

МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: Створення додатку для підрахунку калорій на IOS

Здобувача вищої освіти першого (бакалаврського) рівня,
IV курсу, групи КБ-22-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за галузями)»


(підпис)

Шкодiна Тетяна
Вадимiвна
(прізвище, ім'я, по батькові)

Науковий керівник


(підпис)

Грищенко Світлана
Миколаївна, канд. пед.наук

(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Керівник освітньо-
професійної програми


(підпис)

Ніжегородцев Владислав
Олександрович, канд. пед.
наук, доцент

(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Допущено до захисту за рішенням кафедри комп'ютерних та інформаційних
технологій і систем протокол від «12» травня 2026 № 20

Завідувач кафедри


(підпис)

Філоненко Михайло
Миколайович, канд. фіз.-мат.
наук, доцент

(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Реєстрація № 4527.05.2026

(дата)



(підпис відповідальної особи)

Результати перевірки

до захисту
(до захисту, на доопрацювання)12.05.2026

(дата)



(підпис)

Результат захисту

5 (відмінно)
(оцінка)08.06.2026

(дата)



(підпис)

Голова комісії:



(підпис)

О.С. Дюбанов

Члени комісії:



(підпис)

М.М. Філоненко

(підпис)

В.О. Ніжегородцев

(підпис)

А.А. Омельчук

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	28		08.06.2026 Грищенко С.М. Дюбанов О.С.
2. Дотримання вимог щодо оформлення	від 0 до 10	10		Філоненко М.М. Ніжегородцев В.О.
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		Омельчук А.А. Суховерх Л.П.
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	47		

Оцінювання кваліфікаційної роботи науковим керівником – 45 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
92	A	5 (відмінно)	08.06.2026	Дюбанов О.С. Філоненко М.М. Ніжегородцев В.О. Омельчук А.А. Суховерх Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
 Кафедра комп'ютерних та інформаційних технологій і систем
 Галузь знань: 12 «Інформаційні технології»
 Спеціальність: 122 «Комп'ютерні науки»
 Освітня програма: «Інформаційні управляючі системи і технології (за галузями)»

Затверджую
 Завідувач кафедри  Філоненко М.М.
 «01» грудня 2025

Завдання на кваліфікаційну роботу
 здобувачу вищої освіти
Шкодіній Тетяні Вадимівні
 (прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Створення додатку для підрахунку калорій на IOS

Затверджена наказом ректора від «28» листопада 2025 №1526

2. Строк здачі закінченої роботи «08» травня 2026

3. Мета роботи створення iOS-додатку для ведення щоденника харчування та контролю нутрієнтів з оптимізацією пошуку продуктів.

Об'єкт дослідження процес створення мобільних застосунків для моніторингу показників здоров'я користувачів на базі iOS.

Предмет дослідження методи та інструменти створення додатка для підрахунку калорій.

Орієнтована база дослідження створений мобільний додаток FitFood.

4. Вихідні дані до роботи теоретичні засади розробки на Swift/SwiftUI, документація Apple HealthKit, API OpenFoodFacts та FatSecret.

5. Перелік графічного матеріалу діаграма класів, схема бази даних (SQLite/Firebase), прототипи інтерфейсу, скріншоти тестування функціоналу.

6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)
ЗМІСТ

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР ТЕХНОЛОГІЙ СТВОРЕННЯ.

1.1. Дослідження ринку мобільних додатків для контролю харчування.

1.2 Обґрунтування технологічного стеку створення додатка на iOS.

РОЗДІЛ 2. ТЕХНОЛОГІЯ ПРОЄКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ НА IOS.

2.1. Дизайн та етапи створення користувацького інтерфейсу.

2.2 Архітектура та проєктування баз даних додатку FitFood.

РОЗДІЛ 3. ТЕСТУВАННЯ, ОЦІНКА ЯКОСТІ ТА ПРАКТИЧНЕ ВПРОВАДЖЕННЯ ПРОГРАМНОГО ПРОДУКТУ

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 30.10.2025	До 30.10.2025
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	26.12.2025	26.12.2025
3	II розділ (аналіз експериментальної частини про методики досліджень)	18.02.2026	18.02.2026
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	16.04.2026	16.04.2026
5	Висновки	01.05.2026	01.05.2026
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	07.05.2026	07.05.2026
7	Отримання відгуку керівника	29.05.2026	29.05.2026
8	Отримання зовнішньої рецензії	01.06.2026	01.06.2026

8. Керівник кваліфікаційної роботи


(підпис)

Грищенко С.М.
(прізвище, ініціали)

9. Керівник освітньо-професійної програми


(підпис)

Ніжегородцев
(прізвище, ініціали)

10. Завдання до виконання отримав здобувач вищої освіти


(підпис)

Шкодінa Т.В.
(прізвище, ініціали)

АНОТАЦІЯ

Створення додатку для підрахунку калорій на iOS

Мета роботи полягає у створенні мобільного додатку на iOS для ведення щоденника харчування, та контролю балансу макронутрієнтів для покращення користувацького досвіду шляхом швидкого пошуку продуктів та оптимізації роботи з даними.

Завданнями роботи є аналіз предметної області та аналогів; обґрунтування технологічного стеку; розробка UI/UX інтерфейсу на SwiftUI; інтеграція локальної бази SQLite та хмарної синхронізації Firebase; налаштування взаємодії з FatSecret та OpenFoodFacts; проведення тестування.

Предметом дослідження є методи, алгоритми та інструментальні засоби створення мобільного додатку для підрахунку калорій, що включає проєктування інтерфейсу (UI/UX), реалізацію бази даних продуктів та інтеграцію з системними сервісами (наприклад, Apple HealthKit).

Об'єктом дослідження виступає створення мобільних застосунків для моніторингу та аналізу показників здоров'я користувачів на базі операційної системи iOS.

Базою дослідження виступає створений мобільний додаток FitFood.

За результатами дослідження сформовано архітектуру та програмний код додатку FitFood, що автоматизує облік їжі. Продукт забезпечує автономну роботу, сканування штрих-кодів та синхронізацію даних.

Одержані результати можуть бути використані користувачами для контролю харчування, а застосовані архітектурні підходи - IT-спеціалістами у подальшому створенні подібних мобільних сервісів.

Кваліфікаційна робота містить 65 сторінок, 5 таблиць, 14 рисунків, список використаних джерел 40 найменувань.

Ключові слова: мобільний додаток, iOS, підрахунок калорій, бази даних, щоденник харчування, mHealth, Apple HealthKit.

Рік виконання кваліфікаційної роботи 2026

Рік захисту роботи 2026

ABSTRACT

Creation of a calorie counting application on iOS

The aim of this work is the creation of a mobile application on iOS for maintaining a food diary and controlling the balance of macronutrients to improve the user experience through quick product search and data optimization.

The tasks of the work are as follows analyzing the subject area and analogues; justifying the technology stack; developing the UI/UX interface using SwiftUI; integrating the local SQLite database and Firebase cloud synchronization; setting up interaction with FatSecret and OpenFoodFacts; and conducting testing.

The subject of the research is the methods, algorithms, and instrumental tools for creating a calorie-counting mobile application, which includes interface design (UI/UX), product database implementation, and integration with system services (e.g., Apple HealthKit).

The object of the research is the process of creating mobile applications for monitoring and analyzing user health indicators based on the iOS operating system.

The research is based on the developed FitFood mobile application.

The results of the research include the created architecture and software code of the FitFood application, which automates food tracking. The product provides offline operation, barcode scanning, and data synchronization.

The obtained results can be used by users for nutrition control, and the applied architectural approaches can be used by IT specialists in the further development of similar mobile services.

The qualification work consists of 65 pages, 5 tables, 14 images, a list of 40 references.

Keywords: mobile application, iOS, calorie counting, databases, food diary, mHealth, Apple HealthKit.

Year of qualification work: 2026

Year of work defense: 2026

ЗМІСТ

ВСТУП.....	9
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР ТЕХНОЛОГІЙ СТВОРЕННЯ	12
1.1 Дослідження ринку мобільних додатків для контролю харчування та аналіз аналогів.	12
1.2 Обґрунтування методології, інструментарію та технологій проектування мобільного додатку FitFood на iOS.....	17
РОЗДІЛ 2. ТЕХНОЛОГІЯ ПРОЄКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ НА IOS	
2.1 Дизайн та етапи створення користувацького інтерфейсу	24
2.2 Архітектура та проектування баз даних мобільного додатку FitFood	34
РОЗДІЛ 3. ТЕСТУВАННЯ, ОЦІНКА ЯКОСТІ ТА ПРАКТИЧНЕ ВПРОВАДЖЕННЯ ПРОГРАМНОГО ПРОДУКТУ.....	51
ВИСНОВКИ.....	56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	59

ВСТУП

Сьогодні сучасний світ неможливо уявити без мобільних технологій, які стали невід’ємною частиною нашого повсякденного життя. Зростання популярності смартфонів сприяло стрімкому розвитку індустрії програмного забезпечення, зокрема у сфері охорони здоров’я та фітнесу (mHealth). Користувачі все частіше звертаються до цифрових рішень для відстеження своєї фізичної активності, контролю ваги та ведення здорового способу життя. Збалансоване харчування є одним із ключових факторів підтримки здоров’я, тому інструменти для автоматизації підрахунку спожитих калорій та макронутрієнтів набувають особливої актуальності.

Актуальність теми. В умовах швидкого темпу життя людям складно самостійно контролювати свій раціон, обчислювати співвідношення білків, жирів та вуглеводів, а також шукати харчову цінність кожного окремого продукту. Існуючі рішення часто вимагають постійного підключення до мережі Інтернет або мають обмежені чи незручні бази даних. Тому створення сучасного мобільного додатку, який поєднує локальні та хмарні бази даних, підтримує роботу офлайн, має функцію сканування штрих-кодів та зручний інтерфейс, є актуальним та своєчасним завданням. Такий інструмент здатен значно спростити процес ведення харчового щоденника та підвищити мотивацію користувачів до здорового харчування.

Стан і ступінь розробки проблеми. У спеціальній літературі та на ринку програмного забезпечення широко представлені праці та продукти, присвячені цифровому моніторингу здоров’я. Відзначаються розробки багатьох зарубіжних та вітчизняних ІТ-компаній у сфері фітнес-трекерів. Проте питання одночасної інтеграції кількох міжнародних та локальних баз даних продуктів (таких як FatSecret та OpenFoodFacts) з можливістю автономної роботи та хмарної синхронізації між пристроями все ще потребує вдосконалення та адаптації під потреби сучасного користувача з використанням новітніх декларативних фреймворків, таких як SwiftUI.

Метою кваліфікаційної роботи є створення мобільного додатку на iOS для ведення щоденника харчування, підрахунку калорій та контролю балансу макронутрієнтів, що забезпечить покращення користувацького досвіду шляхом швидкого пошуку продуктів та оптимізації роботи з даними.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- Проаналізувати предметну область, дослідити існуючі аналоги та визначити ключові вимоги до мобільного додатку для підрахунку калорій;
- Обґрунтувати вибір платформи, мови програмування та архітектури баз даних;
- Створити інтуїтивно зрозумілий та сучасний користувацький інтерфейс за допомогою фреймворку SwiftUI;
- Реалізувати інтеграцію з локальною базою даних SQLite для роботи в офлайн-режимі;
- Налаштувати взаємодію з глобальними базами даних FatSecret та OpenFoodFacts для розширеного пошуку продуктів та сканування штрих кодів;
- Впровадити систему збереження та синхронізації даних користувача між пристроями за допомогою Firebase Cloud Firestore;
- Провести тестування функціоналу додатку та оцінити його ефективність.

Об'єктом кваліфікаційної роботи є процес проєктування та створення мобільних застосунків.

Предмет дослідження - архітектура, дизайн та програмна реалізація мобільного додатку для підрахунку калорій для платформи iOS.

Методи дослідження - *теоретичні* - аналіз науково-технічної літератури, узагальнення інформації щодо архітектури мобільних додатків, порівняння існуючих програмних рішень; *емпіричні* - проєктування інтерфейсу користувача, об'єктно-орієнтоване програмування, тестування програмного забезпечення та налагодження роботи баз даних.

Базою дослідження виступає створений мобільний додаток FitFood.

Науково-практична значимість одержаних результатів. Створений додаток вирішує практичну проблему автоматизації обліку спожитої їжі. Використання комбінованої системи баз даних дозволяє додатку працювати без доступу до інтернету, одночасно пропонуючи гнучку синхронізацію при появі мережі. Реалізація пошуку за штрих-кодом та підтримка декількох мов (української та англійської) робить додаток зручним для широкої аудиторії. Одержані результати можуть бути використані користувачами для контролю харчування, а застосовані архітектурні підходи - IT-спеціалістами у подальшому створенні подібних мобільних сервісів.

Особистий внесок здобувача полягає у повному циклі створення мобільного додатку FitFood: від розробки концепції та дизайну інтерфейсу за допомогою SwiftUI до написання програмного коду на мові Swift. Автором самостійно спроектовано локальну базу даних SQLite, реалізовано алгоритми асинхронного пошуку продуктів (з використанням Swift Task), налаштовано розпізнавання штрих-кодів за допомогою DataScannerViewController та інтегровано хмарне сховище Firebase Cloud Firestore.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР ТЕХНОЛОГІЙ СТВОРЕННЯ

1.1 Дослідження ринку мобільних додатків для контролю харчування та аналіз аналогів.

Стрімкий розвиток інформаційно-комунікаційних технологій та повсюдна смартфонізація населення кардинально змінили підходи до моніторингу здоров'я та ведення здорового способу життя. Сьогодні цифровізація охорони здоров'я виокремила в потужний напрямок - mHealth (mobile health), який передбачає використання мобільних пристроїв для надання інформаційних послуг, збору даних про стан здоров'я та контролю за фізіологічними показниками користувачів. В Україні цей напрямок набуває особливої актуальності в контексті реалізації загальнодержавних ініціатив, зокрема Концепції розвитку цифрової економіки та суспільства України[13], що стимулює впровадження електронних рішень у повсякденне життя громадян.

