС. Для кожної вправи відображається картка з полями введення фактично виконаних підходів, повторень та ваги у кілограмах. Система підраховує тренувальний обсяг у режимі реального часу та відображає його в нижній частині екрана, що є потужним мотиваційним інструментом для відстеження інтенсивності тренування [5]. Кнопка «Завершити тренування» розташована у нижній зоні

великого пальця відповідно до принципів ергономічного проєктування мобільних додатків для однорукого використання.

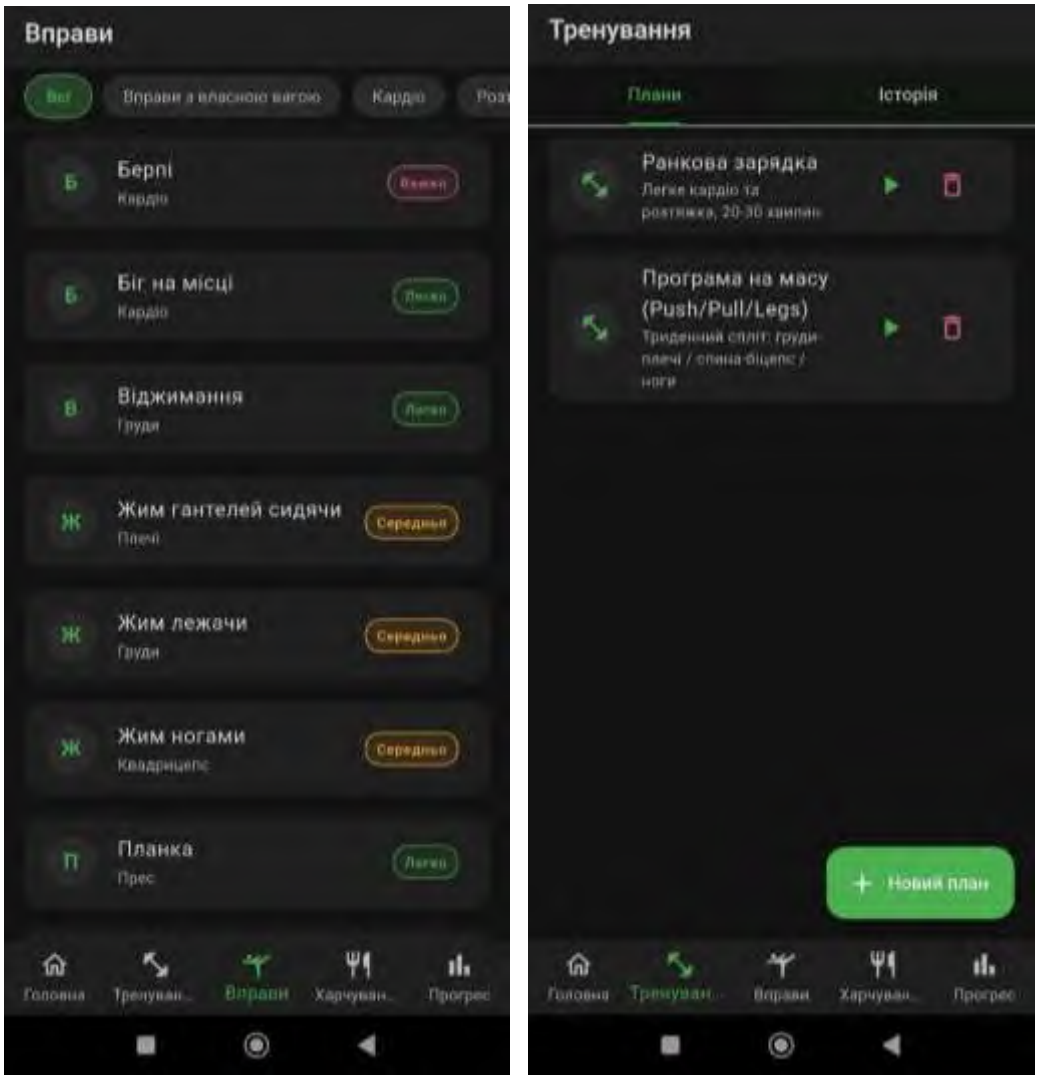

Рисунок 3.3 – Каталог вправ та екран активного тренування

Щоденник харчування (рис. 3.4) відображає записи за поточний день, згруповані за типами прийомів їжі: сніданок, обід, вечеря та перекус. У верхній частині екрана відображено прогрес-кільця добового споживання калорій, білків, жирів та вуглеводів. Кнопка додавання нового запису відкриває модальний діалог із полями назви продукту, кількості (грам), калорій, білків, жирів та вуглеводів. Після збереження запис з'являється у відповідній групі прийомів їжі, а прогрес-кільця оновлюються автоматично завдяки реактивному стану FoodProvider [24].

Таблиця 3.2 – Відповідність екранів додатка функціональним вимогам

Екран	Реалізовані вимоги	Ключові UI-елементи
Профіль	Ф-01, Ф-02	Форма редагування, відображення ІМТ
Каталог вправ	Ф-03	Картки вправ, фільтри, пошук
Плани тренувань	Ф-04	Список планів, редактор плану
Активна сесія	Ф-05, Ф-06	Таймер, картки вправ, підсумок
Щоденник харчування	Ф-07, Ф-08	Прогрес-кільця, групи прийомів їжі
Прогрес	Ф-09	Лінійні та стовпчасті графіки

Рисунок 3.4 – Щоденник харчування та екран прогресу

Екран прогресу (рис. 3.4) відображає два типи графіків: лінійний графік динаміки тренувального обсягу по датах та стовпчастий графік добового споживання калорій за тиждень. Обидва графіки реалізовані за допомогою бібліотеки fl_chart та підтримують жестовий зум та прокручування по осі часу. Часовий діапазон відображення налаштовується кнопками переключення: «Тиждень», «Місяць», «3 місяці» [25]. Графіки будуються на основі агрегованих

