**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО”**

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп’ютерних систем**

“До захисту допущено”
Завідувач кафедри
_____ Євгенія СУЛЕМА
“___” _____ 2025 р

# Дипломний проєкт

**на здобуття ступеня бакалавра**

**за освітньо-професійною програмою “Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем”**

**спеціальності 121 Інженерія програмного забезпечення**

**на тему: “Мобільний застосунок для інтерактивного підбору харчування”**

Виконав:
студент IV курсу, групи КП-13
Єрошин Богдан Дмитрович _____

Керівник:
доцент кафедри ПЗКС, к.т.н., доцент,
Люшенко Леся Анатоліївна _____

Консультант з нормоконтролю:
доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
доцент кафедри СПСКС, к.т.н., доцент
Тарасенко-Клятченко Оксана Володимирівна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп’ютерних систем**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 “Інженерія програмного забезпечення”

Освітньо-професійна програма “Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем”

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

“___” _____ 2024 р

# ЗАВДАННЯ

**на дипломний проєкт студенту**

Єрошина Богдана Дмитровича

1. Тема проєкту “Мобільний застосунок для інтерактивного підбору харчування”, керівник проєкту Люшенко Леся Анатоліївна, доцент, к.т.н., затверджені наказом по університету № 1808-С від “29” травня 2025 р.
2. Термін подання студентом проєкту “13” червня 2025 р.
3. Вихідні дані до проєкту: див. Технічне завдання.
4. Зміст пояснювальної записки:
   - аналіз існуючих рішень поставленої задачі;
   - обґрунтування вибору засобів реалізації;
   - розроблення застосунку;
   - аналіз розробленого застосунку.
5. Перелік обов’язкового графічного матеріалу:
   - діаграма діяльності основного сценарію використання (креслення);
   - логічна модель бази даних (креслення);
   - дії користувача щодо мобільного застосунку для інтерактивного підбору харчування (плакат);
   - архітектура застосунку (плакат).

6. Консультанти розділів проєкту

| 7. Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Нормоконтроль | Онай М.В., доцент | | |

7. Дата видачі завдання “29” жовтня 2024 р.

Календарний план

| № з/п | Назва етапів виконання дипломного проєкту | Термін виконання етапів проєкту | Примітка |
|---|---|---|---|
| 1. | ивчення літератури за тематикою проєкту | 15.11.2024 | |
| 2. | озроблення та узгодження технічного завдання | 27.11.2024 | |
| 3. | озроблення структури мобільного застосунку | 18.12.2024 | |
| 4. | ідготовка матеріалів першого розділу дипломного проєкту | 03.02.2025 | |
| 5. | озроблення дизайну сторінок та графічних елементів | 28.02.2025 | |
| 6. | ідготовка матеріалів другого розділу дипломного проєкту | 14.03.2025 | |
| 7. | рограмна реалізація застосунку | 28.03.2025 | |
| 8. | естування застосунку | 17.04.2025 | |
| 9. | ідготовка матеріалів третього розділу дипломного проєкту | 23.04.2025 | |
| 10. | ідготовка матеріалів четвертого розділу дипломного проєкту | 01.05.2025 | |
| 11. | ідготовка графічної частини дипломного проєкту | 09.05.2025 | |
| 12. | формлення документації дипломного проєкту | 23.05.2025 | |

Студент Богдан ЄРОШИН

Керівник проєкту Леся ЛЮШЕНКО

# АНОТАЦІЯ

Даний дипломний проєкт присвячений розробці мобільного застосунку для користувачів, які прагнуть вести здоровий образ життя, харчуватися збалансовано при цьому не потерпати від одноманітності харчового раціону та враховувати певні обмеження на продукти, або нутрієнти.

У роботі виконано: огляд предметної області; обґрунтування актуальності тематики проєкту; порівняльний аналіз існуючих рішень. Обрано та обґрунтовано вибір засобів розробки, проведено опис програмної архітектури, структури та оцінку розробленого мобільного застосунку.

Реалізований мобільний застосунок допомагає користувачам відстежувати калорійність, баланс поживних речовин і рівень корисності продуктів з урахуванням обмежень різного гатунку. Такий застосунок може стати незамінним помічником для тих, хто піклується про своє здоров'я та бажає покращити якість свого харчування.

Застосунок має адаптивний дизайн, що гарантує його коректне відображення на мобільних пристроях, операційних системах та розмірах екранів пристроїв (смартфонах, планшетах тощо). Також впроваджено надійний захист застосунку за допомогою сучасних методів шифрування, та заходів безпеки, що гарантує конфіденційність інформації користувачів і запобігає несанкціонованому доступу до даних.

У даному дипломному проєкті було розроблено ескізну модель застосунку, алгоритм відбору продуктів харчування та їх нутрієнтів, архітектуру застосунку, дизайн інтерфейсу користувача, структуру бази даних, клієнтський і серверний модуль, а також алгоритм їх взаємодії.

# ABSTRACT

This diploma project is dedicated to the development of a mobile application for users who want to live a healthy lifestyle, have a balanced diet, while not suffering from the monotony of it, while taking into account certain restrictions on products or nutrients.

The work includes: review of the subject area, substantiating the relevance of the project's subject matter, and comparative analysis of existing solutions. The choice of development tools was made and substantiated, description of the software architecture, structure, and evaluation of the developed mobile application was done.

The implemented mobile application helps users track calorie intake, nutrient balance, as well as the level of nutritional value of products, taking into account the restrictions of different varieties. Such an application can become an indispensable assistant for those who care about their health and want to improve the quality of their nutrition.

The application has an adaptive design, which guarantees its correct display on mobile devices, operating systems, and device screen sizes (smartphones, tablets, etc.). Reliable protection of the application using modern encryption methods and security measures is also implemented, which guarantees the confidentiality of user information and prevents unauthorized access to data.

This diploma project contains a sketch model of the application, an algorithm for selecting food products and their nutrients, an application architecture and a user interface design, a database structure, a client and server module, as well as an algorithm for their interaction.

.

ДП.045420-01-90 Мобільний застосунок для інтерактивного підбору харчування.
Відомість проєкту

| Позначення | Найменування | Кіл-ть | Примітка |
|---|---|---|---|
| | Документація проєкту | | |
| | | | |
| ДП.045420-02-91 | Мобільний застосунок для | 5 | |
| | інтерактивного підбору харчування. | | |
| | Технічне завдання | | |
| | | | |
| ДП.045420-03-81 | Мобільний застосунок для | 62 | |
| | інтерактивного підбору харчування. | | |
| | Пояснювальна записка | | |
| | | | |
| ДП.045420-04-51 | Мобільний застосунок для | 5 | |
| | інтерактивного підбору харчування. | | |
| | Програма та методика тестування | | |
| | | | |
| ДП.045420-05-34 | Мобільний застосунок для | 7 | |
| | інтерактивного підбору харчування. | | |
| | Керівництво користувача | | |
| | | | |
| ДП.045420-06-99 | Мобільний застосунок для | 1 | |
| | інтерактивного підбору харчування. | | |
| | Діаграма діяльності основного | | |
| | сценарію використання. Схема | | |
| | алгоритму | | |
| | | | |
| ДП.045420-07-99 | Мобільний застосунок для | 1 | |
| | інтерактивного підбору харчування. | | |
| | Логічна модель бази даних. Схема | | |
| | даних | | |
| | | | |
| ДП.045420-08-98 | Мобільний застосунок для | 1 | |
| | інтерактивного підбору харчування. | | |
| | Компакт-диск | | |
| | | | |

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп'ютерних систем**

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2024 р

# "МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ІНТЕРАКТИВНОГО ПІДБОРУ ХАРЧУВАННЯ"

**Технічне завдання**

ДП.045420-02-91

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Леся ЛЮШЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Богдан ЄРОШИН

2024

## 1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

**Назва розробки**: “Мобільний застосунок для інтерактивного підбору харчування”.

**Галузь застосування**: інформаційні технології.

## 2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп’ютерних систем Національного технічного університету України “Київський політехнічний інститут імені Ігоря Сікорського” (КПІ ім. Ігоря Сікорського).

## 3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для користувачів, які мають намір підтримувати здоровий спосіб життя. Забезпечуючи зручний програмний інструмент для інтерактивного підбору харчування і формуванню харчового раціону, мобільний застосунок враховує індивідуальні потреби користувача, обмеження, а також надає можливості планування та відстежування раціонну харчування для досягнення індивідуальних цілей у харчуванні.

## 4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Застосунок повинен забезпечувати такі основні функції:

1) можливість аналізувати продукти, додані користувачем, визначаючи їхню калорійність, баланс поживних речовин;
2) можливість враховувати певні обмеження задані користувачем, індивідуальні рекомендації щодо покращення харчування;
3) певні застереження щодо раціону харчування;
4) інтерактивний інтерфейс для введення даних і отримання зворотного зв'язку;

5) коректне відображання на різних пристроях (смартфони, планшети), підтримка різних розмірів екранів незалежно від їх розміру та роздільної здатності.

Додаткові вимоги:

1) стійкість та безперебійний доступ до застосунку;
2) захист конфіденційної інформації користувачів за допомогою сучасних методів шифрування та безпеки;
3) швидкий час відгуку при обробці даних і наданні рекомендацій;
4) застосунок має бути стабільним і не допускати збоїв у роботі, навіть при великій кількості запитів;
5) інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам легко вводити дані та отримувати інформацію;
6) підтримка кількох мов для зручності користувачів з різних регіонів як додатковий функціонал;
7) застосунок повинен бути доступний для користувачів з різними рівнями технічних навичок.
8) збереження даних користувачів при порушенні у роботі застосунку.

## 5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання дипломного проєкту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   – “Діаграма діяльності основного сценарію використання”;
   – “Логічна модель бази даних”.

## 6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою проєкту ..... 15.11.2024
Розроблення та узгодження технічного завдання..... 27.11.2024
Розроблення структури мобільного застосунку..... 18.12.2024
Підготовка першого розділу дипломного проєкту ..... 03.02.2025
Розроблення дизайну сторінок та графічних елементів..... 28.02.2025
Підготовка другого розділу дипломного проєкту ..... 14.03.2025
Програмна реалізація мобільного застосунку..... 28.03.2025
Тестування мобільного застосунку ..... 17.04.2025
Підготовка третього розділу дипломного проєкту..... 23.04.2025
Підготовка четвертого розділу дипломного проєкту ..... 01.05.2025
Підготовка матеріалів графічної частини проєкту ..... 09.05.2025
Оформлення технічної документації проєкту..... 23.05.2025

## 7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного застосунку виконується відповідно до “Програми та методики тестування”.

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп'ютерних систем**

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2025 р

# МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ІНТЕРАКТИВНОГО ПІДБОРУ ХАРЧУВАННЯ

**Пояснювальна записка**

ДП.045420-03-81

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Леся ЛЮШЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Богдан ЄРОШИН

2025

**ЗМІСТ**

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ ..... 3
ВСТУП ..... 5
1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ПОСТАВЛЕНОЇ ЗАДАЧІ ..... 6
1.1. Загальні положення та аналіз предметної області ..... 6
1.2. Аналіз існуючих рішень ..... 7
1.3. Актуальність розробки застосунку “AlterMeal” ..... 11
1.4. Загальні вимоги до застосунку ..... 13
1.5. Вимоги до безпеки застосунку ..... 14
1.6. Висновки до розділу ..... 18
2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ ..... 20
2.1. Обґрунтування вибору мови програмування “AlterMeal” ..... 21
2.2. Обґрунтування вибору середовища розробки ..... 24
2.3. Обґрунтування вибору бази даних ..... 27
2.4. Висновки до розділу ..... 29
3. РОЗРОБЛЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ “ALTERMEAL” ..... 31
3.1. Ескізний проект мобільного застосунку “AlterMeal” ..... 31
3.2. Сценарії використання застосунку “AlterMeal” ..... 33
3.3. Архітектура мобільний застосунку “AlterMeal” ..... 38
3.4. Структура бази даних мобільного застосунку “AlterMeal” ..... 40
3.5. Висновки до розділу ..... 44
4. АНАЛІЗ РОЗРОБЛЕНОГО МОБІЛЬНОГО ЗАСТОСУНКУ ..... 46
4.1. Опис інтерфейсу користувача ..... 46
4.2. Тестування мобільного застосунку ..... 51
4.3. Пропозиції для майбутнього покращення мобільного застосунку ..... 55
4.4. Висновки до розділу ..... 56
ВИСНОВКИ ..... 58
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ ..... 60
ДОДАТКИ ..... 62

# СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

*ПЗ* – програмний застосунок.

*БД* – база даних.

*UI* – інтерфейс користувача.

*API (Application Programming Interface)* – підхід до архітектури мережевих протоколів, які надають доступ до інформаційних ресурсів.

*Front-end* – частина ПЗ, яка відповідає за відображення та взаємодію з користувачем.

*Back-end* – серверна частина ПЗ, яка відповідає за обробку запитів користувачів та доступ до бази даних.

*Фреймворк* – це набір заздалегідь розроблених структур, бібліотек, інструментів та шаблонів, які надають базову архітектурну основу для розробки програмного забезпечення.

*Login* – ідентифікатор користувача, який використовується для авторизації в системі.

*Аутентифікація* – це процес перевірки ідентичності користувача, щоб забезпечити, що він справді той, за кого він себе видає.

*Авторизація* – це процес перевірки прав доступу користувача до певних ресурсів або функцій системи.

*СКБД* – Система керування базами даних.

*Юніт-тестування (Unit Testing)* – це метод тестування програмного забезпечення, коли окремі компоненти або модулі програми перевіряються на коректність та працездатність.

*Наскрізне тестування (E2E або End-to-End Testing)* – це метод тестування програмного забезпечення, коли весь процес роботи програми перевіряється від початку до кінця.

*КЖБВ* – калорії, жири, білки, вуглеводи.

*ЖБВ* – жири, білки, вуглеводи.

*Раціон харчування* – це набір продуктів та їх кількість, спожитих протягом певного періоду часу, наприклад, доби, тижня або місяця. Він визначає, які продукти, в яких об'ємах і часто, людина вживає для забезпечення своїх енергетичних та поживних потреб.

*Раціональне харчування* – це повноцінне в кількісному та збалансоване в якісному відношенні харчування, що забезпечує нормальний ріст, фізичний та психофізіологічний розвиток організму, його високу працездатність, активне довголіття та стійкість до несприятливих природних, техногенних, соціальних чинників навколишнього середовища.

*Нутрієнти* – складові частини натуральних харчових продуктів, які організм використовує для побудови, оновлення та нормального функціонування органів, тканин і клітин, а також як джерело енергії для виконання роботи і забезпечення життєдіяльності організму в період спокою. До нутрієнтів відносяться білки, жири, вуглеводи, мінеральні речовини, вітаміни і вода. Серед нутрієнтів виділяють замінні і незамінні харчові речовини.