Одним із найважливіших сегментів mHealth є додатки для контролю харчування та підрахунку калорій, оскільки превентивна медицина та нутриціологія визнають збалансований раціон ключовим фактором профілактики багатьох захворювань та підтримки якості життя.

Дослідження проблематики створення та ефективності використання мобільних додатків для контролю харчування активно ведеться світовою та вітчизняною науковою спільнотою. Аналіз сучасних наукових розробок, зокрема магістерських досліджень М. Харіна[16] та С. Вакуленка[2], а також аналіз веб-орієнтованих систем моніторингу нутрієнтів[6], дозволяє виділити три основні наукові школи (підходи), які розглядають цю проблему під різними кутами: медико-біологічну, поведінкову (UX/UI) та інформаційно-технологічну.

Перша наукова школа - медико-біологічна (представлена працями вітчизняних та зарубіжних дослідників у сфері інформаційних систем нутриціології). Дослідники цього напрямку[12] зосереджують увагу на точності математичних моделей, що лежать в основі розрахунку базового обміну речовин

(BMR - Basal Metabolic Rate) та загальної добової витрати енергії (TDEE - Total Daily Energy Expenditure). У сучасних програмних системах найчастіше використовується формула Міффіна-Сан Жеора, яка, згідно з клінічними та технологічними дослідженнями[32], має найменшу статистичну похибку порівняно з алгоритмами Харріса-Бенедикта. Математично ця модель описується наступним чином:

Для чоловіків:

$$\text{BMR} = 10 \cdot m + 6.25 \cdot h - 5 \cdot a + 5 \quad (1.1)$$

Для жінок:

$$\text{BMR} = 10 \cdot m + 6.25 \cdot h - 5 \cdot a - 1615 \quad (1.2)$$

де m - маса тіла у кілограмах, h - зріст у сантиметрах, a - вік у роках. Для визначення фінальної добової норми калорій (TDEE), базовий показник BMR множиться на коефіцієнт фізичної активності (від 1.2 для малорухливого способу життя до 1.9 для професійних спортсменів). Медична школа наголошує, що головна проблема існуючих додатків полягає у відсутності гнучкої адаптації цих математичних формул під динамічні потреби користувача та складності точного візуального відображення балансу макронутрієнтів (білків, жирів, вуглеводів)[12].

Друга наукова школа - поведінкова та психологічна (User-Centered Design). Вчені цього напрямку[35] стверджують, що навіть найточніша математична модель не буде ефективною, якщо програмний продукт не здатний утримати увагу користувача (User Retention). Згідно з дослідженнями принципів людино-машинної взаємодії, основними причинами відмови від використання додатків-щоденників є надмірне когнітивне навантаження: складність ручного введення даних, перевантаженість інтерфейсу, відсутність мікроінтеракцій та рутинність процесу. Дослідники пропонують впроваджувати технології швидкого збору даних (наприклад, апаратні сканери штрих-кодів), інтуїтивно зрозумілі дашборди з круговими діаграмами (Ring Charts) та мінімалістичний дизайн для зниження порогу входу нових користувачів[35].

Третя наукова школа - інформаційно-технологічна (ІТ-орієнтована). Дослідники в галузі архітектури програмного забезпечення концентруються на оптимізації структур баз даних, алгоритмах синхронізації та безперебійності роботи мобільних клієнтів. Сучасні тенденції показують зміщення фокусу від монолітних рішень, повністю залежних від серверної частини, до гібридних архітектур типу «Offline-first». Основною технічною проблемою, яку виділяють ІТ-фахівці, є залежність більшості існуючих фітнес-додатків від постійного доступу до мережі Інтернет для виконання API-запитів до баз продуктів. Це робить їх недієздатними в умовах відсутності зв'язку, що є критичним недоліком для щоденного інструменту. Вирішенням цієї проблеми є використання вбудованих реляційних баз даних (наприклад, SQLite) безпосередньо на пристрої користувача з подальшою асинхронною синхронізацією.

Порівнюючи ці три наукові підходи, можна зробити висновок, що конкурентоспроможний мобільний додаток для контролю харчування повинен мати комплексну архітектуру. Він має об'єднати точність медичних алгоритмів розрахунку (BMR), простоту поведінкового дизайну та надійність гібридної ІТ-архітектури з використанням елементів інтелектуальних смарт-систем контролю харчових компонентів[17].

Для підтвердження теоретичних висновків та виявлення недоліків існуючих систем контролю харчування було проведено емпіричне дослідження сучасного ринку мобільних додатків у категорії «Здоров'я та Фітнес». У якості об'єктів порівняльного аналізу обрано трьох глобальних лідерів сегменту, які мають найбільшу кількість завантажень та активних користувачів: MyFitnessPal, FatSecret та Lifesum.

MyFitnessPal. Будучи одним із піонерів ринку та найпопулярнішим рішенням у світі[29], додаток володіє найбільшою краудсорсинговою базою продуктів, яка наповнюється самими користувачами. Однак технічний та функціональний аналіз показав, що за останні роки розробники значно ускладнили користувацький досвід. Переважна більшість критично важливих функцій була переведена у платну версію (Premium-підписку). Найбільше невдоволення користувачів викликало обмеження

доступу до сканера штрих-кодів, який є базовим інструментом для швидкого ведення щоденника[29]. Крім того, архітектура додатку є монолітною та вимагає постійного інтернет-з'єднання для валідації даних та пошуку продуктів, що повністю порушує принцип безперебійної роботи (Offline-first).

FatSecret. Цей застосунок відрізняється наявністю безкоштовного сканера штрих-кодів та надзвичайно точною міжнародною базою продуктів, яка проходить модерацію[23]. Проте архітектурний підхід до побудови користувацького інтерфейсу (UI) є застарілим і не відповідає сучасним гайдлайнам компанії Apple (Human Interface Guidelines) для платформи iOS. Візуалізація даних про макронутрієнти (білки, жири, вуглеводи) реалізована складно, що суперечить вимогам поведінкової наукової школи щодо необхідності зниження когнітивного навантаження. З технічної точки зору, додаток не підтримує повноцінну локальну базу для власних продуктів користувача - при втраті з'єднання із сервером внесення нових записів стає неможливим[23].

Lifesum. Даний програмний продукт вирізняється надзвичайно високою якістю візуального дизайну, продуманим UX та використанням сучасних мікроінтеракцій, що забезпечує високий рівень утримання аудиторії[31]. Однак система монетизації додатку є надзвичайно агресивною - безкоштовна версія фактично не дозволяє переглядати деталізовану статистику за макронутрієнтами (відображаються лише загальні калорії). Це робить додаток малокорисним для цільової аудиторії, яка професійно слідкує за балансом БЖВ (білків, жирів, вуглеводів), оскільки доступ до детальної аналітики вимагає обов'язкової оплати[31].

Узагальнюючи результати аналізу існуючих програмних рішень, можна виокремити три ключові невирішені проблеми, які потребують створення нового альтернативного продукту:

1. Проблема доступності даних та автономності: Жоден з проаналізованих додатків не пропонує гібридної системи баз даних, яка б гарантувала миттєвий пошук із локального сховища в умовах відсутності

інтернету, і водночас пропонувала б доступ до глобальних відкритих API при підключенні до мережі.

2. Штучна комерціалізація базового функціоналу: Пошук продуктів за штрих-кодом є необхідністю для автоматизації рутини. Блокування цієї функції конкурентами створює нішу для створення повністю безкоштовного інструменту на базі нативних фреймворків сканування для iOS.

3. Перевантаженість інтерфейсу: Відсутність фокусу на візуалізації ключових показників (макронутрієнтів) за допомогою простих графічних елементів (наприклад, кругових діаграм) у безкоштовному доступі.

З огляду на виявлені проблеми, концептуальною ідеєю даної кваліфікаційної роботи є створення мобільного додатку FitFood для операційної системи iOS. Основною метою проєкту є створення інструменту, що усуває недоліки існуючих аналогів шляхом впровадження сучасних технологічних підходів.

Технічна концепція FitFood передбачає створення системи, що поєднує сучасний, чистий декларативний інтерфейс, реалізований за допомогою новітнього фреймворку SwiftUI, та гібридну архітектуру роботи з даними. На відміну від конкурентів, основу зберігання інформації складатиме локальна база даних SQLite (foods.db), яка дозволить користувачеві додавати власні продукти та вести щоденник у режимі повної автономії (офлайн).

Для розширення функціоналу додатку концепцією передбачено інтеграцію з двома потужними зовнішніми джерелами даних (API): глобальною бібліотекою харчових продуктів FatSecret API та відкритою базою OpenFoodFacts. Така синергія дозволить реалізувати високошвидкісний, точний пошук їжі з різних країн та запровадити інтегрований сканер штрих-кодів, який буде доступний користувачам абсолютно безкоштовно. Візуалізація щоденного прогресу буде побудована на системі анімованих кругових діаграм (Ring Charts) та прогрес-індикаторів, що відображатимуть виконання норми білків, жирів та вуглеводів.

Такий комплексний підхід дозволить створити оптимізований, зручний та безперебійний інструмент для підтримки здорового способу життя, що відповідає найвищим стандартам розробки програмного забезпечення.

1.2 Обґрунтування методології, інструментарію та технологій проєктування мобільного додатку FitFood на iOS.

Перехід від теоретичної концепції до практичної реалізації програмного продукту вимагає ретельного аналізу та вибору технологічного стеку. Зважаючи на вимоги, сформовані під час аналізу предметної області (підрозділ 1.1), мобільний додаток для контролю харчування повинен забезпечувати високу продуктивність, безперебійну роботу в режимі офлайн, миттєву реакцію інтерфейсу на дії користувача та безпечну синхронізацію даних[1]. Для досягнення цих цілей необхідно було обрати оптимальну платформу розробки, мову програмування та архітектурні патерни.

На сучасному ринку мобільної розробки існують два основні підходи: кросплатформна створення (за допомогою фреймворків Flutter, React Native тощо) та нативна створення під конкретну операційну систему (iOS або Android). Попри економічну вигоду кросплатформних рішень, для створення додатку FitFood було обрано виключно нативний підхід для платформи iOS.

Вибір нативного створення для проєкту FitFood ґрунтується на комплексному аналізі підходів до створення мобільних додатків, де ключовим фактором є необхідність повної інтеграції з апаратними можливостями iOS-пристроїв[11]. Зокрема, реалізація безкоштовного сканера штрих-кодів (який є ключовою конкурентною перевагою концепції) вимагає використання нативних оптичних фреймворків Apple, таких як DataScannerViewController[8]. Кросплатформні рішення часто працюють із камерою через сторонні бібліотеки-мости (bridges), що уповільнює розпізнавання та збільшує споживання заряду батареї. Нативне створення гарантує максимальну енергоефективність, швидкість відгуку та 100% відповідність гайдлайнам Apple Human Interface Guidelines, що критично важливо для утримання користувачів.

В якості методології проєктування було обрано ітеративну модель розробки з елементами User-Centered Design (дизайну, орієнтованого на користувача). Ця методологія передбачає поступове нарощування функціоналу: від створення

базового щоденника та локальної бази даних до інтеграції складних хмарних рішень та анімацій.

Основною мовою програмування для реалізації додатку FitFood було обрано Swift. Ця мова, розроблена компанією Apple, прийшла на зміну застарілому Objective-C і на сьогодні є індустріальним стандартом для розробки під екосистему iOS.

Вибір Swift аргументується кількома фундаментальними перевагами, що детально викладені в офіційній технічній документації розробника[40]:

1. Безпека типів (Type Safety) та обробка помилок: Swift є строго типізованою мовою. Механізм опціоналів (Optionals) дозволяє уникнути критичних помилок (так званих null pointer exceptions) ще на етапі компіляції. Для додатку, який працює з медичними та нутриціологічними даними (де відсутність значення калорій чи білків може призвести до неправильного розрахунку норми), така безпека є критичною.

2. Продуктивність та керування пам'яттю: Swift використовує систему автоматичного підрахунку посилань (ARC - Automatic Reference Counting), що звільняє розробника від ручного управління пам'яттю, але працює значно швидше, ніж системи «збирання сміття» (Garbage Collection) в мовах типу Java чи C#.

3. Асинхронність (Concurrency): Для реалізації одночасного пошуку продуктів у локальній базі даних та через глобальні API без зависання інтерфейсу користувача необхідно використовувати багатопотоковість. Сучасний синтаксис Swift пропонує механізм async/await та Task, що дозволяє писати асинхронний код прозоро, надійно і без складних вкладених замикань (callbacks).

Довгий час стандартом для побудови інтерфейсів в iOS був імперативний фреймворк UIKit. Проте для створення FitFood було ухвалено рішення використати новітній декларативний фреймворк SwiftUI, технічні можливості якого ідеально підходять для нашої архітектури[38]. Перехід від імперативного фреймворку UIKit до декларативного SwiftUI розглядається в сучасній індустрії як найважливіша еволюційна зміна, що дозволяє створювати адаптивний інтерфейс із суттєво меншою кількістю коду, перекладаючи управління станами на операційну

систему[39]. Це забезпечує вищу продуктивність та стабільність роботи інформаційної системи.

Декларативний підхід кардинально змінює парадигму розробки: замість того, щоб писати детальні інструкції про те, як змінити інтерфейс (імперативний підхід), розробник описує, як повинен виглядати інтерфейс за певного стану даних. SwiftUI тісно пов'язаний з реактивним програмуванням. Коли користувач додатку FitFood додає до свого щоденника з'їдений продукт (наприклад, порцію хліба), локальний стан додатку (залишок калорій) змінюється. Завдяки механізмам `@State` та `@Published`, SwiftUI автоматично перемальовує всі залежні компоненти на екрані без необхідності писати додатковий код для оновлення[8].