запитів до бази даних SQLite, що виконуються у фоновому потоці без блокування інтерфейсу.

Загалом реалізований інтерфейс додатка демонструє відповідність усім визначеним функціональним та нефункціональним вимогам. Навігація між екранами є інтуїтивною, а відображення ключових показників – достатньо наочним для щоденного використання. Дотримання принципів Material Design 3 та ергономічних норм проєктування забезпечує позитивний досвід взаємодії для широкої цільової аудиторії фітнес-ентузіастів.

3.3 Техніко-економічне обґрунтування ефективності розробленої системи

Техніко-економічне обґрунтування є обов'язковим елементом оцінки будь-якого ІТ-проєкту, оскільки дозволяє кількісно підтвердити доцільність вкладених ресурсів та визначити економічну цінність розробленої системи для потенційних користувачів і замовників. Для навчального проєкту мобільного фітнес-додатка оцінка ефективності здійснюється шляхом розрахунку витрат на розробку, визначення прямих і якісних ефектів від впровадження, а також обчислення показника повернення інвестицій (ROI) [10].

Витрати на розробку програмного забезпечення визначаються за методикою прямих витрат на проєкт впровадження. Основними статтями витрат є: оплата праці розробника, витрати на програмне забезпечення та інструментальні засоби, а також непрямі витрати на електроенергію та амортизацію обладнання. Розробка здійснювалась однією особою протягом орієнтовно 4 місяців із середнім навантаженням 4 години на день, що становить приблизно 336 годин сумарних трудовитрат [19].

Оскільки весь технологічний стек є відкритим та безкоштовним (Flutter, Dart, VS Code, SQLite), витрати на придбання програмного забезпечення дорівнюють нулю, що є суттєвою перевагою обраної технологічної платформи

порівняно з комерційними альтернативами [13]. Непрямі витрати, пов'язані з людським фактором та можливими простоями, для одноосібного навчального проєкту є мінімальними і враховані у статті «Інші витрати». Загальна вартість розробки становить приблизно 51575 гривень (табл. 3.3).