# ВСТУП

ІТ технології швидко розвиваються у сучасному світі, що відкриває безліч можливостей для впровадження інноваційних рішень у різних сферах життя. Однією з таких сфер є сфера харчування людини, де впроваджується весь процес забезпечення інновацій – від виробництва продуктів до їх споживання.

Мобільні застосунки для здорового харчування допомагають користувачам контролювати свій раціон харчування та відстежувати калорійності та баланс поживних речовин. Використання баз даних для відстеження інформації про склад продуктів, їхню поживну цінність, та наявність нутрієнтів певного гатунку є нагальною необхідністю для формування здорового раціону харчування.

Метою дипломного проєкту є створення мобільного застосунку “AlterMeal” для інтерактивного підбору харчування і формуванню харчового раціону, який враховує індивідуальні потреби користувача, медичні, релігійні, інші обмеження, а також забезпечить планування та відстежування раціонну харчування. Застосунок “AlterMeal” забезпечує індивідуальний підхід до раціонального харчування кожного користувача згідно з його цілями. Підбір продуктів на основі смаків та звичок користувача забезпечує м`який перехід до нових харчових звичок.

Застосунок доступний на смартфонах і планшетах, що робить його зручним для використання в будь-якому місці: під час покупок у супермаркеті, на відпочинку, роботі тощо. Користувачі можуть взаємодіяти з застосунком у режимі реального часу.

Застосунок буде корисним для людей, що прагнуть позбавитися зайвої ваги та харчуватися збалансованіше. Він має вирішити проблему зривів від стресу, що спровокований забороною улюблених продуктів. Замість заборони він допоможе підбирати альтернативи улюбленим продуктам.

# 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ПОСТАВЛЕНОЇ ЗАДАЧІ

## 1.1. Загальні положення та аналіз предметної області

Перспективність використання інформаційних технологій в сфері раціонального харчування є беззаперечною. В останній час такі рішення зазнали значного розвитку. Одним із напрямків сфери раціонального харчування є створення сучасних програмних рішень, щодо правильного, здорового та збалансованого харчування [1]. Інтеграція вимог здорового харчування, медичних рекомендацій, екологічних обмежень з інформаційними технологіями виявили необхідність звернути уваги на проблеми та можливості пов’язані з такою інтеграцією . Розглянемо це більш детально.

Персоналізація рекомендацій. Виникла необхідність індивідуальних дієтичних рекомендацій. Є потреба в програмних рішеннях, які дозволяють враховувати індивідуальні особливості користувача (вік, стать, рівень активності, стан здоров'я) і створювати персоналізовані рекомендації щодо харчування.

Освітні інструменти. Потреба в інформаційних платформах дозволяють створювати освітні сайти та застосунки, які інформують користувачів про принципи раціонального харчування, шкоду надмірного споживання певних продуктів і користь збалансованого раціону. Проблема в якості та релевантному контенту.

Відстеження та контроль здоров'я. Необхідність відслідковувати раціон харчування за рекомендаціями лікаря [2]. Таке програмне забезпечення повинно включати дані моніторингу здоров'я, дозволяють користувачам відстежувати параметри здоров'я, включаючи: вагу, рівень цукру в крові, кров'яний тиск та інші параметри тощо.

Стандартизація та регулювання. Завдяки систематизації інформації та даних програмні рішення можуть допомогти у розробці та впровадженні

стандартів харчування, заснованих на наукових дослідженнях та аналізі даних.

Інтерактивність і залучення користувачів. Програмні рішення можуть використовувати ігрові елементи гейміфікацію для підвищення залученості користувачів у процес дотримання здорового харчування, роблячи цей процес цікавішим і мотивуючим. Спільноти та соціальні мережі дозволяють створювати спільноти, які підтримують один одного у дотриманні здорового способу життя, обмінюються рецептами та рекомендаціями.

Екологічні аспекти та сталий розвиток. Програмні платформи можуть інформувати користувачів про екологічний вплив їхнього вибору харчування, заохочуючи до вибору більш екологічних продуктів [3]. Допомагають планувати покупки та уникати надмірних витрат продуктів, що сприяє зменшенню харчових відходів.

Таким чином, розробка програмного забезпечення, що стосується здорового способу життя, є актуальною та надає широкий спектр можливостей для створення та впровадження сучасних вимог і рекомендацій щодо здорового, правильного та збалансованого харчування, сприяючи підвищенню обізнаності в питаннях власного здоров’я, підняттю якості життя та збереженню навколишнього середовища.

## 1.2. Аналіз існуючих рішень

Проблема здорового та корисного харчування вирішується різними способами, залежно від уподобань та потреб користувача. Розглянемо найбільш відомі застосунки для вирішення даної проблеми.

### *1.2.1. YAZIO. https://www.yazio.com/en*

З застосунком YAZIO дуже легко стежити за калоріями, адже тут ви знайдете таблицю калорійності майже всіх продуктів і зручний лічильник (рис. 1.1). Це застосунок може створити персональну програму харчування, яка залежатиме від її мети: схуднення, набір м’язової маси або просто

підтримання фізичної форми. Усі свої прийоми їжі, а також інформацію про заняття спортом і вагу можна записувати в щоденник. Серед інших корисних функцій YAZIO – відстеження кількості споживаних білків, жирів і вуглеводів, а також сканування штрих-кодів продуктів. Передбачені функція інтервального голодування зі зворотнім відліком часу. Застосунок містить перелік готових блюд відповідно до уподобань користувачів [4].

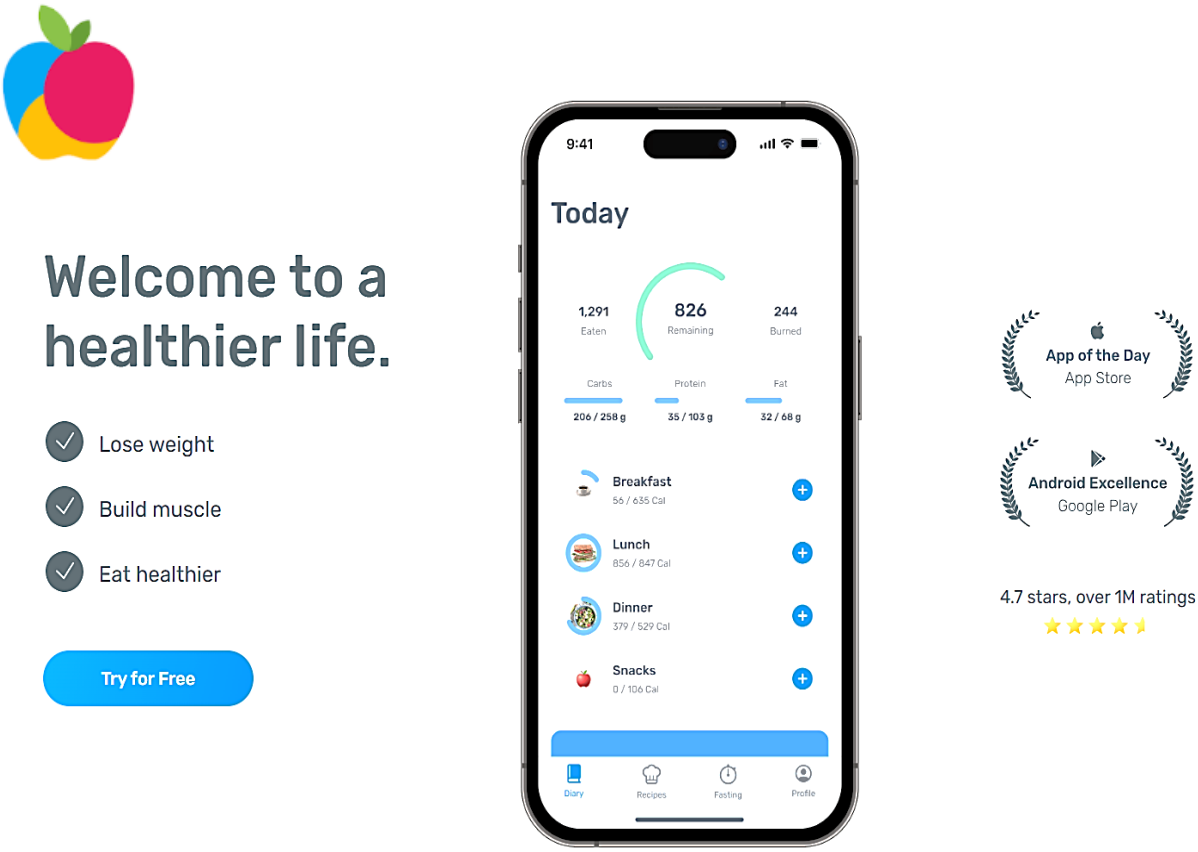



Рис. 1.1. Застосунок для підрахунку калорій “YAZIO”

Застосунок синхронізується з фітнес-трекерами. Також існує PRO - версія з розширеною функціональністю.

Недоліками цього застосунку є:

- не можливість вводити обмеження до вживання харчових продуктів;
- відслідковувати параметри здоров’я (вагу, тиск, рівень глюкози тощо);
- відслідковувати склад харчових продуктів.

### *1.2.2. MyFitnessPal. https://www.myfitnesspal.com*

Одне з найпопулярніших застосунків для підрахунку параметрів харчового раціону таких, як, жири, білки, вуглеводи (ЖБВ), розроблений компанією Under Armour (рис. 1.2). Це програмне забезпечення було випущене у 2005 році та швидко завоювало ринок. Сьогодні “MyFitnessPal” використовують понад 200 мільйонів користувачів по всьому світу.

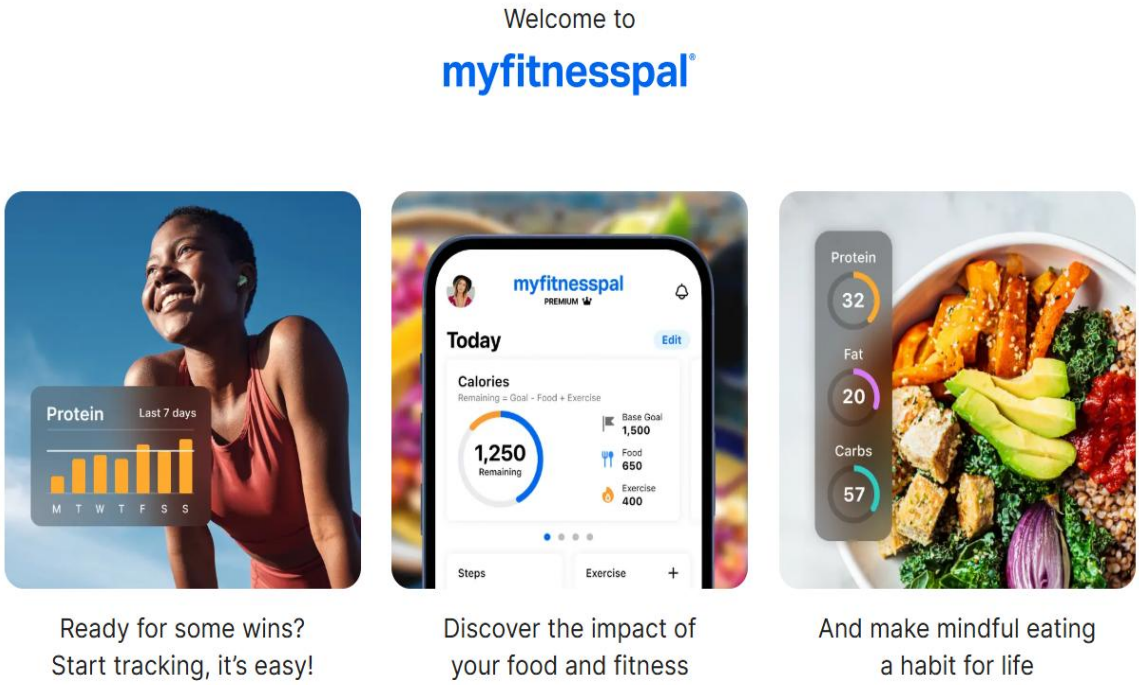



Рис. 1.2. Застосунок “MyFitnessPal”

Застосунок дозволяє стежити за раціоном. У “MyFitnessPal” є величезна база даних з 14 мільйонів продуктів, що дозволяє користувачам швидко знаходити інформацію про харчову цінність і кількість калорій у їхній їжі. У програмі користувачі можуть ставити цілі щодо зміни калорійності їжі, споживання ЖБВ та інших показників, а також відстежувати прогрес у реалізації своїх планів [5].

Крім того, “MyFitnessPal” дозволяє користувачам додавати друзів, обмінюватися з ними результатами, коментувати та “лайкати” їхні пости – підтримка спільноти знімає стрес і полегшує складний шлях відмови від шкідливої їжі на користь здорового способу життя.

“MyFitnessPal” легко інтегрується з популярними застосунками, які допомагають відстежувати фізичну активність, сон, харчування та інші показники здоров’я, такими як Fitbit і Garmin – це дуже зручно для комплексного підходу до здорового способу життя.

Але у “MyFitnessPal” є і недоліки. Помилки в базі даних (як і в будь-якому іншому подібному застосунку) можуть призвести до неправильного підрахунку калорій і поживних речовин.

Обмеження безкоштовної версії: хоча більшість основних функцій застосунку “MyFitnessPal” доступні безкоштовно, деякі розширені функції такі, як видалення реклами та встановлення індивідуальних цілей, вимагають підписки на платну версію. Немає можливості вводити обмеження на продукти харчування.

### *1.2.3. Таблиця калорійності. https://www.tablycjakalorijnosti.com.ua*

“Таблиця калорійності” має українську локалізацію. Містить не тільки базу КЖБВ продуктів харчування, але й за додаткову плату дає більш поглиблений склад продуктів (рис. 1.3). Має перелік готових страв, які підходять для різних систем харчування: вегетаріанство, середземноморська дієта, кето-дієта тощо [6]. “Таблиця Калорійності” існує в версіях для персональних комп’ютерів, мобільних пристроїв, включаючи розумні годинники. Має функції, які додають зручності у користуванні:

- сканування штрих-коду;
- планування здорового харчування;
- контроль за кількістю споживаної води.

До недоліків програми можна віднести обмеження безкоштовної версії, більшість розширених функцій доступні тільки в платному варіанті.

Є стандартні помилки в базі даних. Також у “Таблиці калорійності” досить обмежені можливості для відстежування фізичної активності, на відміну від конкурентів. Крім того, інтерфейс не найпростіший через велике розмаїття функцій і налаштувань, які потребують навичок роботи в застосунку. Немає обмежень, пов’язаних з продуктами харчування та їх складовими.