Використання SwiftUI дозволило максимально ефективно реалізувати складні візуальні вимоги до додатку FitFood, описані в концепції:

- **Кругові діаграми (Ring Charts):** Візуалізація залишку калорій та балансу макронутрієнтів вимагає складних графічних елементів. У SwiftUI ця задача вирішується за допомогою вбудованого геометричного компонента `Circle()`. Використання модифікатора `trim(from:to:)` дозволяє динамічно відображати лише ту частину кола, яка відповідає відсотку спожитих калорій відносно денної норми користувача.
- **Прогрес-індикатори:** Для деталізації спожитих білків, жирів та вуглеводів оптимально підходить нативний компонент `ProgressView`. Його кастомізація дозволяє додати назву нутрієнта, поточне значення, денну норму та кольорову шкалу прогресу, що значно знижує когнітивне навантаження при сприйнятті інформації.
- **Система мікроінтеракцій та анімацій:** Згідно з дослідженнями в галузі mHealth[21], плавні анімації підвищують задоволеність користувачів. SwiftUI надає потужні, але прості інструменти для анімування змін стану. У додатку FitFood плавне заповнення кругової діаграми та поступове збільшення числових значень макронутрієнтів реалізовано за допомогою функції `withAnimation` та модифікатора `.animation(easeInOut)`. Це дозволяє зробити інтерфейс «живим», показуючи користувачеві наслідки додавання кожного продукту.

- Навігаційна система: Для організації модульної структури додатку (щоденник, пошук, статистика, налаштування) використовується компонент TabView, який забезпечує зручне перемикання між екранами звичною для користувачів iOS панеллю TabBar.

Таким чином, комбінація мови Swift та декларативного фреймворку SwiftUI утворює потужний інструментарій, що гарантує високу швидкість розробки, відмовостійкість коду та створення сучасного, чуйного та естетично привабливого користувацького інтерфейсу. Однак, окрім візуальної частини, критичним аспектом є вибір технологій для збереження та обробки великих масивів даних (продуктів), що є основою архітектури «Offline-first»[21].

Сучасні мобільні застосунки у сфері mHealth вимагають побудови складної, багаторівневої архітектури роботи з даними. Успішність додатку для контролю харчування залежить не лише від зручного інтерфейсу, але й від швидкості пошуку продуктів, точності нутриціологічної інформації та здатності системи працювати в умовах нестабільного інтернет-з'єднання. Саме тому для створення додатку FitFood було обрано гібридну архітектуру, що об'єднує локальне збереження даних, хмарну синхронізацію та інтеграцію із зовнішніми відкритими програмними інтерфейсами (API).

Фундаментальною вимогою до розроблюваного програмного продукту є забезпечення його безперебійної роботи навіть за відсутності доступу до мережі Інтернет. Більшість існуючих аналогів на ринку мають суттєвий недолік - вони блокують можливість внесення записів до харчового щоденника в офлайн-режимі. Для вирішення цієї проблеми в основу архітектури FitFood було покладено концепцію «Offline-first».

В якості основного інструменту для локального зберігання інформації було обрано реляційну систему управління базами даних SQLite. На відміну від великих клієнт-серверних рішень, SQLite є вбудованою (embedded) базою даних, яка зберігає весь масив інформації у єдиному файлі (foods.db) безпосередньо на iOS-пристрої користувача. Вибір SQLite обґрунтовується її винятковою надійністю,

швидкістю виконання SQL-запитів на мобільних процесорах та низьким споживанням оперативної пам'яті.

Локальна база даних розроблена таким чином, щоб зберігати критично важливу інформацію про кожен продукт: унікальний ідентифікатор, назву продукту, загальну калорійність, а також точний вміст макронутрієнтів (білків, жирів та вуглеводів). Це дозволяє користувачеві не лише шукати продукти без інтернету, але й створювати власні унікальні записи, формуючи персоналізовану базу даних їжі, яка миттєво доступна для додавання у щоденник.

Незважаючи на переваги локального сховища, жодна вбудована база даних не здатна вмістити інформацію про всі існуючі харчові продукти у світі. Для забезпечення конкурентоспроможності додатку FitFood було прийнято рішення інтегрувати його з глобальними нутриціологічними платформами за допомогою технології REST API.

Основним зовнішнім джерелом даних виступає FatSecret API. Це одна з найбільших у світі міжнародних бібліотек, яка надає верифіковані дані про харчову цінність продуктів, страв із ресторанів та товарів різних брендів. Інтеграція з FatSecret дозволяє розширити можливості пошуку додатку до глобального рівня, гарантуючи, що користувач зможе знайти необхідний продукт, перебуваючи в будь-якій країні світу.

Другим важливим компонентом є інтеграція з OpenFoodFacts API[33]. Ця відкрита, краудсорсингова база даних спеціалізується на зборі інформації про товари масового споживання та їх штрих-коди. Вибір саме цього API зумовлений необхідністю реалізації функції безкоштовного сканера штрих-кодів. Завдяки використанню нативного компонента DataScannerViewController від Apple, додаток FitFood зчитує штрих-код через камеру пристрою. Зчитаний код автоматично передається як параметр у GET-запит до серверів OpenFoodFacts, після чого користувач миттєво отримує деталізовану інформацію про знайдений продукт.

Технічна реалізація взаємодії з цими API побудована на новітніх механізмах асинхронного програмування мови Swift - `async/await` та використанні `Task`. Такий

підхід гарантує, що мережеві запити виконуються у фонових потоках (Background Threads), не блокуючи головний потік інтерфейсу користувача (Main Thread), що забезпечує безперебійність та плавність роботи додатку.

В епоху використання кількох пристроїв (наприклад, iPhone та iPad) критично важливою є можливість збереження прогресу та міграції даних користувача. Збереження даних виключно локально (у SQLite) створює ризик повної втрати інформації у випадку втрати або пошкодження пристрою. Для вирішення цієї архітектурної проблеми було обрано платформу Firebase від компанії Google, а саме її хмарну NoSQL-базу даних Cloud Firestore[25].

Firebase Cloud Firestore - це гнучка, масштабована база даних, призначена для мобільних додатків, яка підтримує синхронізацію в режимі реального часу. Вибір Firestore обґрунтовується її здатністю автоматично вирішувати конфлікти даних та підтримувати кешування запитів: якщо пристрій втрачає з'єднання з мережею, Firestore зберігає локальні зміни і автоматично відправляє їх на сервер одразу після відновлення зв'язку.

У додатку FitFood реалізовано алгоритм подвійного збереження: коли користувач створює новий власний продукт, інформація спочатку записується до локальної бази даних SQLite (забезпечуючи миттєвий відгук інтерфейсу), а потім у фоновому режимі синхронізується з хмарним сховищем Firebase. Такий підхід гарантує резервне копіювання інформації та можливість відновлення всього харчового щоденника на новому пристрої після авторизації.

Крім того, під час розробки структури баз даних (як локальної, так і хмарної) була закладена можливість збереження назв продуктів кількома мовами (зокрема, українською та англійською). Це дозволяє додатку обробляти мультязичні пошукові запити та пропонувати релевантні результати незалежно від мови системи смартфона користувача.

Проведений аналіз предметної області та технічного інструментарію довів, що створення мобільного додатку для контролю харчування є актуальним, але складним інженерним завданням. Дослідження існуючих аналогів на ринку

виявило низку недоліків, серед яких: залежність від постійного інтернет-з'єднання, ускладнений інтерфейс та комерціалізація базових функцій.

Для розв'язання цих проблем було створено концепцію та обрано оптимальний стек технологій. Використання мови Swift та декларативного фреймворку SwiftUI дозволяє створити безпечний та візуально привабливий продукт з анімованими мікроінтеракціями. Впровадження гібридної бази даних, яка складається з локального сховища SQLite та хмарної інфраструктури Firebase Cloud Firestore, забезпечує ідеальний баланс між автономністю (Offline-first) та надійним резервним копіюванням даних. Інтеграція відкритих API (FatSecret, OpenFoodFacts) розширює функціональні можливості додатку, перетворюючи його на глобальний інструмент пошуку харчових продуктів. Такий комплексний технологічний підхід створює міцний фундамент для подальшого проектування та програмної реалізації додатку FitFood, опис яких буде наведено у наступному розділі кваліфікаційної роботи.

РОЗДІЛ 2. ТЕХНОЛОГІЯ ПРОЄКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ НА IOS

2.1 Дизайн та етапи створення користувацького інтерфейсу

Процес розробки програмного забезпечення, зокрема мобільного додатку, є системним набором складних та передбачуваних кроків, які повинні бути виконані для успішного завершення проєкту. Цей процес забезпечує узгодженість ресурсів розробки та гарантує, що всі аспекти проєкту розглядаються та виконуються вчасно. Створення сучасного цифрового продукту складається з двох основних етапів: дизайну (проєктування користувацького досвіду та інтерфейсу) та безпосередньої програмної розробки.

У межах даної кваліфікаційної роботи було створено мобільний додаток FitFood, основним призначенням якого є підрахунок калорійності продуктів, детальний контроль харчування та зручне ведення щоденника спожитої їжі. Створення здійснювалося ексклюзивно для платформи iOS з використанням сучасної мови програмування Swift та новітнього декларативного фреймворку SwiftUI.

На етапі проєктування дизайну (UI/UX) було визначено, що додаток повинен мати інтуїтивно зрозумілу архітектуру, щоб знизити когнітивне навантаження на користувача. Для розробки та тестування візуальних прототипів інтерфейсу було використано інструментарій Figma[24]. Завдяки цьому додаток дозволяє користувачам швидко знаходити потрібні продукти, переглядати їх точну харчову цінність, додавати їх до щоденного раціону та відстежувати життєво важливий баланс білків, жирів та вуглеводів.

Згідно з сучасними методологіями проєктування, ефективний мобільний інтерфейс має базуватися на принципах орієнтованості на користувача (User-Centered Design), що забезпечує логічність навігації та мінімізацію кроків для виконання цільової дії[18].

Дизайн інтерфейсу у сфері mHealth має унікальні виклики, оскільки він повинен поєднувати високу інформативність із граничною простотою. Поведінкова

наукова школа стверджує, що низька якість UX у медичних технологіях може призвести до викривлення аналітики через помилки введення даних користувачем[26]. Саме тому в застосунку FitFood було впроваджено принципи інтуїтивного дизайну, що відповідають гайдлайнам Apple Human Interface Guidelines[28] та спрямовані на підвищення задоволеності користувачів через зручність та доступність цифрового продукту[20, 4].

Для забезпечення зручної навігації архітектура додатку була розділена на кілька основних логічних модулів:

- головний екран щоденника харчування;
- модуль пошуку продуктів та сканування;
- екран детальної статистики споживання;
- розділ налаштувань користувача;
- модуль управління базою даних продуктів;
- фонові системи синхронізації даних.

Навігація між цими основними екранами здійснюється за допомогою стандартного системного компонента iOS - панелі TabBar, яка розташована в нижній частині екрана та містить іконки основних розділів додатку, забезпечуючи швидкий доступ до будь-якої функції в один клік.

Головний екран щоденника (Dashboard) є центральним елементом взаємодії користувача з програмою. Інтерфейс додатку було реалізовано за допомогою SwiftUI, що дозволило створити сучасний, адаптивний та зручний дизайн, який автоматично підлаштовується під розміри різних пристроїв Apple.

Вибір фреймворку SwiftUI для створення інтерфейсу FitFood дозволив реалізувати концепцію реактивного програмування. Це означає, що інтерфейс додатку є прямим відображенням стану даних (State). Наприклад, коли користувач вводить вагу продукту, змінна @State автоматично оновлює всі текстові поля та діаграми на екрані без необхідності ручного перевантаження сторінки. Такий підхід не лише прискорює роботу програми, а й дозволяє створювати складні анімації з мінімальним використанням системних ресурсів iPhone. Декларативний синтаксис фреймворку дозволяє відокремити опис зовнішнього вигляду від бізнес-

логіки, що значно полегшує подальшу підтримку та масштабування програмного продукту.

Особливу увагу приділено естетичним аспектам та візуальній чистоті інтерфейсу, що відповідає сучасним вимогам до графічного оформлення мобільних систем та сприяє цифровій трансформації сервісів охорони здоров'я, покращуючи якість життя населення[7, 3]. При проектуванні колірної палітри було обрано темну тему (Dark Mode) з яскравими акцентними кольорами. Крім того, імплементація Dark Mode та чіткої типографіки забезпечує інклюзивність та доступність сервісу для широкої аудиторії згідно з національними стандартами доступності ІКТ-продуктів[26, 5].

У верхній частині головного інтерфейсу реалізовано ключовий інформаційний блок, який у режимі реального часу показує кількість калорій, що залишилась до вичерпання денної норми користувача. Для максимально ефективної та швидкої візуалізації цих даних використовується кругова діаграма (ring chart), що динамічно заповнюється за годинниковою стрілкою. Приклад візуалізації динамічного оновлення цього блока при зміні даних наведено на рис.