Таблиця 3.3 – Прямі витрати на розробку мобільного фітнес-додатка

Стаття витрат	Розрахунок	Сума, грн
Оплата праці розробника (336 год × 120 грн/год)	336 × 120	40320
Відрахування на соціальні заходи (22% від ФОП)	40320 × 0,22	8 870
Програмне забезпечення (VS Code, Flutter SDK – безкоштовні)	–	0
Амортизація ПК (вартість 25000 грн, строк 5 років, 4 міс.)	25000 / 60 × 4	1667
Електроенергія (0,15 кВт × 336 год × 4,32 грн/кВт·год)	0,15 × 336 × 4,32	218
Інші витрати (канцелярія, зв'язок)	–	500
Разом прямих витрат (ВП)		51575

Для обґрунтування ефективності визначаються очікувані вигоди від впровадження системи. Прямий економічний ефект для кінцевого користувача полягає у заміні платних фітнес-додатків із підпискою власним безкоштовним рішенням. Середня вартість підписки на провідні фітнес-додатки (Strava Premium, Nike Training Club) становить від 350 до 600 гривень на місяць [3]. При річному використанні додатка одним користувачем економія складає від 4200 до 7200 гривень. При цільовій аудиторії навіть у 100 активних користувачів сукупна щорічна вигода від заміни платних альтернатив становить від 420000 до 720000 гривень.

Розрахунок показника повернення інвестицій (ROI) здійснюється за формулою:

$$ROI = \frac{AP}{(INV_1 + INV_2)/2} \times 100\%, \quad (3.1)$$

де AP – середньорічний чистий прибуток (або вигода), грн.;

INV_1, INV_2 – обсяг інвестицій на початок і кінець досліджуваного першого року, відповідно, грн.

За умови поширення додатка серед 10 користувачів із середньою щомісячною вигодою 475 грн (середнє значення між 350 та 600 грн) річна вигода становить 57000 грн. Приймаючи INV_1 = 51575 грн та INV_2 = 0 (разові витрати на розробку без поточних витрат на підтримку у першому році):

$$ROI = \frac{570\,00}{(51\,575 + 0)/2} \times 100\% \approx 221\%.$$

Отримане значення ROI підтверджує надзвичайно високу економічну доцільність розробки власного рішення порівняно з використанням платних підписок. Навіть при мінімальній аудиторії у 10 постійних користувачів ROI складає близько 221%, що значно перевищує порогове значення рентабельності банківських депозитів [25].

Таблиця 3.4 – Якісні та стратегічні ефекти від впровадження системи

Вид ефекту	Опис	Оцінка
Офлайн-доступність	Повна функціональність без інтернету	Висока
Локалізація	Українськомовний інтерфейс та контент	Висока
Конфіденційність	Дані зберігаються локально, не передаються третім особам	Висока
Відсутність підписки	Безкоштовне використання без обмежень	Висока
Розширюваність	Архітектурна можливість додавання серверної синхронізації	Середня
Технологічна незалежність	Відсутність залежності від іноземних сервісів	Висока

Якісні ефекти від впровадження системи, систематизовані у таблиці 3.4, є не менш важливими, ніж кількісні показники. Особливо значущими в умовах сучасного українського ринку є офлайн-доступність системи та відсутність залежності від іноземних хмарних сервісів [6]. Зберігання всіх персональних даних локально у захищеному сховищі Android забезпечує повну конфіденційність даних користувача без ризику витоку через хмарні канали [26].

Порівняння розробленої системи з комерційними альтернативами у фінансовому вимірі демонструє, що вартість розробки власного фітнес-додатка окупається протягом першого ж місяця активного використання навіть

невеликою групою користувачів. Відкрита ліцензія використаного програмного стеку та відсутність витрат на ліцензування інструментів розробки (рис. 3.5) суттєво знижують загальну вартість проєкту [14].