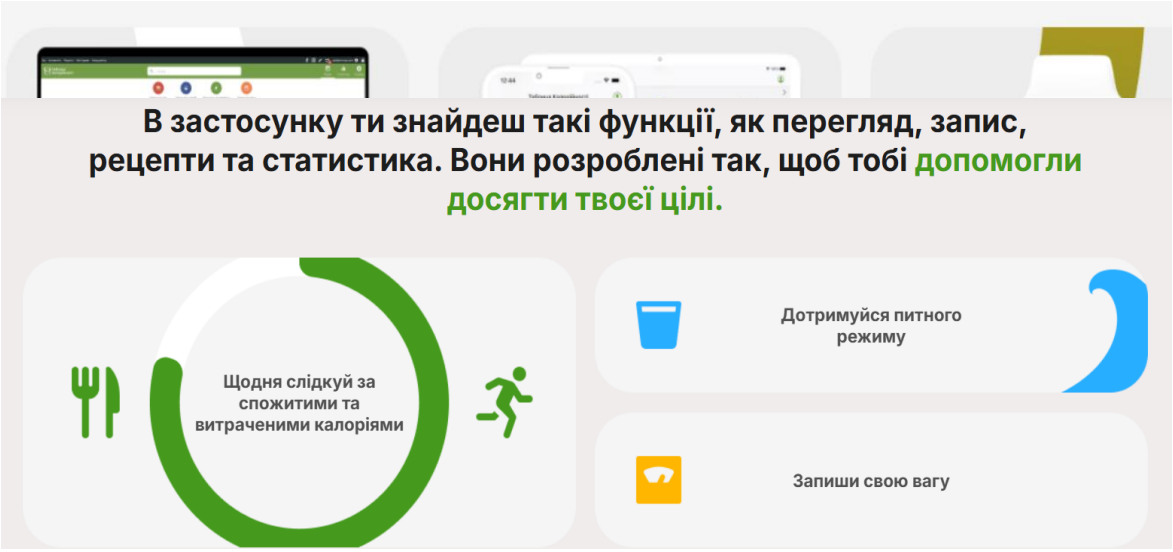



Рис. 1.3. Застосунок “Таблиця калорійності”

### 1.3. Актуальність розробки застосунку “AlterMeal”

Порівнявши особливості згаданих вище застосунків, можна зробити висновок, що кожен з них має певні недоліки та обмеження, які не дозволяють з повна задовольнитися функціональністю. Зв’язку з цим є актуальним застосунок, який може враховувати медичні обмеження такі, як алергії, діабет, особисті непереносимості та медичні поради лікарів тощо. У багатьох людей є вибіркові продукти, які вони не вживають за власним бажанням, або через етичні переконання та релігійні. Окрім того, постійно обмежуючи себе та при вживанні продуктів, які “можна”, замість того, що хочеться призводить до втрати мотивації. Також, нажаль, наші харчові звички зазвичай призводять до того, що одноманітний раціон набридає і споживач не доотримує корисних речовин з їжі.

Не будемо забувати, що важко сформувати бюджетну продуктову корзину, що буде задовольняти різноманітні потреби споживача. Упорядкуємо проблеми у вигляді дерева проблем (рис: 1.4).

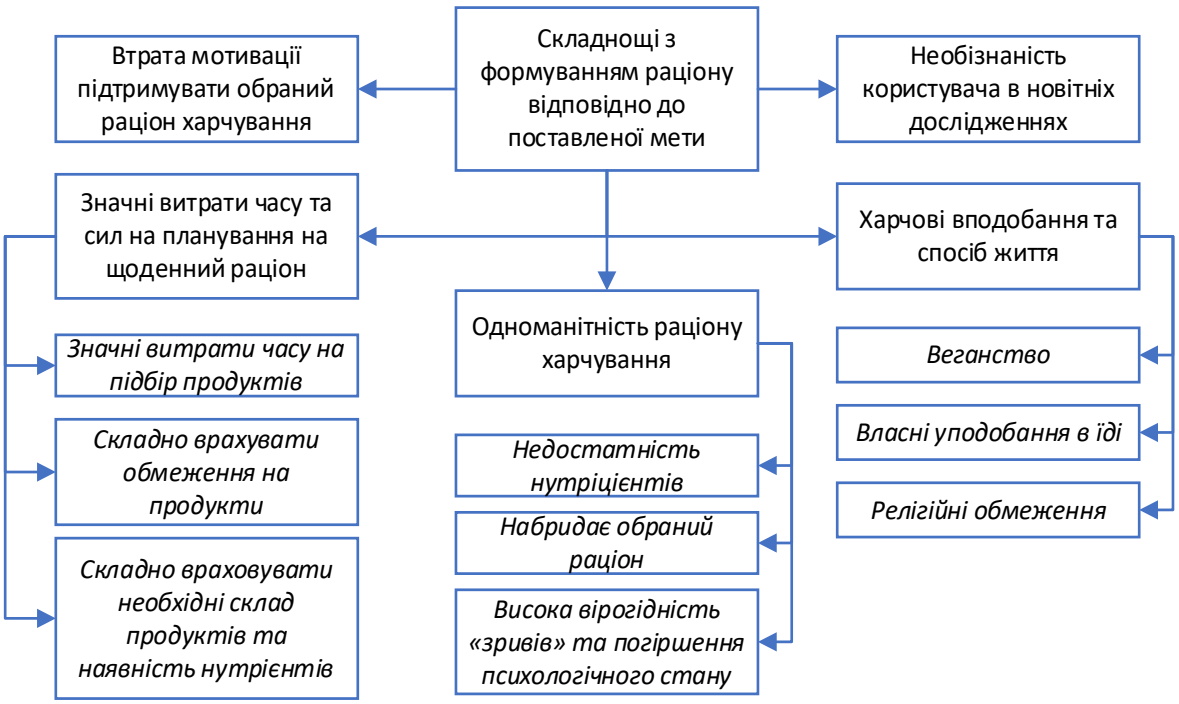



Рис. 1.4. Дерево проблем користувачів, які піклуються про власний харчовий раціон

“AlterMeal” – це застосунок, який аналізуватиме харчові звички споживача та в залежності до його потреб та обмежень. Так, якщо людина має діабет, то застосунок буде знаходити продукти з нижчим глікімічним індексом, а якщо людина має алергію, то будуть обиратися харчові продукти, у складі яких немає складових, на яких діють обмеження. За рахунок того, що продукти будуть підбиратися за подібністю до смаків та бажань споживача, мотивація не буде втрачатися та більш корисне харчування стане просто частиною життя, а не щоденним випробуванням.

Застосунок буде мати наступні конкурентні переваги:

- Індивідуальні рекомендації. Програма враховує унікальні потреби користувача, пропонуючи персоналізовані поради щодо харчування.

- Покращення здоров'я. Застосунок допомагає користувачам дотримуватись збалансованого раціону, що позитивно впливає на здоров'я.
- Зручність. Автоматичний аналіз продуктів у кошику економить час і дозволяє легко планувати покупки.
- Освітній інструмент. Користувачі отримують більше інформації про поживну цінність продуктів і здорове харчування.
- Економія. Пропонуючи більш здорові та часто економічні альтернативи, застосунок допомагає користувачам заощаджувати кошти.

Проаналізувавши проблеми користувачів вдалося розробити рішення, які можна реалізувати у застосунку. За допомогою логіко-структурного аналізу проблеми, яка повинна вирішуватися, завдяки запропонованому програмному забезпеченню стало можливим реалізувати функціональність. У процесі роботи над деревом проблем з’являлися нові ідеї функціональності застосунку такі, як фактор ціни продуктів.

### 1.4. Загальні вимоги до застосунку

Провівши аналіз програмних аналогів та проаналізувавши їх переваги і недоліки, а також дослідивши особливості ринку, було визначено перелік вимог до ПЗ.

Визначимо основні вимоги до функціональності програмного забезпечення, що відповідають потребам користувачів:

- реєстрація та авторизація за допомогою електронної пошти або номеру телефону користувача;
- створення, збереження, редагування та видалення конкретних рекомендацій;
- додавання тегів та категорій до записів для зручного пошуку;
- адаптивний дизайн;
- зручний, естетичний та продуктивний інтерфейс користувача;

- пошук та фільтрація записів за різними критеріями;
- рекомендації щодо харчування з урахуванням індивідуальних цілей (схуднення, підтримка здоров’я, набір м'язової маси тощо);
- можливість налаштування профілю користувача (вік, стать, мета: схуднення, підтримка здоров’я тощо);
- відстеження калорійності та ЖБВ: Збір і відображення даних щодо калорійності, білків, жирів і вуглеводів у спожитих продуктах;
- планування харчування.

Нефункціональні вимоги до мобільного застосунку:

- стійкість та безперебійний доступ до застосунку;
- коректна робота з різними операційними системами: IOS, Android;
- захист конфіденційності, цілісності та доступності даних від несанкціонованого доступу та атак.

Вимоги до якості розроблюваного ПЗ будуть реалізовані завдяки:

- проведенню ручного тестування та контролю якості коду;
- впровадженню кращих практик програмування;
- використанню надійних бібліотек та фреймворків;
- механізмів моніторингу та аналізу виробничого середовища.

### 1.5. Вимоги до безпеки застосунку

У сучасному світі кібербезпека є однією з найважливіших складових захисту інформаційних систем та даних. З поширенням технологій і збільшенням цифрових транзакцій, компанії, уряди та індивідуальні користувачі стали мішенями для кіберзлочинців, що можуть викрадати конфіденційну інформацію, порушувати приватність або завдавати фінансових збитків. Такі кіберзагрози, як хакерські атаки, фішинг, зловмисне програмне забезпечення та викрадення даних, можуть серйозно вплинути на функціонування бізнесів та державних установ, а також завдати шкоди репутації і фінансовій стабільності. Таким чином, кібербезпека стає

важливим елементом для забезпечення безпеки інформаційних систем та захисту особистих даних користувачів.

Захист від кіберзагроз вимагає постійного вдосконалення та адаптації до нових типів атак і технологічних викликів. Важливість кібербезпеки також зростає через інтеграцію нових технологій, таких як Інтернет речей (IoT), штучний інтелект, блокчейн та інші інновації, що створюють нові уразливості. Необхідність захисту від кіберзлочинців також підвищується з розвитком онлайн-платежів, електронної комерції та віддаленої роботи. Тому інвестиції в кібербезпеку, розробка ефективних стратегій захисту та навчання користувачів стають критично важливими для забезпечення надійності та безпеки сучасних інформаційних систем.

Під час проведення аналізу безпеки застосунку “AlterMeal” було виявлено декілька потенційних вразливостей та ризиків, які варто врахувати та виправити для забезпечення максимального рівня захисту даних та конфіденційності користувачів.

Розглянемо можливі негативні сценарії, які можуть виникнути внаслідок перебоїв у роботі серверу, неочікуваного використання застосунку або спроб злому програмного забезпечення.

Табл. 1.1 містить опис вразливих місць та потенційних загроз для застосунку, які можуть вплинути на безпеку бізнесу та користувачів.

Таблиця 1.1

Потенційні загрози застосунку

| № | Вразливість | Загроза | Наслідки |
|---|---|---|---|
| 1. | Відсутність або слабкість механізмів аутентифікації та авторизації | Може дозволити зловмисникам отримати несанкціонований доступ до системи | Може призвести до компрометації даних, крадіжки інформації або здійснення шкідливих дій від імені легітимних користувачів |

Продовження табл. 1.1

| | | | |
|---|---|---|---|
| 2. | Вразливості SQL запитів | Можуть дозволити зловмисникам виконувати SQL запити для отримання несанкціонованого доступу до бази даних | Може призвести до витоку конфіденційних даних, їх модифікації або знищення, а також до порушення роботи системи |
| 3. | Вразливості в сторонніх бібліотеках або фреймворках | Можуть бути використані зловмисниками для компрометації системи або виконання шкідливого коду | Може призвести до порушення безпеки застосунку, витоку даних або збоїв у роботі системи, що вплине на користувачів і бізнес |
| 4. | Надмірний доступ до даних | Може дозволити користувачам або системам отримувати інформацію, яка їм не призначена, підвищуючи ризик витоку або зловживання даними | Може призвести до порушення конфіденційності, компрометації чутливої інформації та потенційних юридичних і репутаційних втрат для компанії |
| 5. | Невірна обробка помилок | Може розкрити технічні деталі системи, які зловмисники можуть використати для атак | Може призвести до збільшення вразливостей, компрометації системи та підвищення ризику несанкціонованого доступу або збоїв |
| 6. | Вразливості через зловмисне програмне забезпечення | Може проникнути в систему через уразливості в програмному забезпеченні або людський фактор, спричиняючи несанкціонований доступ до даних або ресурсів | Може призвести до витоку, пошкодження або крадіжки даних, а також до серйозних порушень роботи системи, що завдасть шкоди бізнесу і користувачам |

Після виявлення всіх можливих загроз, що можуть вплинути на стабільність і безпеку роботи застосунку, було проведено оцінку потенційних ризиків і створено таблицю вимог до безпеки програмного забезпечення, яка наведена в табл. 1.2.

Таблиця 1.2

Безпекові вимоги застосунку

| № | Вразливість | Загроза | Безпекова вимога |
|---|---|---|---|
| 1. | Вразливості SQL запитів | Можуть дозволити зловмисникам виконувати SQL запити для отримання несанкціонованого доступу до бази даних | Коректна перевірка SQL запитів для уникнення SQL ін’єкцій та забезпечення безпеки бази даних |
| 2. | Відсутність або слабкість механізмів аутентифікації та авторизації | Може дозволити зловмисникам отримати несанкціонований доступ до системи | Надійні механізми аутентифікації та авторизації для запобігання несанкціонованому доступу |
| 3. | Вразливості в сторонніх бібліотеках або фреймворках | Можуть бути використані зловмисниками для компрометації системи або виконання шкідливого коду | Перевірка та моніторинг сторонніх бібліотек і фреймворків на наявність відомих вразливостей та їх регулярному оновленні |
| 4. | Надмірний доступ до даних | Може дозволити користувачам або системам отримувати інформацію, яка їм не призначена, підвищуючи ризик витоку або зловживання даними | Обмеження доступу до даних на основі принципу найменших привілеїв, щоб користувачі могли отримувати лише ту інформацію, яка їм необхідна для виконання їхніх обов'язків |

Продовження табл. 1.2

| | | | |
|---|---|---|---|
| 5. | Невірна обробка помилок | Може розкрити технічні деталі системи, які зловмисники можуть використати для атак | Належна обробка помилок, щоб уникнути розкриття технічних деталей системи та забезпечити захист від потенційних атак |
| 6. | Вразливості через зловмисне програмне забезпечення | Може проникнути в систему через уразливості в програмному забезпеченні або людський фактор, спричиняючи несанкціонований доступ до даних або ресурсів | Використання антивірусного програмного забезпечення, фаєрволів та регулярному оновленні системи для запобігання проникненню зловмисного програмного забезпечення |

Виконання всіх вимог безпеки, зазначених у табл. 1.2, гарантує надійність та безпеку функціональних можливостей програмного застосунку.