2.1

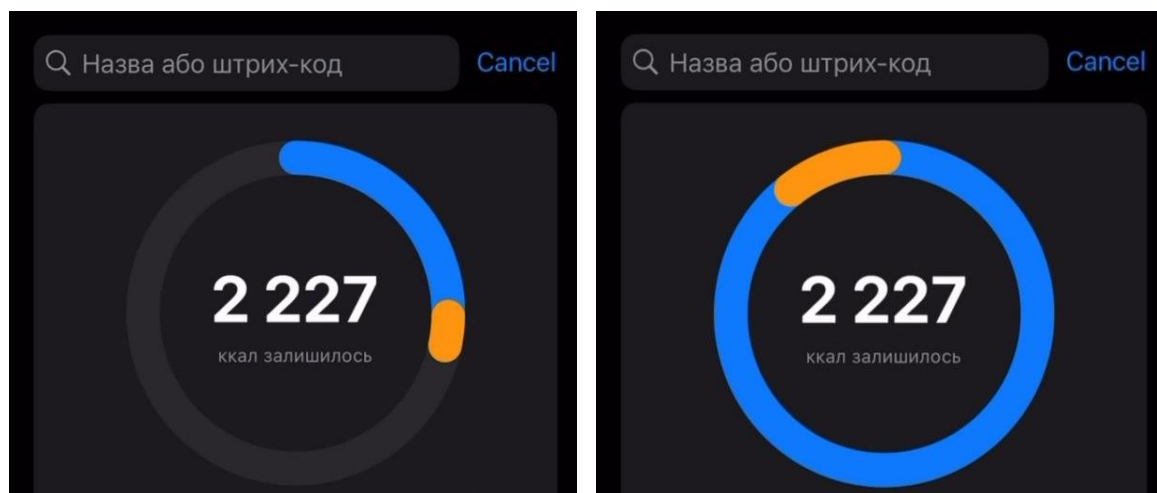


Рис. 2.1 Динамічне оновлення інформації про залишок калорій та стан кругової діаграми

Застосування UUID як первинного ключа в цій таблиці є стратегічним рішенням для майбутньої синхронізації. Це дозволяє уникнути конфліктів

ідентифікаторів при об'єднанні локальних даних користувача з глобальною базою у Firebase. Поле `is_custom` дає змогу програмі чітко розрізняти верифіковані продукти та записи, створені користувачем вручну, що є основою для подальшої реалізації краудсорсингової моделі наповнення бази.

З технічної точки зору, діаграма реалізована за допомогою базового графічного компонента `Circle()`, до якого застосовується математична функція `trim()`. Ця функція дозволяє динамічно відображати лише певну частину кола (від 0.0 до 1.0), яка обчислюється як відношення вже спожитих калорій до загальної денної цілі. Кожен кольоровий сегмент цього кола відповідає за конкретний макронутрієнт: білки, жири та вуглеводи. Такий підхід дозволяє користувачеві буквально за одну секунду оцінити свій поточний прогрес.

Окрім загальної калорійності, критично важливим для цільової аудиторії додатку (спортсменів та людей, що слідкують за вагою) є співвідношення макронутрієнтів. Тому безпосередньо під круговою діаграмою у додатку реалізовано детальне відображення кількості спожитих білків, жирів та вуглеводів. Для цього використовуються лінійні прогрес-індикатори, побудовані на базі компонента `ProgressView`.

Щоб зробити інформацію максимально вичерпною, кожен такий показник на екрані містить:

- текстову назву макронутрієнта;
- поточне числове значення у грамах;
- встановлену денну норму користувача;
- кольорову лінійну шкалу прогресу, яка заповнюється зліва направо.

Процес динамічного заповнення лінійних індикаторів макронутрієнтів під час використання додатка продемонстровано на рис. 2.2.

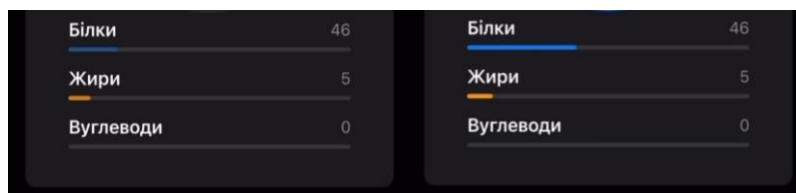


Рис. 2.2 Динамічне заповнення лінійних прогрес-індикаторів макронутрієнтів

Сучасний дизайн мобільних додатків вимагає не лише статичної краси, але й динамічного відгуку на дії користувача. Для суттєвого покращення користувацького досвіду (UX) в додатку FitFood було реалізовано систему анімацій. Зокрема, при додаванні нового продукту до раціону відбувається:

- плавне графічне заповнення кругової діаграми;
- плавне динамічне збільшення числових значень макронутрієнтів;
- анімація заповнення лінійних прогрес-індикаторів.

Програмно ці анімації реалізовані за допомогою вбудованих інструментів SwiftUI, таких як блок `withAnimation` та модифікатор `.animation(.easeInOut)`. Використання кривої Безьє «ease-in-out» робить рух елементів природним (швидкість плавно наростає на початку і плавно сповільнюється в кінці), що робить інтерфейс значно більш сучасним, «живим» та зручним для користувача. Окрім цього, для надання інтерфейсу глибини, у дизайні додатку активно використано картки з ефектом напівпрозорого розмиття (Blur effect).

Окрему увагу під час проєктування дизайну було приділено так званим «порожнім станам» (Empty States). Наприклад, коли на початку дня список спожитої їжі порожній, замість простого білого екрана відображається спеціальний інформаційний блок з тематичною іконкою (мискою салату) та повідомленням «Ваш щоденник порожній. Додайте їжу через пошук». Це є відомим патерном UX-дизайну, який підказує користувачеві його наступний логічний крок і не залишає його розгубленим.

Після розробки архітектури головного екрана (Dashboard) наступним і найважливішим етапом проєктування користувацького інтерфейсу (UI) та досвіду (UX) стала створення модулів взаємодії з даними. У додатках категорії mHealth критичним фактором утримання користувача є швидкість та зручність внесення щоденних записів. Якщо процес пошуку та додавання їжі займає багато часу або вимагає зайвих кліків, користувач швидко втрачає мотивацію до ведення харчового щоденника.

Процес пошуку продуктів у додатку FitFood спроектовано з акцентом на мінімізацію когнітивного навантаження. Користувач може знайти продукт за

назвою або штрих-кодом. Для цього використовується спеціальний пошуковий рядок, інтегрований безпосередньо в інтерфейс щоденника.

При введенні тексту в поле пошуку запускається складний асинхронний процес. Щоб не блокувати роботу інтерфейсу під час завантаження інформації з баз даних (локальної SQLite, FatSecret та OpenFoodFacts), пошук реалізовано з використанням асинхронних задач Swift (Task). Результати відображаються у вигляді динамічного списку під рядком пошуку. Кожен елемент списку являє собою картку (Cell), яка містить назву продукту та його базову харчову цінність (кількість калорій на 100 грамів), що дозволяє користувачеві швидко ідентифікувати потрібний варіант.

Для безпосереднього додавання знайденого продукту до свого щоденного раціону користувачеві достатньо натиснути на кнопку з іконкою «плюс». У цей момент, згідно з найкращими практиками Apple Human Interface Guidelines, система не перекидає користувача на новий екран, а викликає модальне вікно введення ваги порції. Дизайн цього модального вікна (з текстовим запитом «Скільки грамів продукту ви з'їли?») заслуговує на окрему увагу. Рішення про використання кастомної цифрової клавіатури (Numpad) замість стандартної QWERTY-клавіатури iOS базується на законі Фіттса (Fitts's Law). Це дозволяє реалізувати стратегію запобігання помилкам (Error Prevention), усуваючи ризик введення літер замість числових значень ваги[26, 20]. Кастомна клавіатура містить великі, зручні для натискання кнопки від 0 до 9, що дозволяє комфортно користуватися додатком однією рукою. Інтерфейс модального вікна зі створеною кастомною цифровою клавіатурою представлено на рис. 2.3.

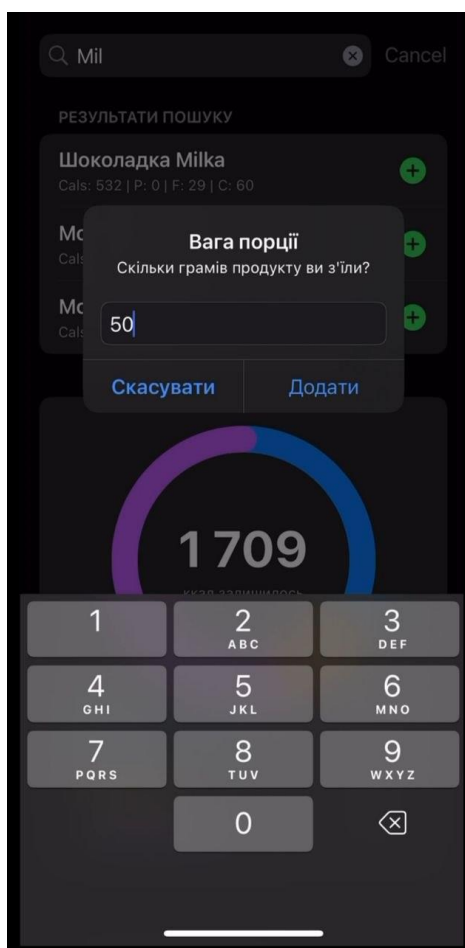


Рис. 2.3 Модальне вікно введення ваги порції з кастоною клавіатурою

Після того, як користувач вводить кількість грамів та натискає кнопку підтвердження («Додати»), вікно автоматично закривається, і в реальному часі відбувається кілька процесів: продукт додається до щоденника, система перераховує калорії відповідно до вказаної ваги та миттєво оновлює показники макронутрієнтів (білків, жирів, вуглеводів) на головних кругових діаграмах. Проектування форми створення власного (користувацького) продукту. Незважаючи на наявність інтеграції з глобальними базами даних, у реальному житті часто виникають ситуації, коли продукт або специфічна домашня страва відсутні у пошуковій видачі. Якщо продукт не знайдено, користувач має можливість додати його вручну. Інтерфейс екрана пошуку з пропозицією створити власний продукт наведено на рис. 2.4.

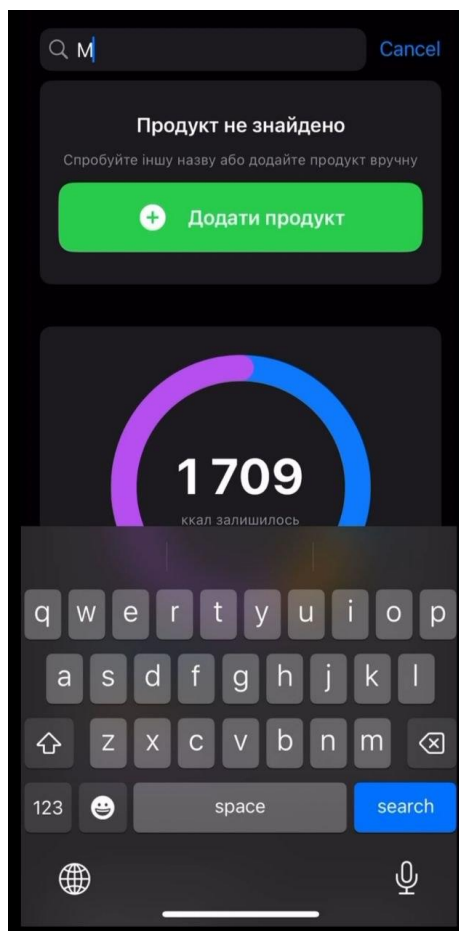


Рис. 2.4 Екран відсутності результатів пошуку та перехід до створення продукту

Для цього було створено окрему форму створення власного продукту. З погляду UX-дизайну, форма розділена на логічні блоки для послідовного заповнення. Форма додавання містить такі обов'язкові поля:

- назва продукту (текстове поле, Text Field);
- калорії (числове поле);
- білки (числове поле);
- жири (числове поле);
- вуглеводи (числове поле).

Кожне числове поле оснащено валідацією на стороні клієнта: система не дозволить зберегти продукт, якщо введено від'ємні значення або залишено порожні обов'язкові рядки. Процес заповнення форми створення власного продукту продемонстровано на рис. 2.5.

Рис. 2.5 Форма створення нового (користувачького) харчового продукту

Після перевірки валідності даних та натискання кнопки збереження, продукт записується у локальну базу даних SQLite, а згодом - автоматично синхронізується з хмарним сховищем, стаючи доступним у глобальному пошуку для цього користувача.

Останнім важливим модулем користувацького інтерфейсу є розділ налаштувань, який надає користувачеві контроль над роботою самого додатку. У цьому розділі користувач може гнучко керувати джерелами інформації, зокрема увімкнути або вимкнути використання різних зовнішніх баз даних (наприклад, тимчасово вимкнути запити до FatSecret для економії інтернет-трафіку або прискорення локального пошуку).

Також в інтерфейсі налаштувань передбачена критично важлива функція управління приватністю та пам'яттю - можливість скинути всі збережені дані

(очищення щоденника та локальної бази), а також переглянути юридичну інформацію про додаток (версія, ліцензія).

Сучасні цифрові продукти не існують в ізоляції, тому для підтримки зворотного зв'язку з аудиторією та формування ком'юніті, в архітектуру додатку було додано блок соціальних мереж. Цей візуальний блок містить стилізовані іконки, які дозволяють користувачеві одним кліком перейти до сторінок розробника на популярних платформах. Підтримуються переходи на YouTube, інтеграція з месенджером Telegram для швидкої технічної підтримки, та посилання на Instagram. Таке рішення значно підвищує рівень довіри до продукту та створює прямий канал комунікації між розробником (автором кваліфікаційної роботи) та кінцевими споживачами. Візуальну реалізацію розділу налаштувань додатку та блоку соціальних мереж наведено на рис. 2.6.

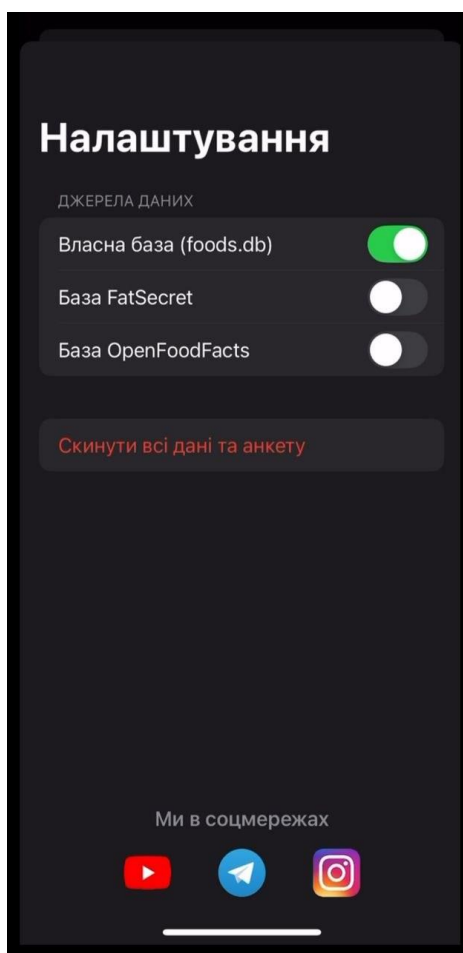


Рис. 2.6 Розділ налаштувань додатку та інтеграція соціальних мереж

Підсумовуючи вищезазначене, можна констатувати, що дизайн та користувацький інтерфейс додатку FitFood спроектовано з урахуванням сучасних ергономічних вимог. Асинхронний пошук, зручне модальне вікно з цифровою клавіатурою для введення ваги, інтуїтивна форма створення власних продуктів та прозорий розділ налаштувань створюють цілісний, комфортний та ефективний інструмент для щоденного моніторингу харчування.