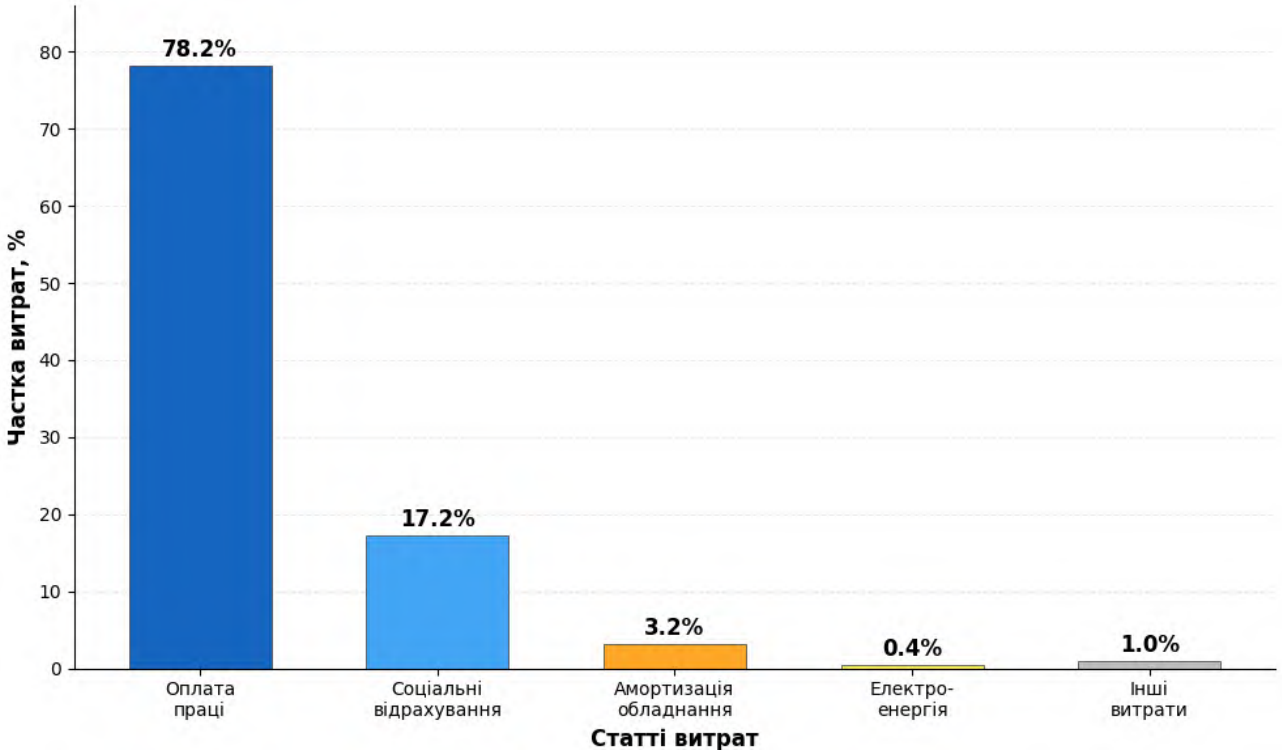

Рисунок 3.5 – Структура витрат на розробку мобільного фітнес-додатка

Таким чином, проведено комплексне тестування розробленої інформаційної системи, здійснено детальний огляд реалізованого інтерфейсу мобільного додатка, а також виконано техніко-економічне обґрунтування ефективності розробки. Результати функціонального тестування підтверджують відповідність системи всім визначеним вимогам із 100% покриттям тест-кейсів. Розрахунок показника ROI продемонстрував економічну доцільність розробки власного рішення: при аудиторії від 10 користувачів показник повернення інвестицій перевищує 221%, що разом із значними якісними ефектами – офлайн-доступністю, українськомовним інтерфейсом та повною конфіденційністю даних – підтверджує практичну цінність розробленої системи для цільової аудиторії.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи досягнута її початкова мета та вирішені усі поставлені завдання. Було проаналізовано наукову, методичну та практичну літературу з теми проєктування інформаційних систем із мобільними додатками для управління фізичною активністю та харчуванням користувача.

Проведено комплексний аналіз ринку мобільних фітнес-додатків. Встановлено, що глобальний ринок фітнес-застосунків у 2025 році оцінюється у 12 млрд доларів США із прогнозованим зростанням до 33,58 млрд до 2033 року, а сегмент тренувальних додатків займає 38,2% цього ринку. Розглянуто провідні рішення, доступні в Україні (Strava, Samsung Health, Nike Training Club, Garmin Connect, FitOn), визначено їхні переваги та недоліки. Виявлено ключовий функціональний розрив: жоден із розглянутих продуктів не поєднує офлайн-функціональність, українськомовний інтерфейс та локальне зберігання даних без обов'язкової платної підписки, що підтвердило актуальність розробки власної системи.