### 1.6. Висновки до розділу

Для вдосконалення своїх харчових звичок, пошуку та отримання оптимального раціону харчування інформаційні технології є невід'ємною складовою, що сприяє оптимізації процесів раціонального харчування. Впровадження програмного забезпечення для користувачів є кроком до покращення здоров'я та підвищення якості життя.

Аналіз існуючих програмних рішень показує, що такі застосунки, як YAZIO, MyFitnessPal і Lifesum, хоч пропонують певну гнучкість і зручність для користувача, але не відповідають конкретним потребам і процесам, зазначеним вище. Хоча ці застосунки корисні для загального використання, вони можуть не вирішувати конкретні проблеми.

Після порівняння функціональності існуючих застосунків зазначимо, що кожен з них має свої недоліки та обмеження. Враховуючи необхідність зручності та простоти в користуванні, а також наявність спеціалізованих шаблонів та інструментів, є потреба в розробці застосунку, який забезпечить:

- індивідуалізацію під конкретні потреби користувачів;
- покращене користувацьке середовище;
- інтеграція з іншими сервісами та додатками;
- покращену базу даних і точність аналізу;
- зосередженість на здоров'ї та безпеці;
- аналіз харчового кошика в реальному часі;
- розширена функціональність для підтримки здоров'я;
- прозорість і контроль.

Згідно зазначеного, було сформульовано перелік вимог до застосунку “AlterMeal”. Серед функціональних вимог відзначається можливість створення та зберігання шаблонів та корисних звичок в харчуванні, функціональність для зазначених вище вимог. Нефункціональні вимоги включають забезпечення стійкості та безперебійного доступу до застосунку, коректну роботу у різних операційних системах, та кіберзахист, у тому числі й конфіденційності даних. Для забезпечення вимог до якості буде використано ручне тестування.

У сучасному цифровому середовищі збільшення кількості кібератак та порушень безпеки даних створює загрози не тільки для особистої інформації, а й для корпоративних даних, що робить питання безпеки ключовим елементом при розробці програмного забезпечення. Забезпечуючи дотримання всіх вимог безпеки та виправляючи існуючі вразливості, можна забезпечити надійність та безпеку функціональності застосунку “AlterMeal”.

# 2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

Перш ніж розпочати реалізацію програмного застосунку, важливо визначити підходи до розробки. Існує кілька підходів до створення мобільних застосунків: нативний, гібридний і кросплатформний. Кожен із них має свої переваги й обмеження, тому вибір базувався не лише на технічних можливостях таких підходів розробки, а й на вимогах до використання застосунку, також необхідно враховувати наявні апаратні ресурсі та досвід використання. Розглянемо більш детально зазначені підходи до розробки.

Нативна розробка – це створення окремих застосунків для Android та iOS, з використанням відповідних середовища та мов програмування. Така розробка забезпечує максимальну продуктивність і доступ до всіх функцій мобільного пристрою. Проте нативна розробка вимагає більше часу, досвіду й зусиль, адже кожна програмна платформа потребує окремого кодування й підтримки.

Гібридний підхід, на основі якого створюються гібридні застосунки. Такі застосунки по суті є вебсайтами, “загорнутими” в оболонку мобільного застосунка. Вони швидші в реалізації, однак часто поступаються в інтерфейсах, продуктивності та інтеграції з пристроєм.

Кросплатформний підхід – “золота середина”. Він дозволяє створювати один застосунок, який працює одразу на кількох операційних системах. Перш за все це єдиний код. Немає потреби в окремій розробці версій для iOS та Android. Це дуже пришвидшує розробку та дає можливість поширити використання застосунку на більшу кількість користувачів.

Для проєкту “AlterMeal” використання кросплатформного підходу є найбільш логічним та дозволяє використати вже наявні знання та досвід розробки на мові програмування C# та в Unity. Багатоплатформне середовище розробки Unity є програмним засобом для розробки комп’ютерних ігр та застосунків, що є рушієм, на якому вони працюють.

### 2.1. Обґрунтування вибору мови програмування “AlterMeal”

На початковому етапі реалізації мобільного застосунку “AlterMeal” одним із найважливіших рішень стало обрання мови програмування. Це рішення впливає не лише на технічні аспекти реалізації функціональності, а й на швидкість розробки, можливість підтримки та розвитку застосунку в майбутньому. Розглянуто декілька актуальних мов: C#, Java, Kotlin, Dart, JavaScript (у зв’язці з React Native), Swift, а також C++ у зв’язці з Qt.

***C#***

C# (у поєднанні з Unity) C# – об’єктно-орієнтована, строго типізована мова, яка поєднує простоту Java з потужністю C++. У зв’язці з Unity C# дає змогу створювати кросплатформні застосунки з одним кодовим базисом для Android і iOS. C# має сучасний синтаксис, велику кількість бібліотек, інтеграцію з REST API, підтримку JSON, роботу з локальними базами даних та ефективний garbage collection. Головна перевага – швидка розробка і нативна інтеграція в Unity, який також підтримує роботу з UI, анімацією та сценою через C#-скрипти [7] .

***Java (Android SDK)***

Java – одна з найстаріших і найстабільніших мов для мобільної розробки. Її перевага полягає у великій кількості документації та підтримки Android-платформи. Проте, вона має складніший синтаксис порівняно з C#. Відсутність підтримки iOS без сторонніх засобів потребує значно більше часу на розробку UI вручну через XML [8].

***Kotlin***

Kotlin – офіційно рекомендована мова для Android-розробки. Kotlin має компактний синтаксис, краще працює з лямбда-виразами та null-безпекою. Не дивлячись на те, використовуються для програмування Android, macOS, Windows, Linux, watchOS, підтримка кросплатформності обмежена, реалізація для iOS здійснюється через Kotlin Multiplatform, що ускладнює розробку для новачків. Як зазначає розробник мови Kotlin, що Kotlin Multiplatform має недостатньо кросплатформні бібліотеки [9].

***Dart***

Dart (у поєднанні з Flutter) Flutter – сучасний фреймворк для кросплатформенної розробки від Google. Dart розроблено спеціально для зручної роботи з інтерфейсом. Переваги Dart: швидке створення UI, гаряче перезавантаження (hot reload), великий набір готових віджетів. Недоліки – менша продуктивність у складних обчисленнях порівняно з C#, складніша інтеграція з платформозалежними API, потреба у вивченні нової мови [10].

Окремо варто відзначити, що Dart має гарну підтримку спільноти, зрозумілу і досить докладну документацію та приклади готових рішень для основних задач мобільної розробки. Це значно спрощує процес впровадження нового функціоналу, прискорює розробку та дозволяє легко знаходити відповіді на технічні питання. Також важливою перевагою є те, що Dart забезпечує стабільну роботу коду на різних платформах, що критично для кросплатформенних застосунків.

***JavaScript (React Native)***

React Native дозволяє розробляти мобільні застосунки за допомогою JavaScript. Переваги JavaScript (React Native)– велика спільнота, легкість початку розробки, підтримка плагінів. Недоліки – відсутність строгого типізування (за винятком використання TypeScript), нижча продуктивність, необхідність взаємодії з нативними модулями через міст (bridge), що ускладнює використання апаратних функцій пристроїв [11].

***Swift***

Swift – офіційна мова для розробки під iOS, яка спеціально була розроблена та підтримується компанією Apple для власних пристроїв. Вона безпечна, швидка, сучасна. Swift має можливості видаляти цілі класи небезпечного коду [12]. Однак, як і Java, вона обмежується лише однією платформою. Для повноцінної підтримки Android довелось би створювати застосунок ще й на Kotlin або Java, що подвоїло б обсяг роботи при розробці.

***C++ (у поєднанні з Qt)***

Qt – потужний фреймворк для кросплатформної розробки з використанням C++, яка є простою та достатньо добре документована з великим співтовариством.. C++у поєднанні з Qt – висока продуктивність та кросплатформність.. Проте, реалізація UI у Qt виглядає дещо складнішою і менш гнучкою для мобільних застосунків. Крім того, C++ складніший у підтримці, має вищу вхідну межу та не так добре підтримується у сфері мобільної розробки, як інші мови [13].

В табл. 2.1 наведено порівняльний аналіз зазначених вище мов програмування.

Таблиця 2.1

Аналіз мов програмування для розробки “AlterMeal”

| Мова програмування | Підтримка Android | Підтримка IOS | Кросплатформа | Простота | Продуктивність | Комфорт у UI |
|---|---|---|---|---|---|---|
| C# | Так | Так | Так | Проста | Висока | Високий |
| Java | Так | Ні | Ні | Середня | Висока | Низький |
| Kotlin | Так | Частково | Частково | Проста | Висока | Середній |
| Dart | Так | Так | Так | Проста | Висока | Високий |
| JS (RN) | Так | Так | Так | Проста | Середня | Середній |
| Swift | Ні | Так | Ні | Середня | Висока | Високий |
| C++ (Qt) | Так | Так | Так | Складна | Середня | Середній |

Отже мова програмування Dart була обрана для розробки мобільного застосунку, оскільки вона оптимально підходить для створення сучасних мобільних застосунків. Dart спеціально розроблений для ефективної роботи із графічними інтерфейсами та забезпечує високу продуктивність на мобільних пристроях. Простий і зрозумілий синтаксис мови дає можливість швидко реалізовувати різноманітні функції, що особливо важливо при створенні застосунків із великою кількістю взаємодії з користувачем.

### 2.2. Обґрунтування вибору середовища розробки

Вибір середовища розробки напряму впливає на ефективність створення програмного продукту. У випадку мобільного застосунку “AlterMeal” особливо важливо знайти таке середовище розробки, яке б забезпечувало зручну розробку, підтримку кросплатформенності, широкі можливості для створення інтерфейсу, а також дозволяло інтегрувати базу даних і сторонні API. Розглянемо та проаналізуємо використання середовищ розробки для реалізації програмного проєкту “AlterMeal” (табл. 2.2).

Таблиця 2.2

Аналіз середовищ розробки мобільного застосунку

| Середовище | Підтримка Android/ IOS | Мова | UI-інструменти | Особливості |
|---|---|---|---|---|
| Unity | Так | C# | Canvas, UI Toolkit | Ігровий рушій з широкими можливостями |
| Android Studio | Лише Android | Java/Kotlin | XML + Jetpack Compose | Офіційне середовище для Android, без iOS |
| Flutter | Так | Dart | Віджети Flutter | Потужний UI, потребує Dart. Кросплатформність |

Продовження табл. 2.2

| | | | | |
|---|---|---|---|---|
| Xamarin | Так | C# | XAML, Forms | Підтримка нативних елементів, складна конфігурація |
| React Native | Так | JavaScript/ TS | Компоненти RN | Потрібна інтеграція з нативним кодом для повної функціональності |
| Xcode | Лише iOS | Swift | SwiftUI | Для розробки під iOS; не підтримує Android |

***Unity***

*Unity* – це потужний рушій для створення кросплатформних застосунків та комп'ютерних ігр, який з моменту свого створення зарекомендувала себе як зручний та масштабований інструмент не лише для ігор, а й для утиліт, симуляцій, візуалізацій та мобільних застосунків. Середовище має вбудовану підтримку мови C#, що стало додатковим плюсом, враховуючи наявні знання [14]. Основні переваги Unity:

- Кросплатформність. Unity дозволяє створювати застосунки одночасно для Android та iOS, що дає можливість уникнути дублювання коду та зменшити витрати часу.
- Візуальний редактор інтерфейсу. Unity має Canvas-систему, яка дозволяє зручно і швидко створювати UI, працювати з адаптивними елементами та анімаціями.
- Підтримка плагінів. За необхідності, можна легко підключити сторонні бібліотеки для роботи з базами даних, мережею чи сенсорами пристрою.
- Велика спільнота та документація. Unity має активну базу користувачів, що полегшує пошук рішень і прискорює процес навчання.

- Редактор сцен. Дозволяє у зручному графічному вигляді створювати структуру застосунку, переходи між екранами тощо.

***Flutter***

Flutter є сучасним фреймворком, який вимагає знань мови програмування Dart та використовує інтерфейс Google для розробки мобільних застосунків на iOS, Android.

Flutter підтримує таку корисну функцію як “гаряче перезавантаження”, що дає змогу швидко тестувати зміни у коді та оперативно виправляти помилки. Велика та активна спільнота розробників Flutter забезпечує доступ до численних прикладів, відкритих бібліотек і детальної документації, що значно полегшує реалізацію навіть нетипових вимог. В результаті, вибір Flutter є найбільш доцільним для розробки простого, надійного та зрозумілого мобільного застосунку який може працювати на різних платформах без зайвих ускладнень [15].

Розглянемо ще декілька середовищ розробки (табл. 2.2), які мають суттєві недоліки з точки зору розробки мобільного застосунку “AlterMeal”, а саме:

- Android Studio та Xcode не дозволяють створювати один застосунок одразу для двох платформ.
- Xamarin хоч і підтримує C#, проте має складнішу архітектуру, не настільки зручні інструменти UI, і потребує більшої конфігурації.
- React Native базується на JavaScript, що потребує інтеграції з TypeScript для строгої типізації та додаткового коду для роботи з апаратною частиною пристрою.

В якості основного середовища розробки для цього застосунку обрано Flutter. Цей фреймворк надає можливість створювати мобільні додатки для Android та iOS з використанням єдиної кодової бази, що дозволяє суттєво економити час на розробку, оновлення та підтримку проєкту. Flutter вирізняється широкими можливостями для побудови сучасних, гнучких

інтерфейсів завдяки великій кількості готових віджетів та елементів дизайну.

### 2.3. Обґрунтування вибору бази даних

Основою роботи мобільного застосунку “AlterMeal” є база даних тому, що для успішного функціонування цього застосунку потрібне потужне сховище продуктової бази, що повинна однозначно та швидко шукати продукти харчування. Зважаючи на особливості функціональності та вимоги до автономної роботи застосунку, було вирішено використовувати на першому етапі локальну базу даних зі збереженням архіву щоденного харчового раціону на сервері.

Локальне зберігання даних має низку переваг у контексті забезпечення стабільної роботи застосунку незалежно від наявності або якості інтернет з’єднання. Окрім того, під час проєктування мобільного застосунку одним із принципових рішень стало забезпечення приватності та безпеки особистої інформації користувача. Такий підхід має кілька вагомих переваг. Кожен користувач працює з застосунком автономно, а всі введені дані зберігаються лише на його пристрої. При бажанні користувач може зберігати щоденний раціон харчування на сервірі в архіві. Це виключає можливість централізованого збору інформації про харчові звички, цілі, раціони чи інші особисті вподобання.

Використання локальної бази даних забезпечує також додаткову гнучкість для користувача. Кожен може самостійно створювати та редагувати свої раціони, додавати улюблені страви, зберігати цілі щодо схуднення або набору ваги – все це належить тільки конкретному пристрою. У разі необхідності дані можна видалити або змінити у будь-який момент, не турбуючись про їхнє поширення чи обробку кимось іншим.