2.2 Архітектура та проєктування баз даних мобільного додатку FitFood

Основою будь-якого сучасного інформаційного забезпечення, зокрема у сфері мобільної охорони здоров'я (mHealth), є надійна, оптимізована та безпечна архітектура баз даних. Від правильності проєктування структур даних залежить швидкість відгуку інтерфейсу, коректність математичних розрахунків макронутрієнтів та загальна стабільність роботи додатку. Зважаючи на архітектурний підхід «Offline-first», обраний для мобільного додатку FitFood, процес проєктування було розділено на два ключові етапи: створення локальної реляційної схеми (SQLite) та моделювання хмарної NoSQL структури (Firebase).

У цій частині дослідження детально розглядається перший етап - проєктування локального сховища даних.

Для забезпечення стабільної роботи додатка FitFood у режимі офлайн та надійного збереження даних було обрано реляційну СУБД SQLite. Вибір цієї технології обґрунтовується її архітектурною специфікою: вона не потребує серверного процесу та зберігає всі дані у вигляді єдиного кросплатформного файлу (foods.db) безпосередньо у файловій системі пристрою[37]. Реалізація локального сховища з використанням Swift-обгортки дозволила інтегрувати SQLite безпосередньо в декларативну архітектуру SwiftUI, забезпечуючи високу швидкість обробки запитів та реактивне оновлення інтерфейсу при зміні стану даних[37].

Під час проєктування схеми локальної бази даних було застосовано принципи нормалізації, що дозволило уникнути дублювання інформації та забезпечити цілісність даних[36]. Концептуальна модель даних додатку FitFood

складається з трьох основних взаємопов'язаних таблиць: Products (Каталог продуктів), DailyLogs (Щоденні записи) та LogItems (Спожиті порції).

Таблиця Products є ядром локального сховища. Вона призначена для збереження довідкової інформації про харчову цінність інгредієнтів та страв. Детальний опис структури цієї таблиці наведено у Таблиці 2.1.

Таблиця 2.1

Структура бази даних: таблиця Products (Каталог продуктів)

Назва поля	Тип даних	Обмеження	Опис призначення поля
id	TEXT / UUID	Primary Key	Унікальний ідентифікатор продукту. Використання UUID дозволяє безконфліктно синхронізувати ці записи з хмарною базою даних.
name	TEXT	Not Null	Найменування харчового продукту (наприклад, «Хліб білий»).
calories	REAL	Not Null	Енергетична цінність продукту з розрахунку на 100 грамів.
proteins	REAL	Not Null	Вміст білків у грамах на 100 грамів продукту.
fats	REAL	Not Null	Вміст жирів у грамах на 100 грамів продукту.
carbs	REAL	Not Null	Вміст вуглеводів у грамах на 100 грамів продукту.
barcode	TEXT	Nullable	Штрих-код продукту (EAN-13 або UPC). Поле заповнюється для магазинних товарів для швидкого пошуку через сканер.
is_custom	INTEGER	Boolean	Прапорець (1 або 0), який вказує, чи був цей продукт створений користувачем вручну, чи завантажений з глобальної бази.

Джерело: створено автором

Таблиця DailyLogs слугує заголовком для конкретного дня (Таблиця 2.2), а таблиця LogItems містить деталізацію того, що саме користувач з'їв (Таблиця 2.3).

Таблиця 2.2

Структура бази даних: таблиця DailyLogs (Щоденні записи)

Назва поля	Тип даних	Обмеження	Опис призначення поля
log_id	TEXT / UUID	Primary Key	Унікальний ідентифікатор запису конкретного дня.
date	TEXT	Unique	Дата запису у форматі ISO8601 («YYYY-MM-DD»). Унікальність гарантує один запис-контейнер на один календарний день.
total_calories	REAL	Nullable	Кешована сума всіх спожитих калорій за день для швидкого відображення на головному екрані додатку.

Джерело: створено автором

Унікальність поля date у цій таблиці гарантує, що для кожної календарної дати в системі існує лише один запис-контейнер. Це значно спрощує логіку вибірки даних для головного екрана та побудови графіків статистичного аналізу. Використання формату ISO8601 забезпечує правильне програмне сортування записів за часом незалежно від локальних налаштувань iPhone, а кешування суми калорій (total_calories) безпосередньо в цій таблиці дозволяє інтерфейсу завантажуватися миттєво. Програмі не потрібно кожного разу перераховувати всі порції з нуля при відкритті застосунку, що позитивно впливає на швидкість відгуку системи.

Для деталізації спожитих продуктів у межах конкретної дати використовується пов'язана таблиця LogItems (Таблиця 2.3), яка дозволяє зберігати необмежену кількість записів про їжу для кожного заголовка дня.

Таблиця 2.3

Структура бази даних: таблиця LogItems (Спожиті порції)

Назва поля	Тип даних	Обмеження	Опис призначення поля
item_id	TEXT / UUID	Primary Key	Унікальний ідентифікатор конкретної з'їденої порції.

Продовження табл. 2.3

log_id	TEXT	Foreign Key	Посилання на конкретний день з таблиці DailyLogs. Налаштовано каскадне видалення (ON DELETE CASCADE).
product_id	TEXT	Foreign Key	Посилання на довідник продуктів у таблиці Products. Запобігає дублюванню назв продуктів у щоденнику.
weight_grams	REAL	Not Null	Фактична вага порції, яку ввів користувач через модальне вікно (наприклад, 150 грамів).

Джерело: створено автором

Важливо зазначити, що зв'язок між таблицями DailyLogs та LogItems реалізовано за принципом «один до багатьох». Це ключове архітектурне рішення для створення додатку FitFood, оскільки воно дозволяє одному календарному дню відповідати будь-якій кількості спожитих порцій. Використання product_id як зовнішнього ключа є критично важливим для оптимізації пам'яті: система не дублює інформацію про нутрієнти щоразу, а просто звертається до довідника продуктів. Це значно підвищує продуктивність бази даних при тривалому веденні щоденника протягом місяців або років.

Завдяки такій реляційній структурі, при додаванні продукту у щоденник, система FitFood створює новий запис у LogItems. Далі, за допомогою SQL-запиту з оператором JOIN, додаток об'єднує дані про вагу порції з базовою харчовою цінністю. Математичний розрахунок виконується миттєво, після чого оновлені дані передаються у SwiftUI для перемальовування інтерфейсу.

Спроектowana локальна схема бази даних SQLite повністю покриває потреби автономної роботи додатку (Offline-first) згідно з принципами нативної розробки[37]. При цьому структура таблиць була спроектована таким чином, щоб забезпечити легку трансформацію даних у формат JSON/Dictionary, що є необхідним для подальшої синхронізації локального сховища з хмарною інфраструктурою Firebase.

Для адміністрування та перевірки коректності роботи локального сховища на етапі створення використовувалося спеціалізоване програмне забезпечення DB Browser for SQLite. Фрагмент заповненої таблиці foods, що містить дані про продукти (зокрема мультімовні назви та розраховані макронутрієнти), які обробляються безпосередньо на пристрої користувача, наведено на рис. 2.7.

	id	name_en	name_ru	name_uk	calories	protein	fat	carbs
1	1	Big Mac	Биг Мак	Биг Мак	267.0	12.6	14.9	20.1
2	2	Cheeseburger	Чизбургер	Чизбургер	263.0	12.5	12.8	24.0
3	3	Hamburger	Гамбургер	Гамбургер	260.0	12.0	9.0	31.0
4	4	McChicken	МакЧикен	МакЧикен	273.0	10.0	13.0	29.0
5	5	Chicken Nuggets 6	Нагетси 6	Нагетси 6	261.0	15.0	16.0	14.0
6	6	French Fries Small	Картофель фри маленький	Картопля фри мала	237.0	3.0	11.0	30.0
7	7	French Fries Medium	Картофель фри средний	Картопля фри середня	337.0	4.0	16.0	44.0
8	8	French Fries Large	Картофель фри большой	Картопля фри велика	444.0	6.0	22.0	68.0
9	9	Coca Cola	Кока Кола	Кока Кола	170.0	0.0	0.0	42.0
10	10	Latte	Латте	Латте	190.0	10.0	7.0	19.0
11	11	Apple Pie	Яблочный пирог	Яблучний пиріг	243.0	2.0	11.0	34.0
12	12	Apple	Яблоко	Яблуко	62.0	0.3	0.2	14.0
13	13	Banana	Банан	Банан	96.0	1.3	0.3	27.0
14	14	Orange	Апельсин	Апельсин	47.0	0.9	0.1	11.8
15	15	Pear	Груша	Груша	67.0	0.4	0.1	16.0
16	16	Grape	Виноград	Виноград	69.0	0.7	0.2	18.1
17	17	Strawberry	Клубника	Полум'я	38.0	0.7	0.3	7.7
18	18	Blueberry	Черника	Ложика	57.0	0.7	0.3	14.5
19	19	Watermelon	Арбуз	Кавун	30.0	0.6	0.2	7.6
20	20	Melon	Дыня	Дяня	34.0	0.8	0.2	8.2
21	21	Kiwi	Киви	Ківі	61.0	1.1	0.5	14.7
22	22	Tomato	Помидор	Помідор	18.0	0.9	0.2	3.9
23	23	Cucumber	Огурец	Огірок	15.0	0.7	0.1	3.6

Рис. 2.7 Структура та наповнення локальної бази даних у DB Browser for SQLite

Після проєктування локальної реляційної схеми SQLite, яка забезпечує базову автономність мобільного додатку, наступним етапом розробки стало проєктування хмарної інфраструктури та інтеграції із зовнішніми сервісами. Це дозволило перетворити FitFood зі звичайного локального щоденника на потужну краудсорсингову платформу з глобальним пошуком.

Концептуальною особливістю додатку FitFood є децентралізований підхід до наповнення бази даних харчових продуктів. Замість того, щоб покладатися виключно на заздалегідь підготовлені списки, у системі реалізовано механізм краудсорсингу. Якщо користувач додає до щоденника унікальну страву (наприклад, з меню місцевого закладу харчування), якої немає в жодній існуючій базі, ця страва зберігається не лише в його локальній базі SQLite, але й автоматично відправляється на хмарний сервер.

Для реалізації цього механізму було використано NoSQL документо-орієнтовану базу даних Firebase Cloud Firestore[25]. На відміну від реляційних таблиць, Firestore зберігає дані у вигляді колекцій та документів (у форматі, близькому до JSON), що забезпечує надзвичайно швидку синхронізацію в режимі реального часу.

Структуру хмарної колекції GlobalProducts, яка відповідає за обмін даними між усіма користувачами додатку, наведено у Таблиці 2.4.

Таблиця 2.4

Структура NoSQL колекції GlobalProducts (Firebase Cloud Firestore)

Ключ (Назва поля)	Тип даних (Firestore)	Опис призначення поля
document_id	String	Унікальний ідентифікатор документа (збігається з UUID з локальної бази SQLite для уникнення дублювання).
name	String	Назва доданого продукту або страви із закладу.
nutrition	Map (Об'єкт)	Вкладений об'єкт, що містить поля calories, proteins, fats, carbs (усі типу Number).
created_by	String	Анонімізований ідентифікатор пристрою користувача, який першим додав цей продукт у систему.
timestamp	Timestamp	Точний час додавання запису на сервер для синхронізації оновлень на інших клієнтських пристроях.

Джерело: створено автором

Завдяки такій архітектурі, як тільки один користувач створює новий продукт, Firestore через механізм веб-сокетів (WebSockets) надсилає оновлення всім іншим встановленим клієнтам FitFood. Таким чином, база даних постійно розширюється та актуалізується зусиллями самої спільноти користувачів.

Практичну реалізацію краудсорсингової моделі та збереження даних у хмарному середовищі можна спостерігати безпосередньо через консоль управління Firebase. На рис. 2.8 наведено структуру згенерованого документа в колекції products. Як видно з прикладу, продукт, доданий користувачем через інтерфейс

застосунку, успішно синхронізувався на сервері із збереженням усіх необхідних полів (калорії, БЖВ, дата створення).