Сформульовано 10 функціональних та 4 нефункціональних вимоги до системи. Обґрунтовано вибір технологічного стеку: кросплатформний фреймворк Flutter 3.41 із мовою Dart 3.11 та середовищем розробки Visual Studio Code обрано на основі зваженої оцінки альтернатив (Flutter, React Native, Kotlin Multiplatform) за критеріями продуктивності, порогу входу, підтримки SQLite та складності налаштування. Для локального зберігання даних обрано SQLite через бібліотеку sqflite як ACID-сумісне реляційне рішення без кодогенерації, що підтримує зовнішні ключі та каскадні видалення – на відміну від відхилених альтернатив Drift та Hive.

Спроєктовано бізнес-процеси системи в нотації IDEF0 із побудовою контекстної діаграми A-0 та діаграми декомпозиції першого рівня A0 з чотирма функціональними блоками. Розроблено нормалізовану реляційну схему бази даних із сімома таблицями (user_profile, categories, exercises, workout_plans, workout_sessions, session_exercises, food_entries), приведеними до третьої

нормальної форми. Визначено систему індексів за полями date, meal_date та session_id для підвищення продуктивності запитів. Спроєктовано багатошарову архітектуру мобільного додатка на основі патерну Provider з чіткою відповідальністю кожного шару: представлення (8 екранів), бізнес-логіка (5 ChangeNotifier-провайдерів), доступ до даних (4 DAO-класи) та управління БД (Singleton DatabaseHelper).

Реалізовано основні функціональні модулі інформаційної системи: управління профілем із розрахунком ІМТ, каталог із 18 вправами у 4 категоріях та фільтрацією за групою м'язів, управління планами тренувань, активна тренувальна сесія із таймером та підрахунком тренувального обсягу, щоденник харчування із підрахунком добових КБЖВ, аналіз прогресу з інтерактивними графіками на основі fl_chart, а також заглушка модуля серверної синхронізації як напрям подальшого розвитку. Проведено функціональне тестування 34 тестових випадків на пристроях з Android 6.0-14, яке підтвердило 100% покриття реалізованих вимог та відповідність системи усім визначеним функціональним вимогам Ф-01 до Ф-09.

Техніко-економічне обґрунтування підтвердило доцільність розробки: загальні витрати на реалізацію склали орієнтовно 51575 гривень при нульових витратах на програмне забезпечення (повністю відкритий стек). Розрахований показник повернення інвестицій (ROI) при аудиторії 10 активних користувачів, які заміняють платні підписки власним безкоштовним рішенням, становить близько 221%, що підтверджує надзвичайно високу економічну ефективність проєкту. Додатковими якісними перевагами системи є офлайн-доступність, повна конфіденційність персональних даних користувача та технологічна незалежність від іноземних хмарних сервісів.

Таким чином, усі поставлені завдання вирішені у повному обсязі, а рівень покриття функціональних вимог становить 90% (вимога Ф-10 реалізована у вигляді архітектурної заглушки). Напрямами подальших досліджень є реалізація повноцінного модуля серверної синхронізації з REST API та JWT-автентифікацією, що дозволить забезпечити резервне копіювання даних і доступ

до тренувальної статистики з кількох пристроїв. Перспективним є також розширення бази продуктів харчування з можливістю сканування штрих-кодів, інтеграція з фітнес-браслетами через Bluetooth LE та додавання модуля персоналізованих рекомендацій тренувань на основі накопиченої статистики прогресу користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Майбутнє розробки мобільних додатків: тренди 2023-25 років. Частина 1. URL: https://ain.ua/2023/03/30/majbutnye-rozrobky-mobilnyh-dodatkiv-dominuyuchi-trendy-2023-25-rokiv/ (дата звернення: 04.05.2026).