У підсумку, локальне зберігання даних – це не лише простий і зручний варіант реалізації, а й важлива складова загальної стратегії захисту особистої інформації. Саме така концепція була покладена в основу

застосунку, щоб кожен користувач міг самостійно контролювати власні дані та не турбуватися про їхню безпеку. Завдяки цьому користувачі можуть взаємодіяти із застосунком у будь-який момент – без необхідності підключення до мережі.

Як правило основний вибір стоїть між реляційними та неряційними базами даними. В зв'язку з тим, що продуктова база є структурованою таблицею, то немає сенсу використовувати нереляційну базу типу MongoDB, яка є досить гнучкою для створення бази даних для інформації різного типу без необхідності строгого структурування.

Серед реляційних баз даних вибір стоїть між PostgreSQL і SQLite. Розглянемо PostgreSQL – це масштабна реляційна база даних, яка має дуже великий спектр використання, має можливість створювати складні, комплексні SQL-запити [16]. Проте ця база даних є найкращою для використання в вебзастосунках, які є не такими вибагливі до обчислювальних ресурсів. Розглянемо більш детально недоліки використання PostgreSQL в мобільних застосунках:

- необхідно більше оперативної пам'яті та інших апаратних ресурсів;
- необхідно ресурси на адміністрування та конфігурування;
- не пристосований для мобільних застосунків;
- передбачає серверні рішення;
- складне масштабування.

В зв'язку з цим для мобільного застосунку краще використовувати SQLite, яка адаптована до розробки мобільних застосунків. SQLite підтримується більшістю мобільних платформ та має перевірену репутацію у сфері мобільної розробки. Ця система дозволяє створювати структуровані таблиці, встановлювати зв'язки між ними, виконувати запити фільтрації, сортування, пошуку – що є критично важливим для реалізації логіки застосунку.

Завдяки використанню SQLite забезпечується можливість зберігати повний перелік продуктів з описом харчової цінності та категоріями,

інгредієнтами та нутрієнтів. Незважаючи на те, що база даних включає багато записів, її загальний розмір залишається незначним – у межах декількох мегабайт. Такий обсяг даних не створює навантаження на пам’ять пристрою та забезпечує швидкий доступ до необхідної інформації. Крім того, у разі потреби дані можуть бути структуровані за категоріями або розділені на окремі таблиці, що підвищує ефективність роботи з ними [17].

У підсумку, рішення на користь локальної бази даних дозволяє забезпечити стабільну та передбачувану роботу застосунку в будь-яких умовах. Обрані програмні засоби відповідають потребам проєкту та підтримує реалізацію всіх ключових функцій без додаткових зовнішніх залежностей.

### 2.4. Висновки до розділу

Таким чином, для реалізації мобільного застосунку “AlterMeal” було розглянуто декілька актуальних мов програмування: C#, Java, Kotlin, Dart, JavaScript (у зв’язці з React Native), Swift, а також C++ у зв’язці з Qt. В результаті аналізування переваг та недоліків наведених вище засобів програмування було обрано Dart , яка у поєднанні дає досить значні можливості реалізації кросплатформних мобільних застосунків. Dart розроблено спеціально для зручної роботи з інтерфейсом. Переваги Dart: швидке створення UI, гаряче перезавантаження (hot reload), великий набір готових віджетів, потужна спільнота розробників.

Для розробки мобільного застосунку “AlterMeal” особливо важливо знайти таке середовище розробки, яке б забезпечувало зручність, підтримку кросплатформності, широкі можливості для створення інтерфейсу, а також дозволяло інтегрувати базу даних і сторонні API. В якості основного середовища розробки для застосунку обрано Flutter. Цей фреймворк надає можливість створювати мобільні застосунки для Android та iOS з використанням єдиної кодової бази, що дозволяє суттєво економити час на розробку, оновлення та підтримку проєкту. Flutter вирізняється широкими

можливостями для побудови сучасних, гнучких інтерфейсів завдяки великій кількості готових віджетів та елементів дизайну.

Для вибору відповідної бази даних необхідно визначитись з типом бази даних, а саме: реляційна, нереляційна. Специфіка мобільного застосунку “AlterMeal” передбачає лише використання реляційних баз даних. Для мобільного застосунку краще використовувати SQLite, яка адаптована до розробки мобільних застосунків. SQLite підтримується більшістю мобільних платформ. Ця БД дозволяє створювати структуровані таблиці, встановлювати зв’язки між ними, виконувати запити фільтрації, сортування, пошуку – що є критично важливим для реалізації логіки застосунку.

# 3. РОЗРОБЛЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ “ALTERMEAL”

## 3.1. Ескізний проект мобільного застосунку “AlterMeal”

Проаналізувавши дерево проблем (рис. 1.4) та основні вимоги до мобільного застосунку “AlterMeal” створимо та опишімо діаграму прецедентів (рис. 3.1).

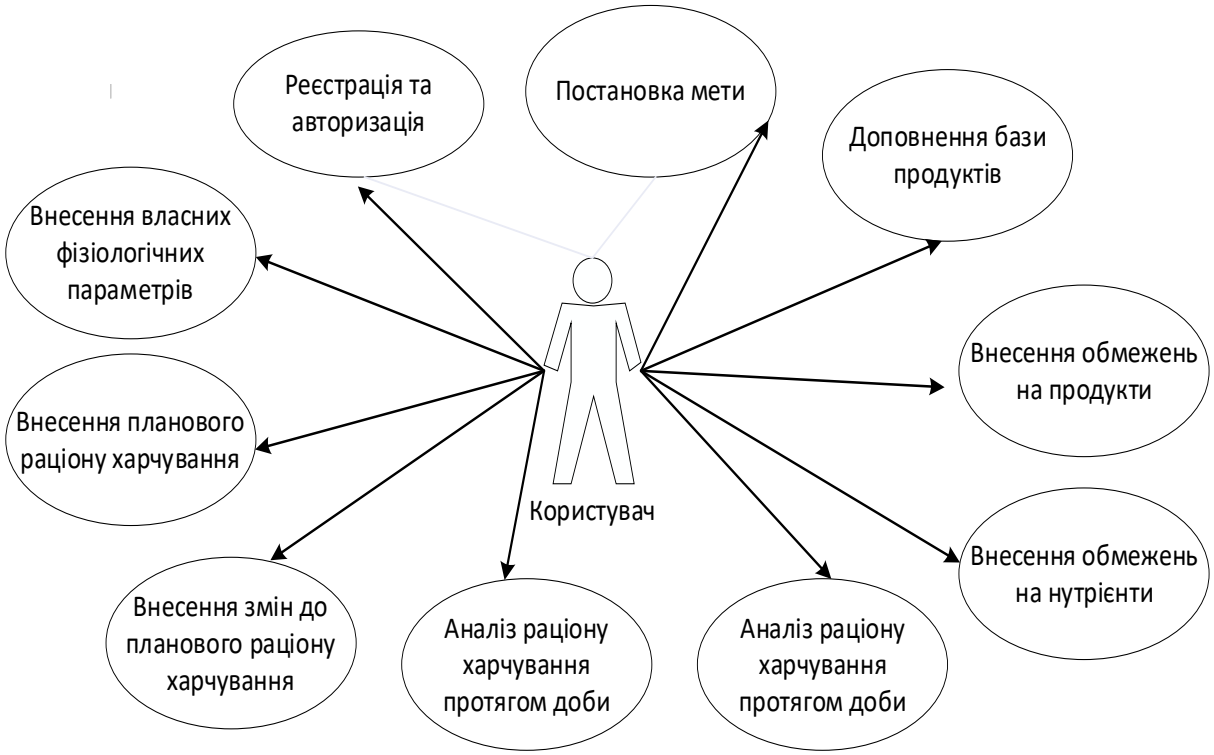



Рис.3.1. Діаграма прецедентів застосунку “AlterMeal”

Розглянемо більш детально кожен прецедент наведеної вище діаграми, а саме:

- Реєстрація та авторизація. Є обов’язковим для забезпечення збереження особистих даних в системі та зменшенням ризиків витоку цієї інформації.
- Внесення мети. Описується мета.
- Внесення власних фізіологічних параметрів. До таких параметрів відноситься вік (вноситься дата народження), зріст, вага, стать.

Можуть вноситися додаткові параметри за бажанням користувача. Ця інформація є умовно постійною і не вимагає щоденних змін.

- Внесення обмежень на продукти. Ця інформація вноситься коли є обмеження на певні продукти харчування (цільне молоко, молочні продукти загалом, на певні види м’яса тощо). Інформація є умовно-постійною. Може змінюватися при необхідності дотримання посту, зміни дієти, або виникненні інших обставин.
- Внесення обмежень щодо нутрієнтів. Є важливим для реалізації індивідуального підходу до раціону харчування. Особливо важливо для реалізації здорового способу життя, та врахування індивідуальної непереносимості або алергії.
- Внесення планового раціону харчування. Як правило, існуючі застосунки дають можливість фіксування спожитого раціону харчування, що не дає можливості скоректувати раціон. Саме планування дає можливість скоректувати раціон для здорового способу життя.
- Внесення змін до планового раціону харчування. Ця функціональна можливість дає гнучкість при різних обставинах мінливості життя людини і дозволяє робити реальне планування харчування.
- Фіксація реального раціону харчування. Користувач може перевести планові показники раціону харчування в спожиті, змінивши статус кожного запланованого блюда, або продукту.
- Аналіз раціону харчування протягом доби. Ця функція дозволяє проаналізувати раціон з точки зору обмежень, які вводив користувач та медичних рекомендацій.

Ескізний проєкт застосунку “AlterMeal” дає уяву про структуру ключових форм, які реалізують основну функціональність мобільного застосунку. Розглянемо екранні форми застосунку:

- Форма “Раціон” включає в себе всі записані за добу продукти.

- Форма “План”, де користувач може планувати певні продукти на перед, за потреби переносячи їх по одному кліку в “Раціон”.
- Форма “Продукти” містить базу даних продуктів, з якої можна додавати собі їжу в “План” або відразу в “Раціон”.
- Форма “Параметри” дозволяє вносити та переглядати фізіологічні параметри користувача з можливістю редагування або видалення.
- Форма “Аналіз”. Аналізується спожитий раціон за день з рекомендаціями та побажаннями.
- Форма “Входу та реєстрації”. Реєстрація є обов’язковою. Після реєстрації виникає тільки вікно входу до власного облікового запису.

### 3.2. Сценарії використання застосунку “AlterMeal”

Для опису сценаріїв будемо використовувати нотацію IDEF3, яка формалізує послідовність реалізацій можливих сценаріїв при виконанні певних регламентованих операцій, робіт, дій.

На рисунку 3.1 наведено сценарій, який описує реєстрацію та внесення особистих даних, а також обмежень для формування раціону харчування:

- створення нового облікового запису в застосунку;
- перевірку на існування облікового запису;
- заповнення програмної форми про особисті дані (“Параметри”);
- заповнення програмної форми, яка пов’язанні з обмеженнями на:
  – продукти;
  – нутрієнти;
  – калорії, білки, жири.

В екранній формі “План” передбачено планування майбутнього раціону на день. Планування передбачено для того, щоб користувач міг виявити перебирання калорій, або продуктів, які мають обмеження щодо нутрієнтів.

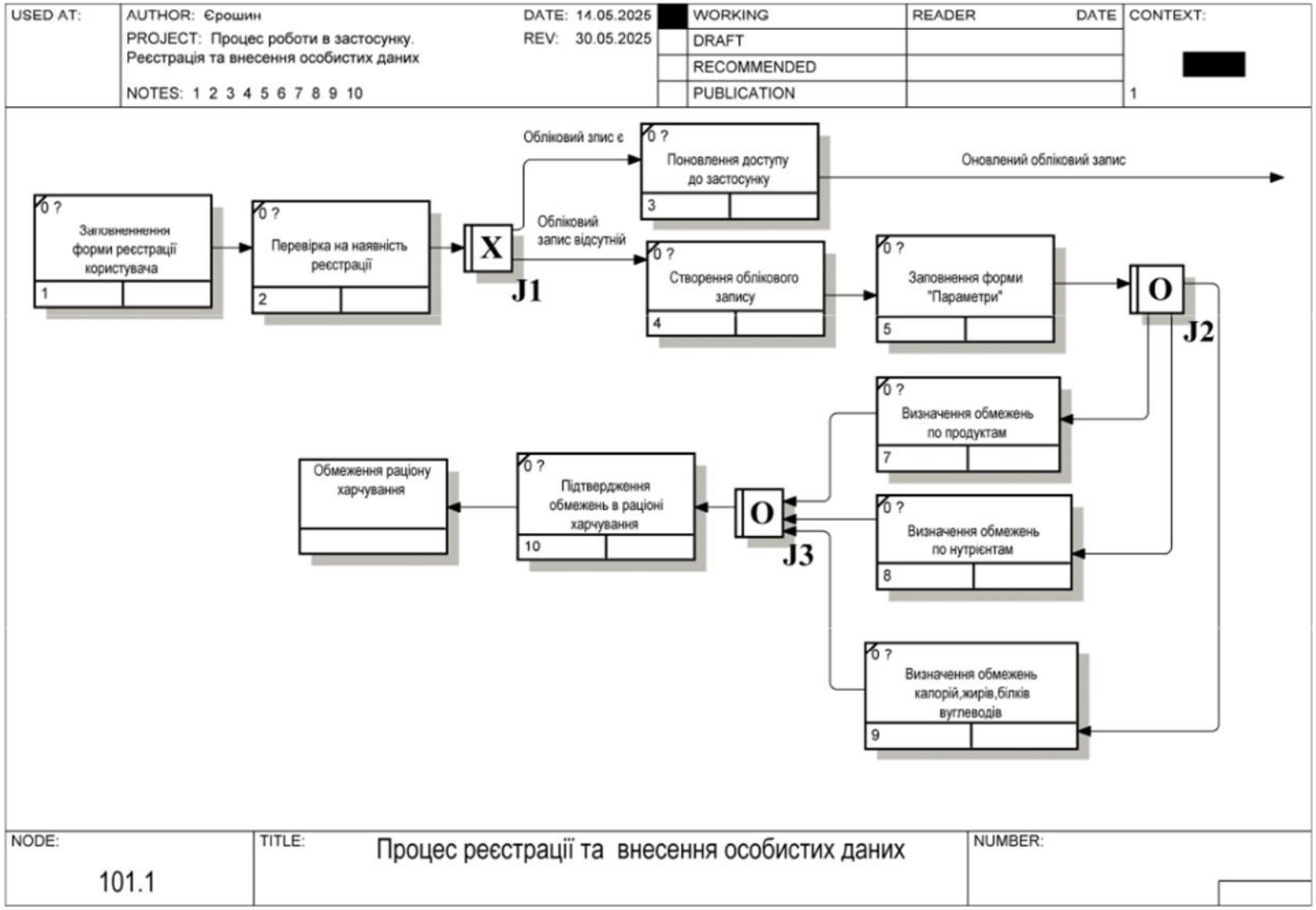



Рис. 3.1. Діаграма IDEF3. Реєстрація та внесення особистих даних

Процес формування щоденного раціону харчування відображений на рис. 3.2. Цей процес регламентує наступні дії користувача:

- користувач відкриває програмну форму планування раціону харчування на день;
- в цій формі користувач обирає продукти, які планує вживати протягом дня (для користувача доступні тільки ти продукти, які не підпадають під обмеження (рис. 3.1);
- при необхідності користувач може змінювати запланований раціон харчування протягом дня;
- при вживанні запланованих продуктів користувач відмічає це у застосунку.