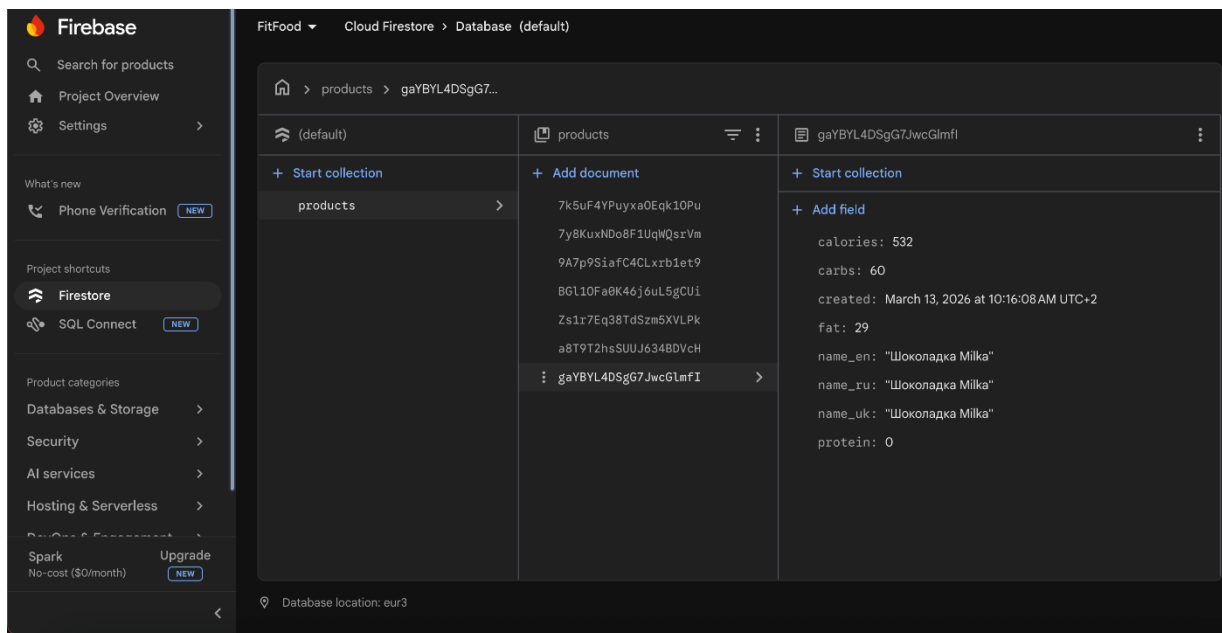


Рис. 2.8 Відображення синхронізованого продукту в консолі Firebase Cloud Firestore

Для забезпечення максимального покриття харчових продуктів, у додатку реалізовано паралельну роботу з двома найбільшими світовими базами даних через технологію REST API. Архітектура пошуку спроектована таким чином, щоб закрити різні сегменти ринку:

1. **FatSecret API:** Використовується як первинне джерело інформації для великої бази закордонних товарів (переважно американських та німецьких виробників), а також класичних міжнародних страв.
2. **OpenFoodFacts API:** Ця база даних містить понад один мільйон записів. Її критичною перевагою для вітчизняного користувача є наявність величезної кількості продуктів з українських супермаркетів. Саме до цього API звертається сканер штрих-кодів додатку FitFood[33]. Коли камера зчитує штрих-код, додаток формує GET-запит до серверів OpenFoodFacts і миттєво отримує детальний JSON-файл зі складом продукту, який програмно парситься та конвертується у локальну модель Swift.

Безпосередня програмна реалізація взаємодії з API OpenFoodFacts виконана в середовищі Xcode на мові Swift з використанням сучасного асинхронного підходу. Фрагмент програмного коду, що відповідає за формування URL-запиту за штрих-кодом (fetchOFFByBarcode), безпечний парсинг JSON-відповіді та мапінг нутрієнтів у внутрішню структуру даних додатку, наведено наведено нижче:

// MARK: - OpenFoodFacts

```
func fetchOFF(query: String) async ->[FSProduct] {
    let trimmed = query.trimmingCharacters(in: .whitespacesAndNewlines)
    guard !trimmed.isEmpty else { return[] }

    if CharacterSet.decimalDigits.isSuperset(of: CharacterSet(charactersIn: trimmed))
    {
        return await fetchOFFByBarcode(trimmed)
    } else {
        return await fetchOFFBySearch(trimmed)
    }
}

func fetchOFFByBarcode(_ barcode: String) async ->[FSProduct] {
    guard let url = URL(string:
"https://world.openfoodfacts.org/api/v2/product/^(barcode).json") else {
        return[]
    }

    do {
        let (data, _) = try await URLSession.shared.data(for: URLRequest(url: url))
        let json = try JSONSerialization.jsonObject(with: data) as?[String: Any]

        guard let product = json?["product"] as?[String: Any] else { return[] }
    }
```

```

let name =
    (product["product_name_uk"] as? String)?
        .trimmingCharacters(in: .whitespacesAndNewlines)
    ?? (product["product_name"] as? String)?
        .trimmingCharacters(in: .whitespacesAndNewlines)
    ?? ""

guard !name.isEmpty else { return[] }

let nutrients = product["nutriments"] as?[String: Any]
let cals = numericValue(from: nutrients?["energy-kcal_100g"])
let proteins = numericValue(from: nutrients?["proteins_100g"])
let fats = numericValue(from: nutrients?["fat_100g"])
let carbs = numericValue(from: nutrients?["carbohydrates_100g"])

guard cals > 0 else { return[] }

return[
    FSProduct(
        food_name: name,
        food_description: "Cals: \(Int(cals.rounded())) | P:
\(formatNumber(proteins)) | F: \(formatNumber(fats)) | C: \(formatNumber(carbs))",
        food_id: "off_barcode_\(barcode)"
    )
]
} catch {
    if (error as NSError).code != -999 {
        print("OFF barcode error: \(error)")
    }
}

```

```

        return[]
    }
}

func fetchOFFBySearch(_ query: String) async ->[FSProduct] {
    var comps = URLComponents(string:
"https://world.openfoodfacts.org/cgi/search.pl")!
    comps.queryItems = [
        URLQueryItem(name: "search_terms", value: query),
        URLQueryItem(name: "search_simple", value: "1"),
        URLQueryItem(name: "action", value: "process"),
        URLQueryItem(name: "json", value: "1"),
        URLQueryItem(name: "page_size", value: "10"),
        URLQueryItem(name: "fields", value:
"product_name_uk,product_name,nutriments,code")
    ]

    guard let url = comps.url else { return[] }

    do {
        let (data, _) = try await URLSession.shared.data(for: URLRequest(url: url))
        let json = try JSONSerialization.jsonObject(with: data) as?[String: Any]
        let products = json?["products"] as?[[String: Any]] ??[]

        return products.compactMap { item in
            let name =
                (item["product_name_uk"] as? String)?
                    .trimmingCharacters(in: .whitespacesAndNewlines)
                ?? (item["product_name"] as? String)?
                    .trimmingCharacters(in: .whitespacesAndNewlines)
                ?? ""

```

```

guard !name.isEmpty else { return nil }

let nutrients = item["nutriments"] as?[String: Any]
let cals = numericValue(from: nutrients?["energy-kcal_100g"])
let proteins = numericValue(from: nutrients?["proteins_100g"])
let fats = numericValue(from: nutrients?["fat_100g"])
let carbs = numericValue(from: nutrients?["carbohydrates_100g"])

guard cals > 0 else { return nil }

let code = item["code"] as? String ?? UUID().uuidString

return FSProduct(
    food_name: name,
    food_description: "Cals: \(Int(cals.rounded())) | P:
\(formatNumber(proteins)) | F: \(formatNumber(fats)) | C: \(formatNumber(carbs))",
    food_id: "off_\(code)"
)
}
} catch {
    if (error as NSError).code != -999 {
        print("OFF search error: \(error)")
    }
    return[]
}
}

```

Як видно з наведеного коду, алгоритм автоматично визначає тип запиту (штрих-код чи текст), після чого виконує мережевий запит `URLSession.shared.data у`

фоновому потоці. Особливу увагу приділено обробці помилок та перевірці наявності локалізованої української назви продукту (`product_name_uk`).

Виклик нативного сканера штрих-кодів у декларативному інтерфейсі SwiftUI реалізовано за допомогою системного компонента `DataScannerViewController`. Для забезпечення безперебійної роботи була написана логіка перевірки підтримки камери пристроєм (`isSupported`). Програмний код ініціалізації вікна сканера, обробки зчитаного коду та виклику модальних вікон наведено нижче:

```
.sheet(isPresented: $isScannerPresented) {
    if DataScannerViewController.isSupported {
        ScannerView { code in
            isScannerPresented = false
            searchText = code
        }
    } else {
        Text("Камера не підтримується")
        .padding()
    }
}

.sheet(isPresented: $showingSettings) {
    settingsView
}

.sheet(isPresented: $showingAddFood) {
    AddFoodView()
}

.alert("Бага порції", isPresented: $showingWeightAlert) {
    TextField("Грами", text: $weightInput)
    .keyboardType(.numberPad)

    Button("Додати", action: confirmAdd)
    Button("Скасувати", role: .cancel) { }
```

```

} message: {
    Text("Скільки грамів продукту ви з'їли?")
}

```

Як видно з наведеного лістингу, реалізація базується на використанні модифікаторів `.sheet` та `.alert`, що дозволяє органічно інтегрувати складні системні контролери (як-от камера) у декларативну ієрархію представлень SwiftUI. Використання типу клавіатури `.numberPad` для введення ваги порції забезпечує додатковий рівень валідації даних, обмежуючи користувача лише числовими значеннями.

Процес розпізнавання штрих-коду товару через камеру пристрою наведено на рис. 2.9.



Рис. 2.9 Робота інтегрованого сканера штрих-кодів

Оскільки одночасне використання локальної бази (SQLite), краудсорсингової хмарної бази (Firebase) та двох зовнішніх API може створювати надмірне навантаження на інтернет-трафік користувача, в архітектуру було закладено

модуль динамічної маршрутизації. У розділі налаштувань додатку користувач має можливість самостійно обирати активні бази даних. Система дозволяє використовувати як одну конкретну базу, так і будь-яку їх комбінацію (2 або 3 одночасно). Це реалізовано за допомогою логічних прапорців (Boolean flags), які зберігаються в пам'яті пристрою і перевіряються алгоритмом безпосередньо перед виконанням пошукового запиту. Інтерфейс модуля динамічної маршрутизації джерел даних наведено на рис.2.10.

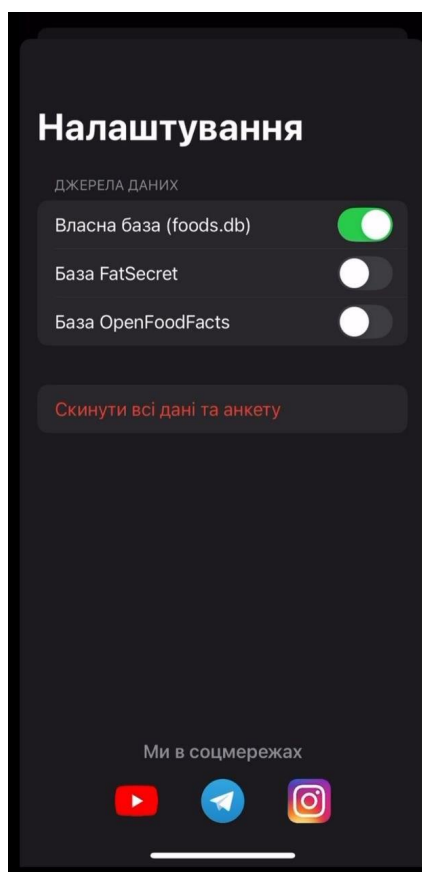


Рис. 2.10 Розділ налаштувань для керування активними базами даних

Персоналізація є невід'ємною складовою систем контролю здоров'я. Для того, щоб додаток виконував свою головну функцію - підрахунок цільової норми калорій - було спроектовано модуль первинного опитування (Onboarding). При першому запуску додатку FitFood користувач потрапляє на екран анкети, де система збирає базові антропометричні параметри (Таблиця 2.5). Етапи взаємодії користувача з інтерфейсом анкети продемонстровано на рис. 2.11.

Таблиця 2.5

Структура локальних даних профілю користувача (User Profile)

Параметр	Тип даних	Значення / Опис
gender	Enum	Стать (Чоловіча / Жіноча) для вибору правильних коефіцієнтів у формулі.
age	Integer	Вік користувача у повних роках.
Height_cm	Integer	Зріст користувача в сантиметрах.
Weight_kg	Double	Поточна вага користувача у кілограмах.
goal	Enum	Ціль: Схуднення (дефіцит), Підтримка (баланс), Набір маси (профіцит).

Джерело: створено автором

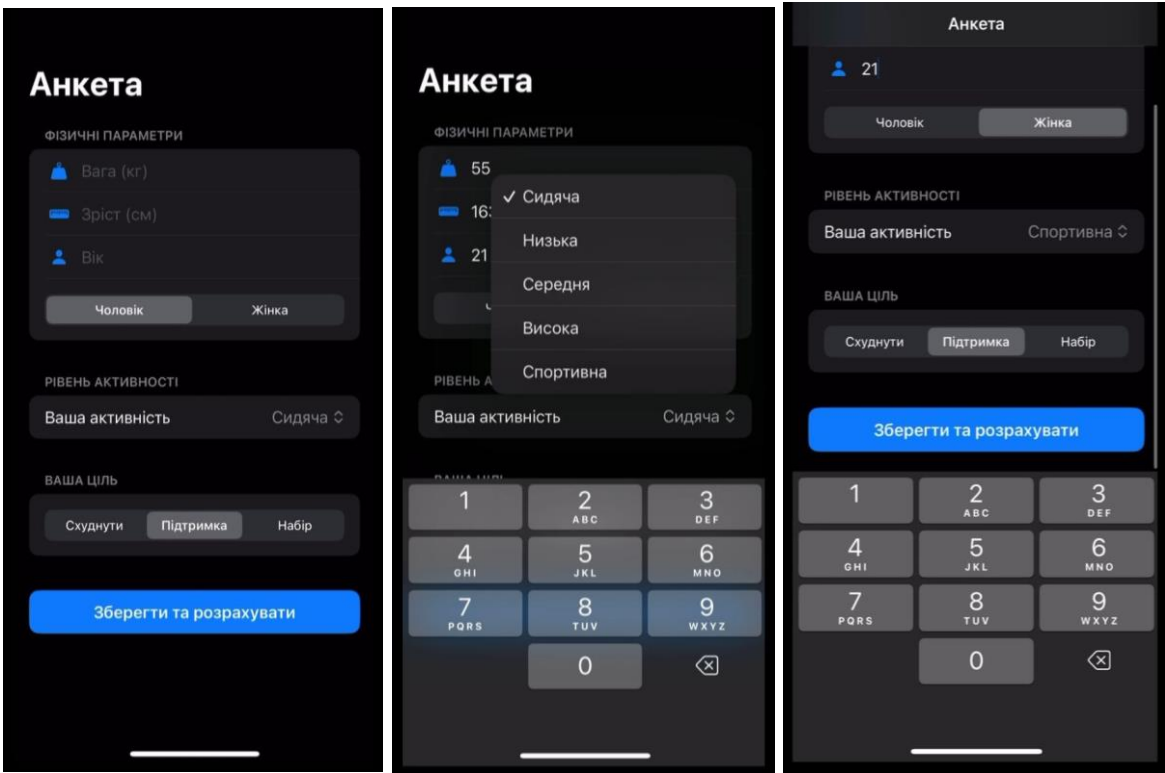


Рис. 2.11 Етапи заповнення анкети користувача (Onboarding) для індивідуального розрахунку калорійності

Зібрані дані не відправляються на сторонні сервери, а зберігаються виключно в безпечному локальному середовищі (Keychain або UserDefaults). На основі цих параметрів ядро додатку виконує розрахунок за медичною формулою (Міффліна-Сан Жеора). До отриманого показника базового обміну речовин (BMR)

застосовується коефіцієнт цілі: мінус 15-20% для схуднення або плюс 10-15% для набору маси. Розрахована фінальна цифра (норма калорій) передається на головний екран (Dashboard) і стає основою для побудови кругових діаграм, які заповнюються в міру споживання їжі протягом дня.