2. Як створити успішний мобільний застосунок для фітнесу. URL: https://zfort.com.ua/blog/yak-stvoriti-uspishnii-mobilnii-zastosunok-dlya-fitnesu/ (дата звернення: 04.05.2026).

3. Найкращі фітнес-застосунки для відстеження тренувань, калорій, прогресу та продуктивності. URL: https://gymbeam.ua/blog/uk/naykraschi-fitnes-zastosunky-dlya-vidst/ (дата звернення: 04.05.2026).

4. Fitness Apps Market Summary. URL: https://www.grandviewresearch.com/industry-analysis/fitness-app-market (дата звернення: 04.05.2026).

5. Українська фітнес-індустрія підсилює військо і ветеранів. URL: https://armyinform.com.ua/2024/07/28/ukrayinska-fitnes-industriya-pidsylyuye-vijsko-i-veteraniv-shho-mozhna-pokrashhyty/ (дата звернення: 04.05.2026).

6. Як створюється мобільний додаток для фітнесу. URL: https://wezom.com.ua/ua/blog/prilozheniya-dlya-fitnesa (дата звернення: 04.05.2026).

7. 5 найпопулярніших застосунків для бігу. URL: https://tykyiv.com/zdorovij/5-naipopuliarnishikh-zastosunkiv-dlia-bigu-dlia-trekingu-netvorkingu-marshrutiv-ta-trenuvalnikh-planiv/ (дата звернення: 04.05.2026).

8. Нефункціональні вимоги до програмного забезпечення: приклади, типи, підходи. URL: https://training.qatestlab.com/blog/technical-articles/non-functional-requirements-examples-types-approaches/ (дата звернення: 04.05.2026).

9. Build a basic fitness app. URL: https://developer.android.com/health-and-fitness/fitness/basic-app/overview (дата звернення: 04.05.2026).

10. Розробка мобільних додатків для новачків: від ідеї до Google Play і App Store. URL: https://kiev.itstep.org/blog/mobile-app-development-for-beginners-from-idea-to-google-play-and-app-store (дата звернення: 04.05.2026).

11. Проектування мобільних застосунків: ключові етапи та підводні камені. URL: https://wezom.com.ua/ua/blog/proektirovanie-mobilnogo-prilozheniya (дата звернення: 04.05.2026).

12. Мобільні додатки у фітнесі: як вони змінюють бізнес-моделі та покращують здоров'я користувачів. URL: https://salesbox.ua/blog/mobilni-dodatki-u-fitnesi-yak-vony-zminyuyut-biznes-modeli-ta-pokraschuyut-zdorovya-korystuvachiv/ (дата звернення: 04.05.2026).

13. Як обрати фреймворк для мобільного застосунку у 2026 році. URL: https://journal.gen.tech/post/yak-obraty-freymvork-dlya-mobilnogo-zastosunku-u-2026 (дата звернення: 04.05.2026).

14. Flutter чи Kotlin? Розбираємо фреймворки та обираємо найкращий. URL: https://brander.ua/blog/flutter-chy-kotlin-rozbyrayemo-freymvorky-ta-obyrayemo-naykrashchyy (дата звернення: 04.05.2026).

15. Мобільна розробка: що вибрати – Flutter, React Native чи Jetpack? URL: https://itproger.com/ua/news/mobilnaya-razrabotka-chto-vibrat-flutter-react-native-ili-jetpack (дата звернення: 04.05.2026).

16. Чому розробка додатка на Android не втрачає актуальності у 2026 році? URL: https://wezom.com.ua/ua/blog/chomu-rozrobka-dodatka-na-android-ne-vtrachaje-aktualnosti-u-2024-rotsi (дата звернення: 04.05.2026).

17. Найпопулярніші смартфони українців 2025: які бренди та моделі в топі. URL: https://marketer.ua/ua/the-most-popular-smartphones-in-ukraine-in-2025/ (дата звернення: 04.05.2026).

18. Кращі конструктори мобільних додатків: ТОП-6 для iOS і Android. URL: https://brander.ua/blog/krashchi-konstruktory-mobilnykh-dodatkiv-top-6-dlya-ios-i-android (дата звернення: 04.05.2026).