Розглянемо більш детально основний сценарій використання застосунку “AlterMeal”, який реалізований в нотації UML (Unified Modeling Language) (рис. 3.3).

Користувач відкриває застосунок. Після цього автоматично починається перевірка на обмеження щодо продуктів харчування (якщо є відмітка про не відображення продуктів з обмеженнями). Якщо наявні обмеження по продуктам, то застосовується вибірка, яка виключає всі обмеження продуктів. Якщо обмеження щодо продуктів харчування відсутні, то автоматично починається перевірка на обмеження щодо нутрієнтів. Якщо наявні обмеження щодо нутрієнтів, то застосовується вибірка, яка виключає всі обмеження продуктів щодо нутрієнтів.

Користувач відкриває форму для планування харчового раціону на день. Починає вносити продукти в плановий раціон. Після закінчення вводу планового харчового раціону відбувається розрахунок калорій, жирів, білків, вуглеводів (КЖБВ). Якщо показники КЖБВ перевищують зазначені обмеження, то виникає попередження про це. Планування раціону діяльності харчування є ітераційним процесом. При виконанні умов щодо відповідності щодо КЖБВ процес планування переходить до фіксації спожитої їжі протягом дня. При цьому відбувається розрахунок КЖБВ.

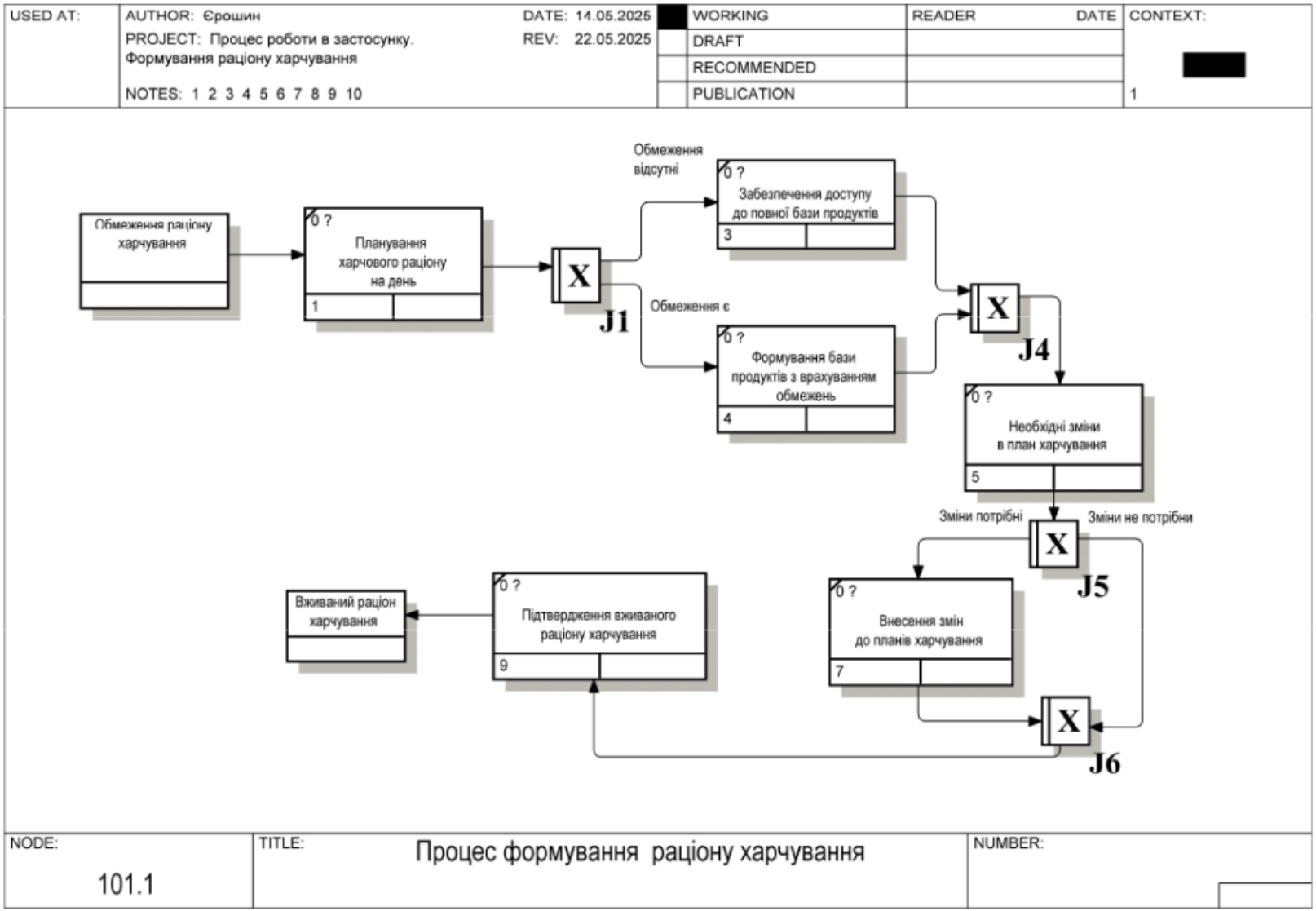



Рис. 3.2. Діаграма IDEF3. Формування раціону харчування

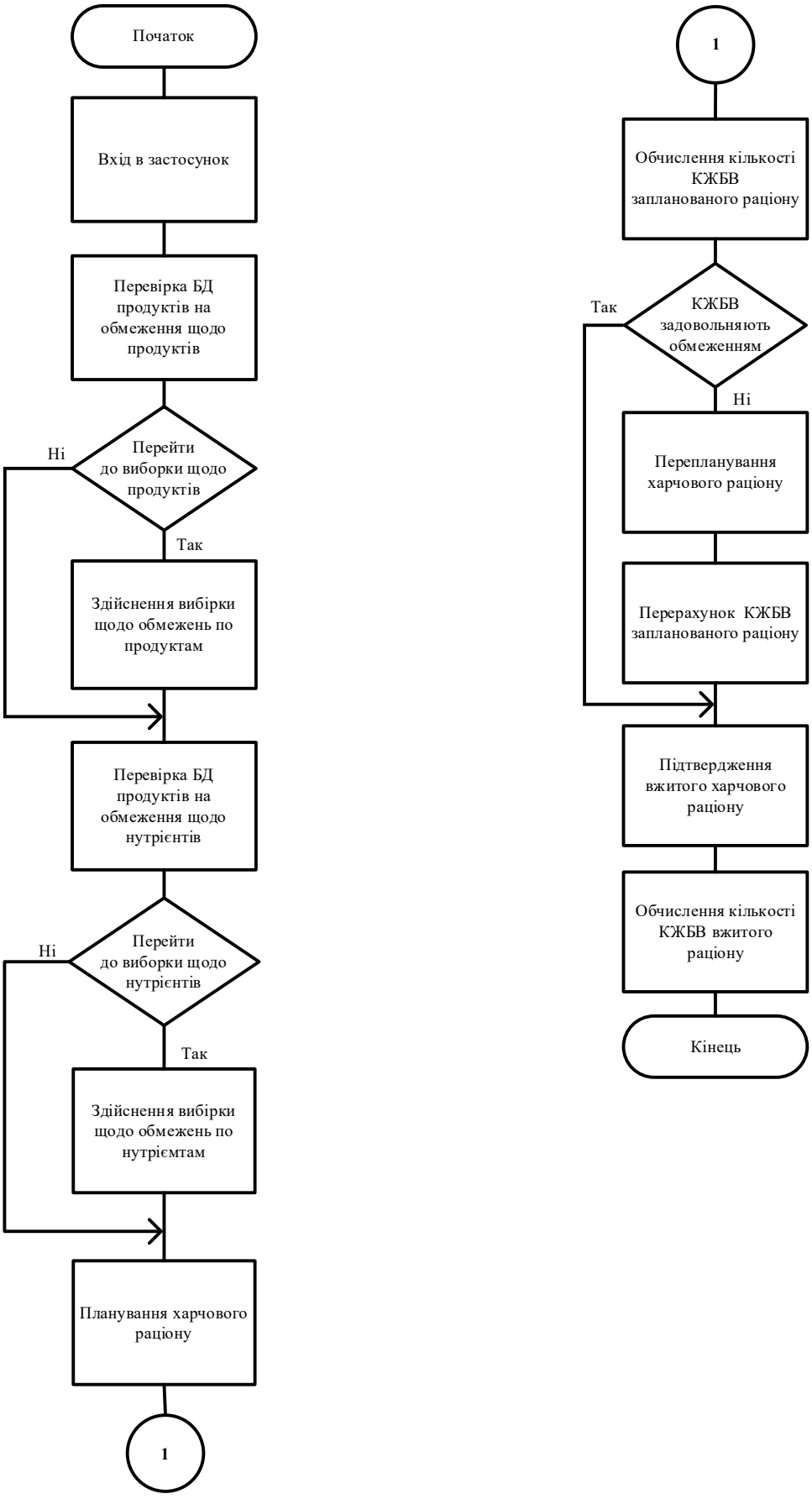



Рис. 3.3. Діаграма діяльності (UML- нотація) основного сценарію з автоматичною фільтрацією обмежень

Після закінчення фіксації спожитого раціону, відбувається збереження його не тільки на телефоні, а й на сервері в архів для подальшого аналізування за потреби.

### 3.3. Архітектура мобільний застосунку “AlterMeal”

У застосунку “AlterMeal” застосовано архітектурний патерн MVC (Model-View-Controller) [18]. Дана архітектура забезпечує розподіл між основними компонентами застосунку: Model, View і Controller. Такий підхід дозволяє чітко структурувати кодову базу (Front-end та Back-end), полегшує підтримку, тестування та майбутнє розширення функціональності. Розглянемо кожну компоненту більш детально.

Model – цей компонент відповідає за зберігання та обробку даних, що використовуються у застосунку. Сюди відноситься обробка даних про продукти, рецепти, цілі користувача, історію пошуку тощо. Model взаємодіє з Controller для виконання запитів на отримання, зміну чи збереження даних у локальній базі (SQLite).

View – цей компонент відповідає за відображення інформації та взаємодію користувача з інтерфейсом. У застосунку реалізовано низку екранних форм (екран планування раціону, перегляд раціону, введення обмежень тощо), кожна з яких відповідає за зручне і зрозуміле представлення даних. View отримує оновлення через Controller, а також ініціює події на внесення змін.

Controller – відповідає за взаємозв’язок між View та Model. Controller обробляє дії користувача (додавання продукту, розрахунок калорійності, планування та відстежування раціону), координує доступ до даних, виконує бізнес-логіку застосунку і забезпечує актуалізацію відображення в інтерфейсі. Окрім того Controller реалізує сценарії додавання, редагування або видалення даних, розрахунок калорійності, фільтрація та пошук інформації у локальній базі. Controllers приймають команди від інтерфейсу, взаємодіють із моделями та повертають результати для відображення.

Розглядаємо Пакет ui. Цей модуль містить компоненти, що безпосередньо взаємодіють із користувачем. Сюди входять окремі екрани застосунку, форми для введення даних, списки продуктів, а також елементи налаштувань. Кожен екран (перегляд раціону, або додавання страви) реалізовано як окремий клас із відповідними графічними елементами.

Пакет data. У цьому пакеті знаходяться всі компоненти для роботи з локальною базою даних SQLite: моделі даних (Product, Meal, Goal), допоміжні класи для з'єднання та запитів до БД, а також репозиторій, що забезпечує єдиний доступ до всіх CRUD-операцій (створення, читання, оновлення, видалення).

Для збереження щоденного харчового раціон з урахуванням id_u користувача та дати споживання дані передаються в архів на сервер за підтвердженням користувача. Обрана архітектура відповідає вимогам до приватності: всі дані зберігаються локально, кожен користувач має повний контроль над власною інформацією, має можливість зберігати архів шоденого споживання.

В застосунку використовується інформація Сільськогосподарської науково-дослідної служби (Agricultural Research Service) Міністерства сільського господарства США (U.S. Department of Agriculture), яка є у відкритому доступі.

На цьому етапі створення застосунок має спрощену клієнт-серверну архітектуру, тобто на сервері зберігається тільки архів, який буде у подальшому аналізуватися для створення рекомендацій з використанням штучного інтелекту (рис. 3.4).

Завдяки такій структурі забезпечується гнучкість і масштабованість застосунку. Додавання нових функцій ( впровадження підрахунку нових нутрієнтів, або додаткових типів раціонів) не потребує суттєвих змін у вже реалізованих компонентах. Кожна зона відповідальності чітко ізольована, що суттєво спрощує тестування, пошук і виправлення помилок.

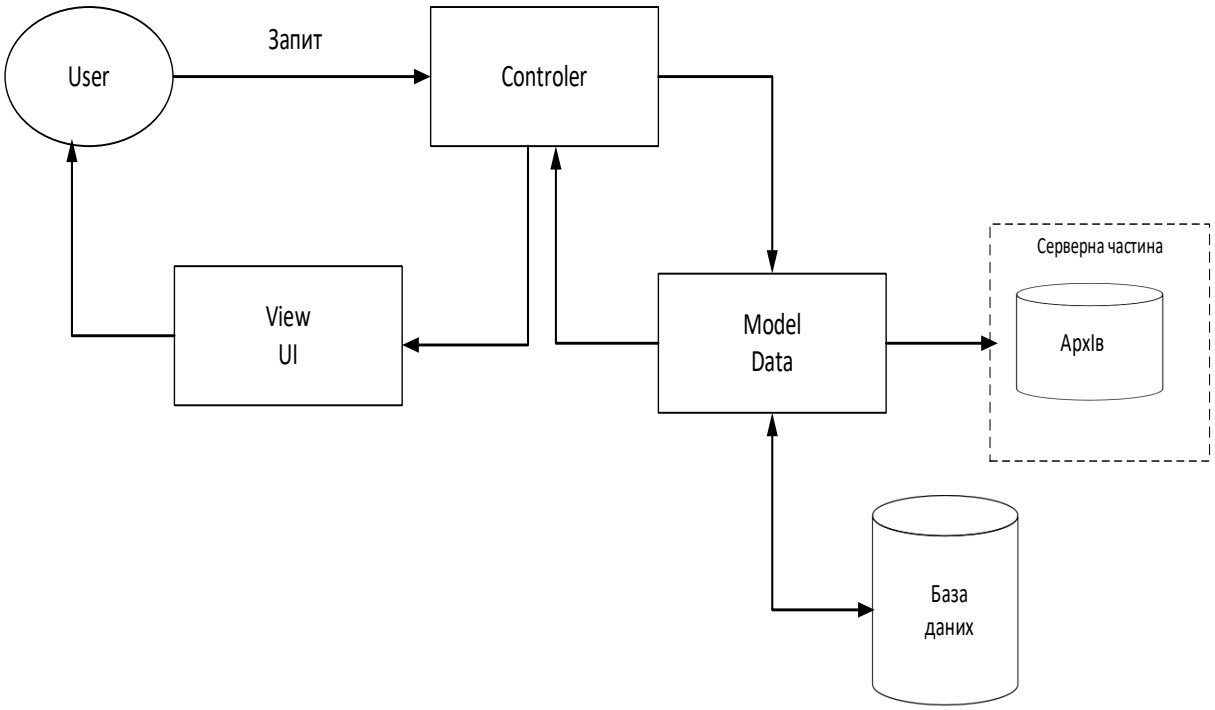



Рис. 3.4. Архітектура мобільного застосунку, яка базується на архітектурному патерні MVC

### 3.4. Структура бази даних мобільного застосунку “AlterMeal”

Завдяки використанню SQLite в мобільному застосунку забезпечується можливість зберігати повний перелік продуктів з описом харчової цінності та категоріями. SQLite є реляційною базою даних. На рисунку 3.5 неведена структура бази даних мобільного застосунку “AlterMeal”.