Технологія проєктування мобільного додатку FitFood була реалізована з дотриманням найвищих стандартів сучасної розробки програмного забезпечення.

Спроектований інтерфейс на базі фреймворку SwiftUI забезпечує низьке когнітивне навантаження завдяки використанню ергономічних елементів (кастомна цифрова клавіатура) та мікроінтеракцій (анімації прогрес-індикаторів).

Особливої уваги заслуговує спроектована комбінована архітектура баз даних. Використання локальної SQLite забезпечує високу швидкість та незалежність від інтернету, тоді як інтеграція з Firebase створює унікальну краудсорсингову мережу продуктів між усіма користувачами додатку. Одночасне підключення зовнішніх API (FatSecret та OpenFoodFacts) із функцією сканування штрих-кодів робить каталог продуктів максимально вичерпним як для українського, так і для міжнародного ринку. Гнучка система налаштувань баз даних та персоналізований алгоритм розрахунку калорійності роблять розроблений додаток потужним та конкурентоспроможним інструментом у сфері mHealth.

Результатом усіх етапів проєктування, описаних у даному розділі, стало створення цілісного програмного продукту - мобільного додатку FitFood для платформи iOS. Створений продукт є результатом успішної інтеграції сучасних UI-рішень та складної багатокомпонентної архітектури баз даних.

Технічні особливості створеного продукту:

- Платформена приналежність: додаток оптимізовано виключно для пристроїв на базі iOS, що гарантує високу швидкість роботи та енергоефективність.
- Графічний інтерфейс: створено на основі декларативного фреймворку SwiftUI, що дозволило реалізувати динамічні анімації та адаптивність під різні діагоналі екранів iPhone.
- Автономність: завдяки імплементації локальної бази SQLite, продукт зберігає повну функціональність у режимі офлайн.

- Синхронізація: реалізовано автоматичне фонове оновлення даних через хмарну платформу Firebase, що забезпечує збереження інформації користувача.
- Інформаційна наповненість: інтеграція з API FatSecret та OpenFoodFacts надає продукту доступ до понад мільйона найменувань продуктів.

Таким чином, у ході проєктування було створено програмний продукт, який не лише відповідає теоретичним вимогам, але й є готовим до практичного впровадження інструментом у сфері mHealth. Створений додаток FitFood вирішує проблему швидкого та зручного контролю харчування, поєднуючи в собі точність математичних розрахунків та зручність сучасного мобільного інтерфейсу.

РОЗДІЛ 3. ТЕСТУВАННЯ, ОЦІНКА ЯКОСТІ ТА ПРАКТИЧНЕ ВПРОВАДЖЕННЯ ПРОГРАМНОГО ПРОДУКТУ

Логічним завершенням етапу програмної реалізації будь-якого мобільного додатку є комплексне тестування (Quality Assurance). Оскільки створений додаток FitFood працює з персональними даними користувачів та виконує обчислення індивідуальної норми калорій, забезпечення безперебійної роботи є критичним завданням.

Тестування створеного мобільного застосунку на iOS проводилося вручну (за методом Black Box), без використання складних автоматизованих фреймворків. Весь функціонал перевірявся безпосередньо на фізичному пристрої (iPhone) з актуальною версією операційної системи iOS.

Основна увага приділялася перевірці логіки введення даних, правильності математичних розрахунків і реакції інтерфейсу на невалідні (помилкові) дії користувача. Крім того, для оцінки зручності інтерфейсу (UI/UX) було залучено невелику фокус-групу, що складалася з 12 осіб (студентів), які протягом тижня використовували додаток для ведення власного харчового щоденника.

Першим етапом стало тестування модуля первинного опитування (Onboarding) та валідації полів. Критично важливо, щоб система не дозволяла користувачеві вводити від'ємні значення ваги або залишати поля порожніми.

При спробі зберегти анкету з порожніми полями «Вага» або «Зріст», система успішно блокує перехід на наступний екран та виводить відповідне попередження про помилку. Також було протестовано введення некоректних символів (наприклад, літер замість цифр) у форму додавання власного продукту. Завдяки імплементації цифрової клавіатури (Numpad), додаток фізично унеможливорює введення тексту там, де очікуються числові значення макронутрієнтів. Результат тестування реакції системи на спробу зберегти продукт із незаповненими даними наведено на рис. 3.1.

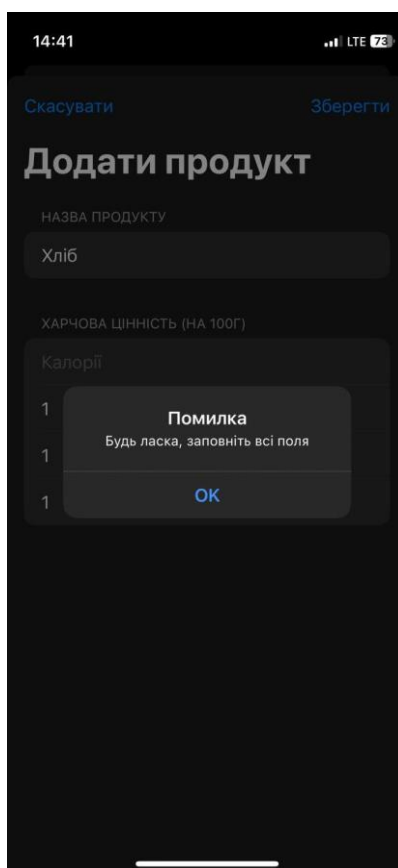


Рис. 3.1 Реакція системи на спробу збереження невалідних даних

Другим етапом стало тестування гібридної бази даних та роботи в режимі офлайн (Offline-first). Для цього на фізичному пристрої було повністю вимкнено підключення до мережі Інтернет (режим «У літаку»). У цьому стані було проведено спробу пошуку раніше збереженого продукту та додавання його до щоденника.

Тестування підтвердило, що локальна база даних SQLite відпрацьовує запити миттєво. Додаток не видає помилок про відсутність мережі при роботі з локальними продуктами, а математичні алгоритми продовжують коректно віднімати спожиті калорії від добової норми на головному екрані. Після відновлення підключення до інтернету, створений офлайн продукт успішно та без дублювання синхронізувався з хмарним середовищем Firebase Cloud Firestore. Процес успішного додавання продукту в режимі відсутності інтернету продемонстровано на рис. 3.2.

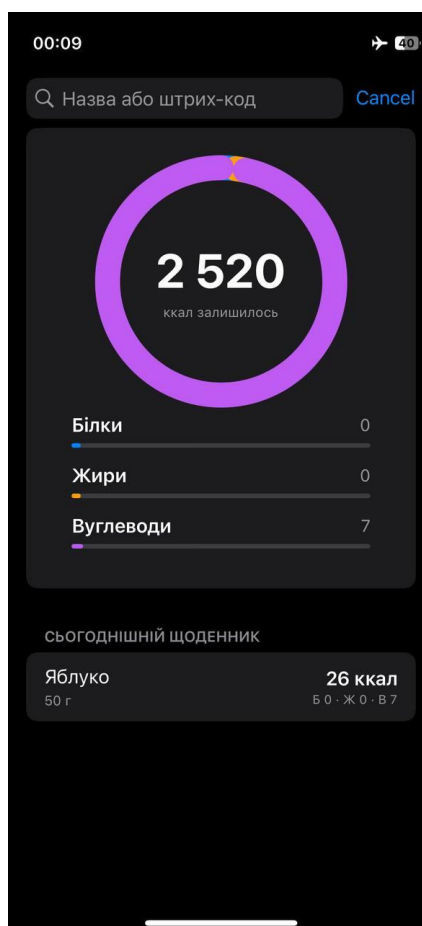


Рис. 3.2 Коректна робота локальної бази даних та розрахунків в офлайн-режимі

Третім етапом стало тестування інтеграції з глобальними базами даних та роботи сканера штрих-кодів. Було перевірено реакцію застосунку на сканування штрих-коду продукту, якого немає в базі OpenFoodFacts. У цьому випадку додаток не завершує роботу екстрено (не «крашиться»), а коректно обробляє помилку (Error Handling), виводячи повідомлення про те, що продукт не знайдено, і пропонуючи користувачеві ввести дані самостійно.

При скануванні валідного штрих-коду (наприклад, популярного шоколаду або напою), система за лічені мілісекунди отримує JSON-відповідь від сервера та підтягує точну кількість білків, жирів та вуглеводів. Інтерфейс успішно розпізнаного продукту через API-запит наведено на рис. 3.3.

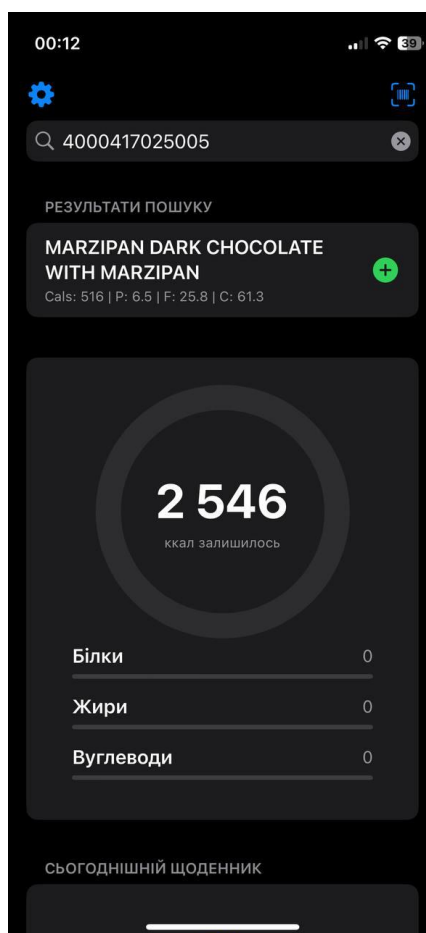


Рис. 3.3 Успішне отримання даних про продукт через OpenFoodFacts API

Проведене ручне тестування не виявило серйозних проблем у працездатності мобільного застосунку. Усі знайдені під час бета-тестування дрібні недоліки (наприклад, проблеми з центруванням деяких текстових елементів на екранах старих моделей iPhone) були оперативно усунуті завдяки адаптивності фреймворку SwiftUI.

Результати тестування фокус-групою з 12 осіб показали високий рівень інтуїтивності створеного інтерфейсу. Користувачі відзначили відсутність зайвого когнітивного навантаження, швидкість роботи сканера штрих-кодів та зручність наявності темної теми (Dark Mode). Продуктивність додатку також відповідає сучасним вимогам: завдяки оптимізації алгоритмів у Swift, завантаження головного екрану займає менше однієї секунди.

Підсумовуючи вищезазначене, можна констатувати, що створений мобільний додаток FitFood є повністю робочим програмним продуктом.

Застосування сучасного стеку технологій на базі екосистеми Apple дозволило випустити стабільний реліз. Система працює коректно, ефективно обробляє помилки введення та успішно функціонує як у глобальній мережі, так і в автономному режимі, що робить її готовою до практичного використання у сфері mHealth.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було успішно створено мобільний додаток FitFood, основним призначенням якого є підрахунок калорійності продуктів, детальний контроль харчування та зручне ведення щоденника спожитої їжі. Для початку було проаналізовано актуальність створення такого продукту та сучасний стан предметної області мобільної охорони здоров'я (mHealth) в Україні та світі. Створення такого мобільного додатку є надзвичайно актуальним завданням, оскільки в умовах швидкого темпу життя користувачі все частіше використовують цифрові інструменти для моніторингу свого здоров'я та підтримки збалансованого раціону.

Стан сфери фітнес-додатків активно розвивається, проте наявні на ринку рішення часто мають суттєві недоліки. Більшість із них вимагають постійного підключення до мережі Інтернет, перевантажені рекламою або обмежують базовий функціонал (наприклад, сканер штрих-кодів) платною підпискою. Тому створення альтернативного, швидкого та безкоштовного інструменту має високий потенціал для залучення широкої аудиторії та є конкурентоспроможним рішенням.

Далі було проаналізовано сучасні інструменти реалізації мобільних додатків та обґрунтовано вибір технологічного стеку. Для забезпечення високої продуктивності та повної відповідності стандартам компанії Apple, створення здійснювалося ексклюзивно для платформи iOS з використанням мови програмування Swift та новітнього декларативного фреймворку SwiftUI. Сучасні інструменти розробки дозволили створити ефективний клієнтський додаток із плавною анімацією, високою стабільністю та надійною архітектурою.

Етапи створення додатку FitFood включали планування, проєктування дизайну (UI/UX) та безпосередню програмну розробку. Планування полягало у визначенні ключових потреб цільової аудиторії та розробці алгоритмів розрахунку індивідуальної норми калорій (онбординг). Дизайн включав створення головного екрана (Dashboard) з використанням кругових діаграм та прогрес-індикаторів для

наочної візуалізації макронутрієнтів (білків, жирів, вуглеводів). Створення охоплювало написання бізнес-логіки та налаштування складної системи баз даних.