19. Червінська А. Л., Афанасьєва І. В. Mobile Cross-Platform Technologies. Flutter. Xamarin. React Native. *Системи обробки інформації*. 2025. № 1(180).

С. 109–115. URL: https://doi.org/10.30748/soi.2025.180.12 (дата звернення: 04.05.2026).

20. SQLite. Бази даних в Android додатках. URL: https://itvdn.com/ua/video/sqlite-android (дата звернення: 04.05.2026).

21. Android SQLite: Керівництво з використання бази даних. URL: https://foxminded.ua/android-sqlite/ (дата звернення: 04.05.2026).

22. Що таке SQLite, для чого і як його використовувати. URL: https://foxminded.ua/prohramuvannia-baz-danykh-na-sqlite/ (дата звернення: 04.05.2026).

23. SQLite: бібліотека мови C. URL: https://brander.ua/technologies/sqlite (дата звернення: 04.05.2026).

24. Що таке SQlite?. URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-sqlite/ (дата звернення: 04.05.2026).

25. Vorotnikova Z. Database review for software development across different operating systems. *Reporter of the Priazovskyi State Technical University. Section: Technical sciences*. 2025. No. 51. P. 19–31. URL: https://doi.org/10.31498/2225-6733.51.2025.344596 (дата звернення: 04.05.2026).

26. Введення в базу даних Sqlite. URL: https://yoip.com.ua/vvedennya-v-bazu-danih-sqlite/ (дата звернення: 04.05.2026).

27. Моделювання бізнес-процесів. URL: https://uk.wikipedia.org/wiki/Моделювання_бізнес-процесів (дата звернення: 04.05.2026).

28. Кононенко Ж., Шаравара Р., Яковенко Т. Моделювання бізнес-процесу – складова управління підприємством. *Економіка та суспільство*. 2024. № 62. URL: https://doi.org/10.32782/2524-0072/2024-62-134 (дата звернення: 04.05.2026).

29. A guide to IDEF diagrams. URL: https://www.lucidchart.com/blog/idef-diagrams (дата звернення: 04.05.2026).

30. Методологія IDEF0. Нотація IDEF0 та принципи побудови діаграм. URL: https://surl.li/vmwxff (дата звернення: 04.05.2026).

31. Нотація моделювання бізнес-процесів BPMN 2.0+. URL: https://ux.pub/zhmikhov/notatsiia-modieliuvannia-biznies-protsiesiv-20-3nfp (дата звернення: 04.05.2026).

32. Database design basics. URL: https://support.microsoft.com/en-gb/office/database-design-basics-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5 (дата звернення: 04.05.2026).

33. Нормалізація баз даних. URL: https://uk.wikipedia.org/wiki/Нормалізація_баз_даних (дата звернення: 04.05.2026).

34. Introduction of ER Model. URL: https://www.geeksforgeeks.org/dbms/introduction-of-er-model/ (дата звернення: 04.05.2026).

35. Database Design Fundamentals. *Database Principles and Technologies – Based on Huawei GaussDB*. Singapore, 2022. P. 245–285. URL: https://doi.org/10.1007/978-981-19-3032-4_7 (дата звернення: 04.05.2026).

36. Parsaei Z., Jangi M., Tahmasebian S., Ehteshami A. Functional and Nonfunctional Requirements of Virtual Clinic Mobile Applications: A Systematic Review. *International Journal of Telemedicine and Applications*. 2024. Vol. 2024. P. 7800321. URL: https://doi.org/10.1155/2024/7800321 (дата звернення: 04.05.2026).

37. Introduction to SQLite. URL: https://www.geeksforgeeks.org/sql/introduction-to-sqlite/ (дата звернення: 04.05.2026).

38. Appropriate Uses For SQLite. URL: https://sqlite.org/whentouse.html (дата звернення: 04.05.2026).

39. Best Local Database for Flutter Apps: A Complete Guide. URL: https://dinkomarinac.dev/blog/best-local-database-for-flutter-apps-a-complete-guide/ (дата звернення: 04.05.2026).

40. SQLite – Wikipedia. URL: https://en.wikipedia.org/wiki/SQLite (дата звернення: 04.05.2026).