База даних містить  шість сутностей (таблиць). В таблиці бази User зберігаються незмінні дані про користувача (табл. 3.1). В свою чергу таблиця Personal (табл. 3.2) містить особисті дані користувача, які є змінними. Використання двох сутностей замість однієї дозволяє раціонально використовувати апаратні ресурси. Таблиці Product (табл.3.3) містять дані продуктової бази. В таблиці Diet (табл. 3.4) наведений харчовий раціон.  В таблиці Nutrient наведені дані по нутрієнтам (табл. 3.5). Дані про щоденний раціон зберігаються в таблиці Archive (табл. 3.6). Сутності пов’язані одна з одною реляційними зв’язками один до багатьох.

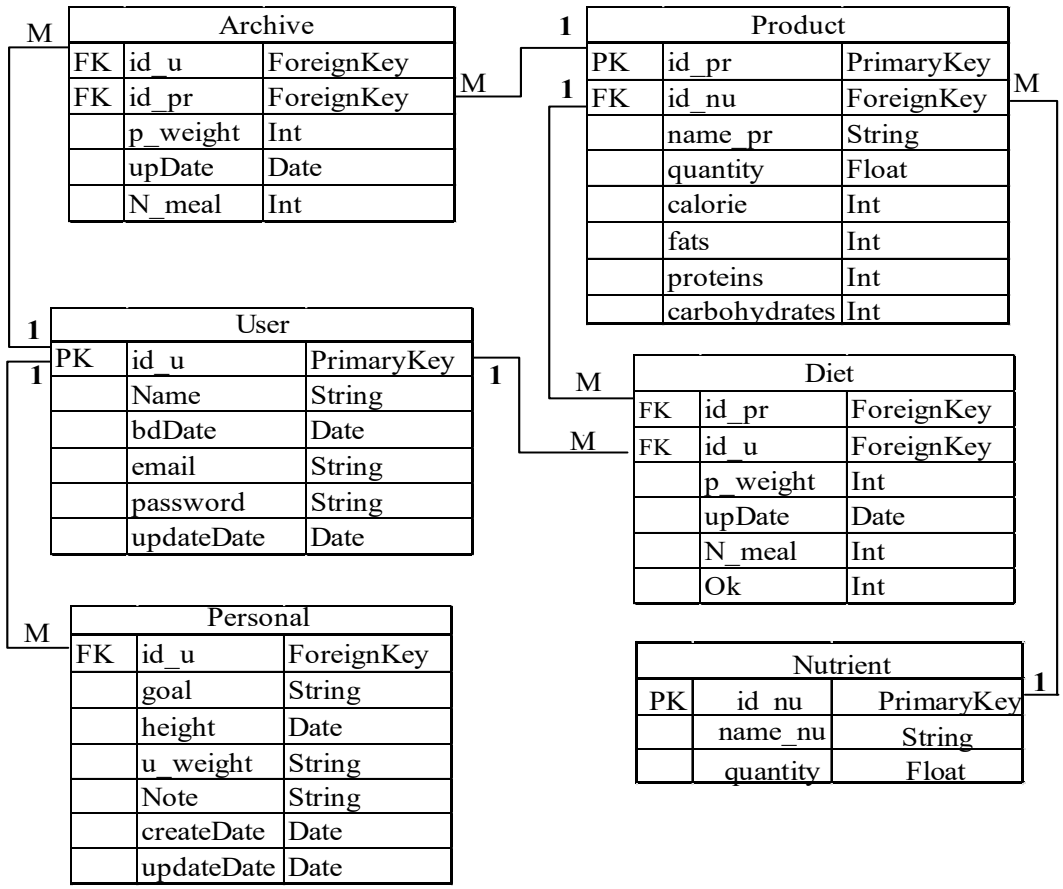



Рис. 3.5. База даних мобільного застосунку “AlterMeal”

Таблиця 3.1

Поля таблиці User

| *Назва поля* | *Опис* | *Тип даних* |
|---|---|---|
| id | Унікальний ідентифікатор користувача | PrimaryKey |
| Name | Ім’я користувача | String |
| bdDate | Дата народження | Date |
| email | Електронна пошта | String |
| password | Пароль | String |
| createDate | Дата створення | Date |
| updateDate | Дата оновлення | Date |

Таблиця 3.2

Поля таблиці Personal

| *Назва поля* | *Опис* | *Тип даних* |
|---|---|---|
| id | Унікальний ідентифікатор користувача | ForeignKey |
| goal | Мета користувача | String |
| height | Зріст користувача | Date |
| u_weight | Вага користувача | String |
| Note | Додаткова інформація | String |
| createDate | Дата створення | Date |
| updateDate | Дата оновлення | Date |

Таблиця 3.3

Поля таблиці Product

| *Назва поля* | *Опис* | *Тип даних* |
|---|---|---|
| id_pr | Унікальний ідентифікатор продукта | PrimaryKey |
| id_nu | Унікальний ідентифікатор нутрієнта | ForeignKey |
| name_pr | Назва продукту | String |
| quantity | Кількість нутрієнта на 100 гр. продукта | Float |
| calorie | Кількість калорій на 100 гр. продукта | Int |
| fats | Кількість жирів на 100 гр. продукта | Int |
| proteins | Кількість білків на 100 гр. продукта | Int |
| carbohydrates | Кількість білків на 100 гр. продукта | Int |

Таблиця 3.4

Поля таблиці Diet

| *Назва поля* | *Опис* | *Тип даних* |
|---|---|---|
| Id_u | Унікальний ідентифікатор користувача | PrimaryKey |
| id_pr | Унікальний ідентифікатор продукта | ForeignKey |
| p_weight | Вага спожитого продукта в грамах | Int |
| upDate | Дата оновлення харчового раціону | Date |
| N_meal | Номер прийому їжі | Int |
| Ок | Відмітка про прийом їжі | Int |

Таблиця 3.5

Поля таблиці Nutrien

| *Назва поля* | *Опис* | *Тип даних* |
|---|---|---|
| id_nu | Унікальний ідентифікатор нутрієнта | PrimaryKey |
| name_pr | Назва нутрієнта | String |
| quantity | Одиниця виміру нутрієнта | Float |

Таблиця 3.6

Поля таблиці Archive

| *Назва поля* | *Опис* | *Тип даних* |
|---|---|---|
| Id_u | Унікальний ідентифікатор користувача | ForeignKey |
| id_pr | Унікальний ідентифікатор продукта | ForeignKey |
| p_weight | Вага спожитого продукта в грамах | Int |
| upDate | Дата оновлення харчового раціону | Date |
| N_meal | Номер прийому їжі | Int |

### 3.5. Висновки до розділу

Ескізний проєкт застосунку “AlterMeal” дає уяву про структуру ключових форм (сторінок), які реалізують основну функціональність мобільного застосунку. Відповідно до дерева проблем було створено діаграму прецедентів, яка відображає функціональні можливості застосунку.

Для опису сценаріїв будемо використовувати нотацію IDEF3, яка формалізує послідовність реалізацій можливих сценаріїв при виконанні певних регламентованих операцій, робіт, дій. Окрім того, створена діаграма діяльності, яка описує основний алгоритм дій користувача в застосунку.

У мобільному застосунок “AlterMeal” застосовано архітектурний патерн Model-View-Controller. MVC архітектура забезпечує розподіл між основними компонентами застосунку: Model, View і Controller. Розділення логіки на окремі модулі та компоненти відповідно до принципів MVC дозволяє легко підтримувати й розширювати функціональність застосунку. Основні дані представлені у вигляді моделей (продукти, користувач, план харчування), а вся логіка щодо їх зберігання та обробки реалізована за допомогою відповідних класів-репозиторіїв.

Такий підхід дозволяє чітко структурувати кодову базу (Front-end та Back-end), полегшує підтримку, тестування та майбутнє розширення функціональності. Розглянемо кожну компоненту більш детально

Використанню SQLite в мобільному застосунку забезпечується можливість зберігати повний перелік продуктів з описом харчової цінності та категоріями. SQLite є реляційною базою даних. Структура бази даних мобільного застосунку “AlterMeal” складається з шести сутностей, які пов’язані між собою зв’язками один до багатьох. Кожна з таблиць має власну структуру та набір полів для зберігання інформації про продукти, користувачів, записи раціону тощо. Така організація дозволяє швидко виконувати пошук, фільтрацію та аналіз даних.

Застосунок “AlterMeal” забезпечує повну автономність роботи, високий рівень приватності, зручність користування та можливість гнучкого налаштування під потреби кожного користувача, що є ключовими факторами його ефективності у сфері мобільних застосунків для здорового способу життя.

# 4. АНАЛІЗ РОЗРОБЛЕНОГО МОБІЛЬНОГО ЗАСТОСУНКУ

У мобільному застосунку “AlterMeal” уся логіка, інтерфейс та обробка даних реалізовані локально на мобільному пристрої користувача за допомогою кросплатформенної технології Flutter. Застосунок не розділяється на класичні клієнтську і серверну частини, а натомість побудований як повністю автономний мобільний застосунку, у якому вся робота з даними та бізнес-логіка відбуваються на пристрої користувача. Всі дані про продукти, налаштування, харчові обмеження та історію раціону зберігаються у локальній реляційній базі даних SQLite, що забезпечує швидкий доступ, безпеку та приватність персональної інформації. На серверній частині зберігається архів щоденного продуктового раціону.

Особливістю застосунку є можливість розрахунку індивідуальної добової потреби в калоріях з урахуванням віку, статі, ваги, зросту, рівня фізичної активності та цілі користувача (схуднення, набір маси, підтримка ваги тощо). Це дає змогу формувати персоналізовані раціони й отримувати рекомендації щодо оптимального харчування.

## 4.1. Опис інтерфейсу користувача

Інтерфейс мобільного застосунку “AlterMeal” реалізовано з урахуванням сучасних принципів зручності для користувача. Вся функціональність розподілена між кількома основними формами, що відповідають за ключові сценарії роботи з додатком.

Ключові екранні форми застосунку охоплюють перегляд і планування раціону, сторінку з додаванням продуктів та страв, екран аналізу харчування, а також розділ налаштувань, де користувач може вказати власні фізіологічні параметри, цілі, харчові обмеження й алергії. Авторизація і створення облікового запису також реалізовані у мобільному застосунку, що дозволяє індивідуалізувати рекомендації та відстежувати власний прогрес.

Форма “Реєстрації” виступає першим екраном для нового користувача (рис. 4.1). В цій формі користувачу необхідно пройти реєстрації облікового запису, або, за наявпості акаунту, виконати вхід.

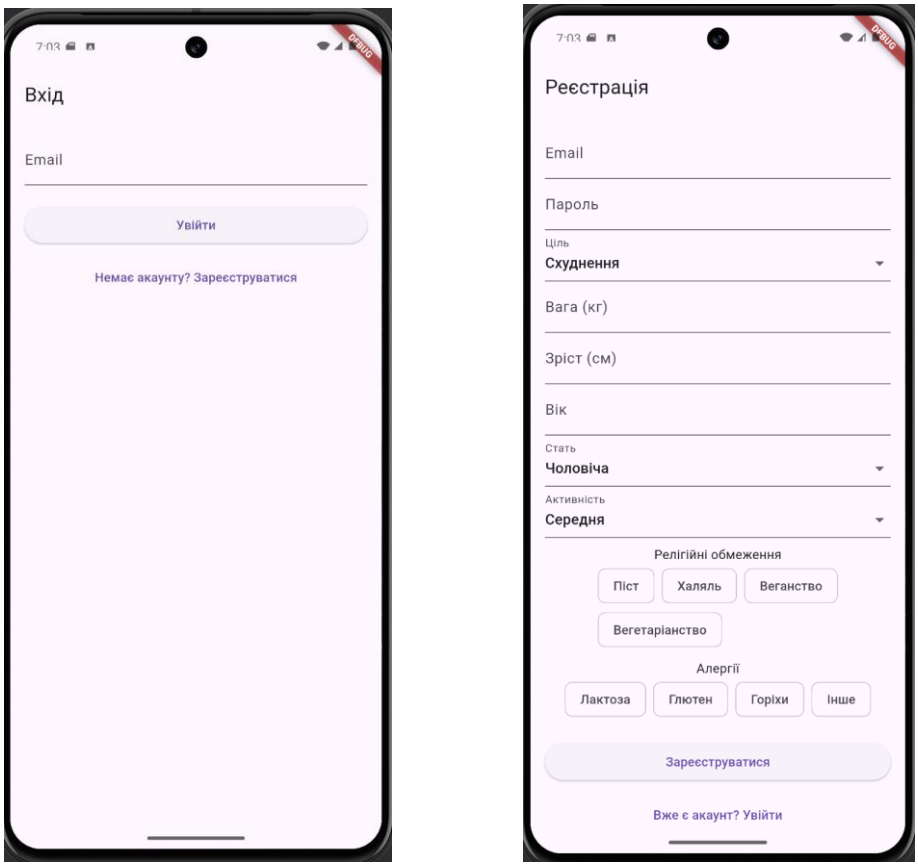



Рис. 4.1. Форма реєстрації або входу в застосунок

Форма “Раціон” призначена для ведення щоденного харчового раціону (рис. 4.2). Тут користувач може додавати продукти до свого раціону, переглядати калорійність кожної порції, а також бачити загальну кількість спожитих калорій за день у порівнянні з особистою добовою нормою. У нижній частині цієї форми відображається прогрес виконання норми калорій із кольоровою індикацією (зелений, жовтий, червоний), що дозволяє користувачу швидко орієнтуватися у відповідності фактичного харчування до поставленої цілі.