У процесі програмної реалізації було створено комплексну гібридну архітектуру баз даних, яка є головною технічною перевагою додатку. Для забезпечення автономної роботи (в режимі офлайн) було спроектовано локальну реляційну базу даних SQLite. Водночас було інтегровано хмарну NoSQL платформу Firebase Cloud Firestore, яка дозволила реалізувати інноваційну краудсорсингову модель: унікальні продукти, додані одним користувачем, автоматично синхронізуються та стають доступними для всієї спільноти. Крім того, налаштовано безперебійну взаємодію з глобальними відкритими API (FatSecret та OpenFoodFacts), що забезпечило додаток функцією миттєвого пошуку за штрих-кодом та величезною базою українських і закордонних товарів.

Для забезпечення зручності користування додатком було створено інтуїтивно зрозумілий та логічний інтерфейс. Користувач має змогу з легкістю знайти необхідну інформацію, додати власні продукти, скористатися кастомною цифровою клавіатурою для введення ваги порції та гнучко керувати активними базами даних у налаштуваннях. Усі зміни супроводжуються плавними мікроінтеракціями, що значно знижує когнітивне навантаження.

Отже, в ході виконання кваліфікаційної роботи було в повному обсязі досягнуто поставленої мети. Головним практичним результатом дослідження є створений та готовий до впровадження програмний продукт - мобільний додаток FitFood для платформи iOS. Створений застосунок є повноцінним інструментом для ведення харчового щоденника, який успішно функціонує в офлайн-режимі завдяки локальній базі даних SQLite та підтримує надійну хмарну синхронізацію через Firebase. Завдяки імплементації інтегрованого сканера штрих-кодів на базі API OpenFoodFacts та реактивного інтерфейсу на SwiftUI, створений програмний продукт повністю відповідає сучасним вимогам ринку mHealth та є конкурентоспроможною альтернативою існуючим комерційним рішенням. Було отримано практичні навички повного циклу створення iOS-додатків: від проектування інтерфейсів у SwiftUI до побудови складних систем синхронізації баз

даних та роботи з REST API. Набуті знання, технічний досвід та компетенції формують міцний професійний фундамент, який може бути успішно використаний у подальшій кар'єрі у сфері інженерії програмного забезпечення. Ознайомитися з кодом створеного додатку можна за посиланням: <https://github.com/taniashkodina3-art/FitFood/tree/main>

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бугайова Ю. Д. Інформаційна система оцінки здоров'я та фізичних навантажень: кваліфікаційна робота на здобуття освітнього ступеня «бакалавр»: спец. 122 «Комп'ютерні науки». Ю. Д. Бугайова ; ЧНУ ім. Петра Могили. Миколаїв, 2025. 104 с. URL: <https://krs.chmnu.edu.ua/jspui/handle/123456789/4152> (дата звернення: 11.03.2026).
2. Вакуленко С.Р. Розробка програмного забезпечення мобільного застосунку для підтримки здорового харчування з базою рецептів і підрахунком калорій: кваліфікаційна робота ..., магістра : 122 Комп'ютерні науки. Станіслав Русланович Вакуленко - Полтава, 2025. - 72 с. URL: <http://dspace.puet.edu.ua/handle/123456789/15891> (дата звернення: 10.03.2026).
3. Гаврилюк В.А., Томчук М.А. Інтерфейс користувача мобільного застосунку для інтелектуального планування завдань. Вінницький національний технічний університет. URL: <https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/48530/25485-90002-1-PB.pdf?sequence=1&isAllowed=y> (дата звернення: 10.03.2026).
4. Дизайн інтерфейсів користувача: навч. посіб. Житомир: ДУ «Житомирська політехніка», 2023. URL: https://learn.ztu.edu.ua/pluginfile.php/242117/mod_resource/content/0/%D0%B5%D0%BA%D1%86%D1%96%D1%8F%205_6.pdf (дата звернення: 11.03.2026).
5. Інформаційні технології. Вимоги щодо доступності продуктів та послуг ІКТ (EN 301 549 V3.2.1 (2021-03), IDT): ДСТУ EN 301 549:2022.[Чинний від 2022-12-01]. Київ: ДП «УкрНДНЦ», 2022. 158 с. URL: https://zakon.isu.net.ua/sites/default/files/normdocs/dstu_en_301_549_2022_informaciyni_tekhnologii_vimogi_schodo_.pdf (дата звернення: 02.04.2026).
6. Іншин В. Д. Розробка веб-додатку «EDA» для організації та контролю за харчуванням людини: кваліфікаційна робота на здобуття освітнього ступеня

- «бакалавр»: спец. 122 «Комп'ютерні науки» / В. Д. Іншин; Національний технічний університет «Дніпровська політехніка». Дніпро, 2022. 54 с. URL: <https://ir.nmu.org.ua/server/api/core/bitstreams/5718b2d8-a940-464b-88c9-98ef499f6cc3/content> (дата звернення: 02.04.2026).
7. Квітка С., Миргородська М. Цифрова трансформація системи охорони здоров'я: фактори впливу на якість життя населення. Аспекти публічного управління. 2024. Т.12, №1. URL: <https://aspects.org.ua/index.php/journal/article/view/1045> (дата звернення: 11.03.2026).
 8. Комаров В. О. Мобільний застосунок автоматизованого підрахунку та контролю калорій та макроелементів: кваліфікаційна робота на здобуття освітнього ступеня «бакалавр»: спец. 121 «Інженерія програмного забезпечення». В. О. Комаров; ЧНУ ім. Петра Могили. - Миколаїв, 2025. - 85 с. URL: <https://krs.chmnu.edu.ua/jspui/handle/123456789/4220> (дата звернення: 15.03.2026).
 9. Німець К. С. Проблеми та перспективи використання систем комп'ютерного зору у робототехніці. Харківський національний університет радіоелектроніки. 2023. URL: <https://openarchive.nure.ua/server/api/core/bitstreams/de37cce2-ee37-4db4-9b2e-1492c7eb6623/content> (дата звернення: 10.03.2026).
 10. Олініченко К. С., Чміль Г. Л., Пахуча Е. В. Трансформаційні зміни маркетингових сервісів в умовах карантину та війни. *Економіка і регіон*. 2023. Вип.1(88). С. 70-80. DOI: [https://doi.org/10.26906/EiR.2023.1\(88\).2875](https://doi.org/10.26906/EiR.2023.1(88).2875). URL: <https://journals.nupp.edu.ua/eir/article/view/2875> (дата звернення: 19.04.2026).
 11. Підходи до розробки мобільних додатків: навч. посіб. Харків: ХНЕУ ім. С. Кузнеця, 2022. URL: <https://repository.hneu.edu.ua/handle/123456789/35853> (дата звернення: 15.03.2026).

- 12.Плескач В. Л., Вакуленко Є. О., Сердюк А. А. Інформаційні системи в нутриціології. *Проблеми програмування*. 2024. № 2-3. С. 199. URL: <http://doi.org/10.15407/pp2024.02-03.199> (дата звернення: 15.03.2026).
- 13.Про схвалення Концепції розвитку цифрової економіки та суспільства України на 2018-2020 роки: розпорядження Кабінету Міністрів України від 17 січня 2018 р. № 67-р. URL: <https://zakon.rada.gov.ua/laws/show/67-2018-%D1%80#Text> (дата звернення: 10.03.2026).
- 14.Роль комп'ютерного зору в сучасному світі: досягнення та виклики. *ResearchGate*. 2024. URL: https://www.researchgate.net/publication/382754859_ROL_KOMP'UTERNOGO_O_ZORU_V_SUCASNOMU_SVITI_DOSAGNENNA_VIKLIKI_TA_PERSPEKTIVI (дата звернення: 19.03.2026).
- 15.Смагіна О. О., Переяславська С. О. Якість програмного забезпечення та тестування: *навчальний посібник*. Старобільськ: ДЗ «ЛНУ імені Тараса Шевченка», 2021. 147 с. URL: <https://dspace.luguniv.edu.ua/xmlui/handle/123456789/7510> (дата звернення: 02.04.2026).
- 16.Харін М. О. Мобільний додаток «Щоденник Здоров'я»: магістерська дисертація на здобуття ступеня магістра: спец. 122 «Комп'ютерні науки». М. О. Харін; КПІ ім. Ігоря Сікорського. Київ, 2024. 110 с. URL: <https://ela.kpi.ua/server/api/core/bitstreams/7390403e-857b-4caf-80a9-bb692abe01d8/content> (дата звернення: 02.04.2026).
- 17.Хас Я. Р., Батюк А. Є. Смарт-система контролю алергенів у харчових продуктах методами і засобами штучного інтелекту. *Науковий вісник НЛТУ України*. 2025. Т. 35, № 6. С. 98-105. DOI: <https://doi.org/10.36930/40350613>. URL: <https://nv.nltu.edu.ua/index.php/journal/article/view/2820> (дата звернення: 02.04.2026).

- 18.Чемерис Г. Ю. UX/UI дизайн: навчальний посібник. Запоріжжя: ЗНУ, 2021. 290 с. URL: <https://dspace.znu.edu.ua/xmlui/handle/12345/5157> (дата звернення: 02.04.2026)
- 19.Штанова А. Л. Мобільні застосунки в системі мобільного маркетингу. *Проблеми сучасних трансформацій*. Серія: економіка та управління. 2024. № 13. DOI: <https://doi.org/10.54929/2786-5738-2024-13-04-05> (дата звернення: 19.03.2026).
- 20.Як створити інтуїтивно зрозумілий додаток для охорони здоров'я: принципи UX/UI. *Wezom Blog*. URL: <https://wezom.com.ua/ua/blog/yak-stvoriti-intuyitivno-zrozumiliy-dodatok-dlya-ohoroni-zdorovya-golovni-printsipi-uxui> (дата звернення: 11.03.2026).
- 21.A Personalized AI-Based Nutrition Tracking App. *International Journal of Innovative Research in Technology (IJIRT)*. URL: https://ijirt.org/publishedpaper/IJIRT181493_PAPER.pdf (дата звернення: 11.03.2026).
- 22.Access data offline. *Firebase Documentation*. URL: <https://firebase.google.com/docs/firestore/manage-data/enable-offline> (дата звернення: 02.04.2026).
- 23.FatSecret: Calorie Counter and Diet Tracker[Електронний ресурс]. URL: <https://www.fatsecret.com> (дата звернення: 02.04.2026).
- 24.Figma Design Tool. URL:<https://www.figma.com/> (дата звернення: 10.03.2026).
- 25.Firebase Firestore - NoSQL Database. URL: <https://firebase.google.com/docs/firestore> (дата звернення: 10.03.2026).
- 26.Healthcare UX: How to design medical apps. *Excited Agency*. URL: <https://excited.agency/ua/blog/healthcare-ux> (дата звернення: 11.03.2026).
- 27.HealthKit. *Apple Developer Documentation*. URL: <https://developer.apple.com/documentation/healthkit> (дата звернення: 03.04.2026).

28. Human Interface Guidelines. *Apple Developer Documentation*. URL: <https://developer.apple.com/design/human-interface-guidelines> (дата звернення: 02.04.2026).
29. Kowalski D. Best Meal Tracker Software. *Gitnux*. 2026. URL: <https://gitnux.org/best/meal-tracker-software/> (дата звернення: 02.04.2026).
30. Lewis J. R. The System Usability Scale: Past, Present, and Future. *International Journal of Human-Computer Interaction*. 2018. Vol. 34, No. 7. P. 577-590. DOI: <https://doi.org/10.1080/10447318.2018.1455307>. URL: <https://www.researchgate.net/publication/324116412> (дата звернення: 02.04.2026).
31. Lifesum. Healthy Eating & Diet. URL: <https://apps.apple.com/us/app/lifesum-food-calorie-tracker/id286906691> (дата звернення: 10.03.2026).
32. Mifflin M. D., St Jeor S. T., Hill L. A. et al. A new predictive equation for resting energy expenditure in healthy individuals. *The American Journal of Clinical Nutrition*. 1990. Vol. 51, No. 2. P. 241-247. URL: <https://www.sciencedirect.com/science/article/abs/pii/S0002916523166986> (дата звернення: 10.03.2026).
33. Open Food Facts API Documentation. URL: <https://world.openfoodfacts.org/data> (дата звернення: 10.03.2026).
34. Pricing plans. *Firebase*. URL: <https://firebase.google.com/pricing> (дата звернення: 02.04.2026).
35. Sinurat F. A. R. Developing a Calorie Requirement Application Based on a BMI Calculator for Android Using User-Centered Design (UCD). *Buletin Poltanesa*. 2025. Vol. 26, No. 2. DOI: 10.51967/tanesa.v26i2.3557. URL: <https://www.researchgate.net/publication/398926132> (дата звернення: 10.03.2026).
36. Siska Narulita et al. Diagram Unified Modelling Language (UML) untuk Perancangan Sistem Informasi. *Bridge: Jurnal Publikasi Sistem Informasi*, 2024. URL: <https://doi.org/10.62951/bridge.v2i3.174> (дата звернення: 03.04.2026).

- 37.SQLite with SwiftUI Tutorial: Getting Started. *Kodeco*. URL: <https://www.kodeco.com/6620276-sqlite-with-swift-tutorial-getting-started> (дата звернення: 02.04.2026).
- 38.SwiftUI Framework Documentation. Apple Developer. URL: <https://developer.apple.com/swiftui/> (дата звернення: 10.03.2026).
- 39.SwiftUI vs UIKit: що обрати для iOS-розробки у 2025 році. *Brander Blog*. URL: <https://brander.ua/blog/swiftui-vs-uikit-shcho-obraty-dlya-ios-rozrobky-u-2025-rotsi> (дата звернення: 03.04.2026).
- 40.The Swift Programming Language. *Apple Developer Documentation*. URL: <https://docs.swift.org/swift-book/> (дата звернення: 02.04.2026).