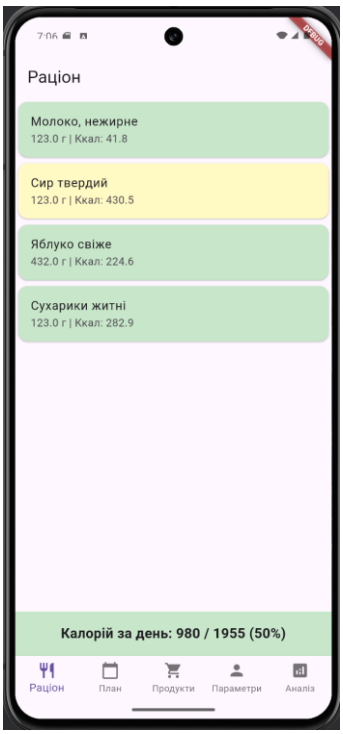



Рис. 4.2. Форма “Раціон” для ведення щоденного раціону

Форма “План” (рис. 4.3) дає можливість планувати майбутнє харчування. Тут користувач може сформувати перелік страв і продуктів, які планує вживати, що дозволяє краще контролювати раціон і робити відповідальний вибір.

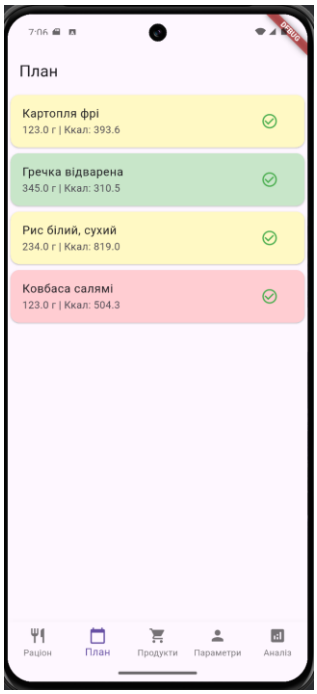



Рис. 4.3. Форма “План” для планування щоденного раціону

Форма “Продукти” містить базу даних продуктів харчування з інформацією про калорійність, білки, жири, вуглеводи та інші нутрієнти (рис. 4.4). У цій формі реалізовано можливість пошуку, фільтрації та додавання обраного продукту до раціону чи плану харчування. Для кожного продукту також відображається підказка щодо його відповідності калорійності категорії та персональним обмеженням користувача.

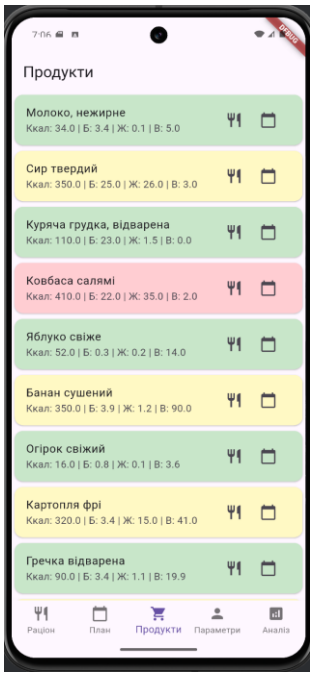



Рис. 4.4. Форма “Продукти” для план щоденного раціону

Форма “Параметри” призначена для налаштування особистих даних користувача, таких як вага, зріст, вік, стать, рівень фізичної активності та ціль (схуднення, набір маси тощо). Окрім того, у цій формі користувач може вказати власні харчові та релігійні обмеження, алергії, а також змінити e-mail (рис. 4.5).

Форма “Аналіз” виконує аналітичні функції: аналізує раціон і план користувача щодо відповідності поставленій цілі, наявності продуктів, що порушують зазначені обмеження або алергії (рис. 4.6). Якщо у раціоні чи плані виявлені невідповідності, користувач отримує зрозуміле попередження.

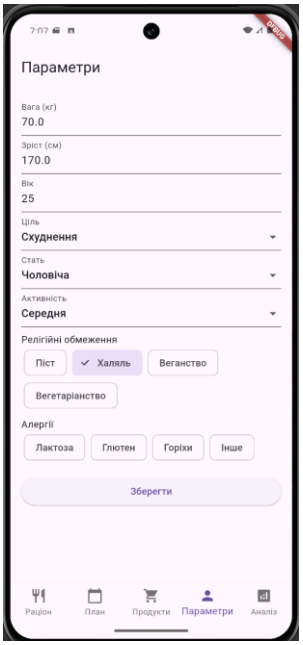



Рис. 4.5. Форма “Параметри” для внесення особистих даних

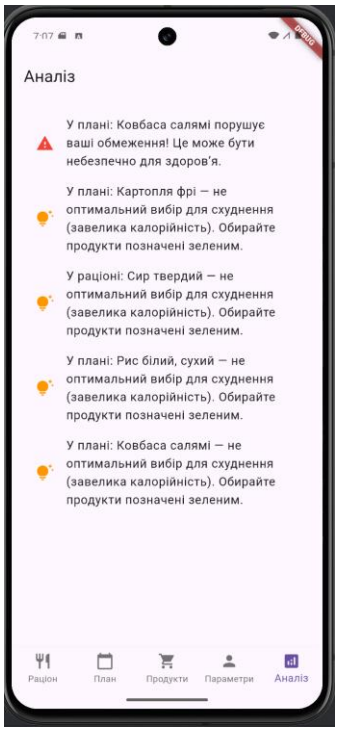



Рис. 4.6. Форма “Аналіз” для аналізу невідповідностей

Екранні форми реалізовано з використанням адаптивних елементів інтерфейсу, які коректно відображаються на різних розмірах екранів. Колірна індикація та зручні інструменти вибору (мульти-вибір алергенів або

обмежень) дозволяють швидко взаємодіяти із застосунком навіть недосвідченим користувачам.

Завдяки запропонованій структурі інтерфейсу, застосунок забезпечує зручний доступ до всіх функцій і дозволяє легко відслідковувати стан свого харчування, дотримуватись обмежень і досягати поставлених цілей.

### 4.2. Тестування мобільного застосунку

У процесі розробки мобільного застосунку “AlterMeal” велика увага приділялась якісному тестуванню, щоб забезпечити коректну роботу всіх функцій, коректність роботи інтерфейсу та відсутність критичних помилок. Тестування виконувалося комбіновано – як вручну, так і за допомогою автоматизованих юніт-тестів.

Ручне тестування проводилося на різних пристроях та емуляторах. Спочатку перевірялась робота всіх основних форм застосунку – екрану реєстрації та входу, налаштувань користувача, додавання продуктів до раціону й плану, функцій видалення та редагування даних, а також коректність підрахунку калорій. Окрема увага приділялася перевірці логіки обмежень: для тесту створювалися різні користувачі із зазначенням алергій та релігійних обмежень, після чого вносилися відповідні продукти до списку – це дозволяло переконатися, що система правильно попереджає користувача у випадку невідповідності продукту його обмеженням.

У ході тестування також моделювались різні користувацькі сценарії: зміна параметрів (вага, зріст, вік, стать, активність, ціль), граничні та некоректні значення, робота індикатора норми калорій, швидкість оновлення інформації при додаванні або видаленні продуктів. Значна увага приділялася зручності інтерфейсу – всі повідомлення, підказки та помилки тестувались на предмет зрозумілості та помітності для користувача. Результати тестування наведені в табл. 4.1.

Методи тестування включали тестування за сценаріями (усі можливі шляхи використання застосунку: від початкової реєстрації до щоденного використання), тестування на стійкість до помилок (введення некоректних

Таблиця 4.1

Таблиця сценаріїв для тестування застосунку

| № | Мета тест-кейсу | Дія | Очікуваний результат |
|---|---|---|---|
| 1. | Перевірка заповнення повноти форми реєстрації | Користувач заповнює не всі поля | Відкривається повідомлення “Заповніть всі поля” |
| 2. | Перевірка наявності облікового запису | Вводиться email, користувач, який вже є | Відкривається повідомлення “Реєстрація вже є” |
| 3. | Перевірка заповнення форми “Параметри” | Вводяться особисті дані користувача. Зберігаються | Коректно відображається та зберігається дані щодо користувача |
| 4. | Перевірка відповідності зображення форми “Продукти” | Відкривається форма “Продукти” | Форма “Продукти” повністю відображає відповідний вміст |
| 5. | Перевірка виділення продуктів, або нутрієнтів в формі “Продукти”, які мають обмеження | Відкривається форма “Продукти” | Відображаються дані в формі “Продукти” з підсвіткою обмежень |
| 6. | Перевірка спроможності вносити в форму “План” обрані планові продукти харчування | Внести в форму “План” продукти, які планується вживати. Зберегти заплановані продукти | Коректно відображаються дані в формі “План” |

Продовження табл. 4.1

| | | | |
|---|---|---|---|
| 7. | Перевірка можливості зберігати та змінювати форми “План” | Внести в форму “План” зміни до вибраних продуктів. Зберегти змінені заплановані продукти | Коректно відображаються змінені дані в формі “План” |
| 8. | У формі “План” перевіряється можливість пошуку, фільтрації продуктів харчування | У формі план зробити фільтрацію продуктів | Продукти у формі відображаються у фільтрованому вигляді |
| 9. | У формі “Раціон” та додавання обраного продукту до раціону чи плану харчування | Відкривається форма “Раціон”. Додається продукт харчування | Всі продукти доступні і відображаються корктно |
| 10. | У формі “Раціон” перевіряється можливість пошуку, фільтрації продуктів харчування | У формі “Раціон” зробити фільтрацію продуктів | Продукти у формі “ Раціон” відображаються у фільтрованому вигляді |
| 11. | Перевірити роботу форми “Аналіз” на обмеження продуктів | Вводяться продукти із списку обмежень | В формі “Аналіз” виводиться інформація про вибір продукту з обмеження |
| 12. | Перевірити роботу форми “Аналіз” на обмеження КЖБВ | Користувач вводить в форму “План” продукти х перевищенням КЖБВ | В формі “Аналіз” виводиться інформація про перевищення КЖБВ |

В результаті ручного тестування застосунку були виявлення помилки. На рис. 4.7 наведено некоректне зображення форми “Продукти”, яке було виправлено.

Юніт-тести було розроблено для найбільш важливих частин логіки застосунку [19]. Зокрема, тестувалися функції розрахунку добової норми калорій з урахуванням усіх заданих параметрів користувача, а також функції перевірки відповідності продуктів обмеженням користувача (алергії та релігійні обмеження). Це дозволило автоматично перевіряти правильність основної бізнес-логіки навіть при подальших змінах у коді.

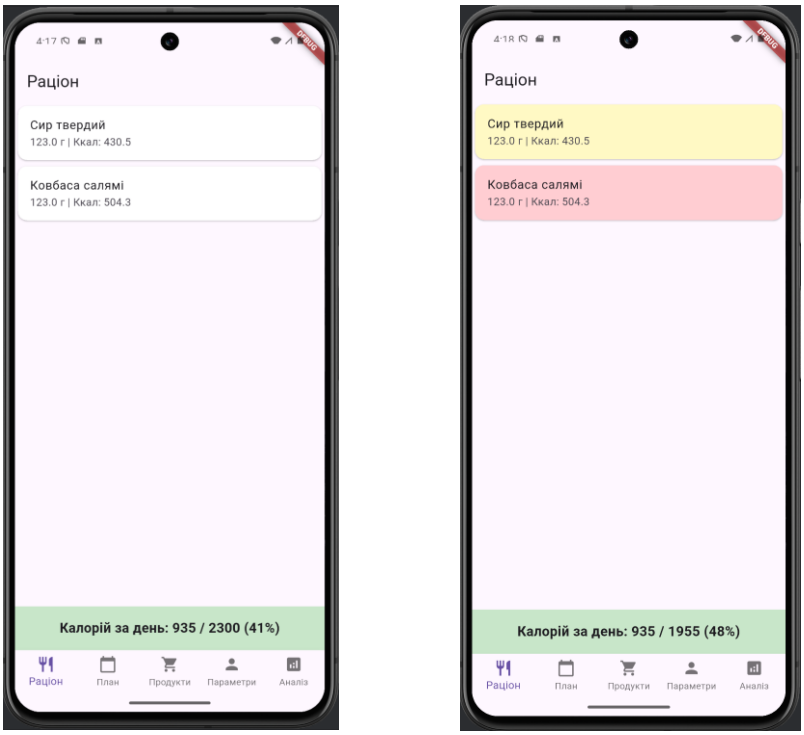



Рис. 4.7. Виявлення помилок та виправлення

Запуск юніт-тестів здійснювався за допомогою стандартного інструментарію Flutter (flutter_test). Для цього у проекті було створено окремий файл із тестами.

Результати виконання тестів відразу відображали успішність чи наявність помилок у конкретних функціях, що дозволяло швидко реагувати та усувати недоліки.

Такий підхід до тестування забезпечив впевненість у стабільності, надійності та зручності використання застосунку “AlterMeal”. Проведення як ручного, так і автоматизованого тестування дозволило виявити і усунути

основні помилки ще на етапі розробки, а також сформувати продукт, який дійсно відповідає очікуванням користувачів.

### 4.3. Пропозиції для майбутнього покращення застосунку

У процесі розробки та тестування мобільного застосунку “AlterMeal” було визначено низку напрямків, які можуть стати основою для подальшого розвитку та вдосконалення функціональності. Застосунок уже виконує основні завдання щодо підрахунку калорій, персоналізації раціону та врахування особистих обмежень користувача. Проте для підвищення цінності продукту та забезпечення ще більшого користувацького досвіду доцільно розглянути такі майбутні покращення:

- Можливості додавання власних продуктів. На даний момент користувач може працювати лише з базою даних, яка є у застосунком. Запровадження функцій додавання, редагування та видалення власних продуктів дозволить враховувати унікальні продукти та створювати страви, які не присутні у стандартній базі.
- Розширення бази рецептів та автоматичний підбір альтернатив. Можливість додавати власні рецепти, зберігати їх та ділитися з іншими користувачами зробить застосунок більш інтерактивним і персоналізованим. Також можна впровадити систему рекомендацій альтернативних продуктів або страв відповідно до заданих користувачем медичних, релігійних чи смакових обмежень.
- Синхронізація даних із хмарними сервісами. Додавання функції резервного копіювання і синхронізації раціону та налаштувань у хмарі дозволить користувачам відновлювати свої дані на різних пристроях, а також не втрачати інформацію у разі зміни або втрати телефону.
- Інтеграція з фітнес-трекерами та іншими застосунками здоров'я. Синхронізація із популярними фітнес-трекерами чи платформами, такими як Google Fit або Apple Health, надасть можливість

автоматично враховувати рівень фізичної активності користувача та більш точно підраховувати денну норму калорій.

- Додаткові аналітичні інструменти. Запровадження розширеної статистики, графіків змін ваги, динаміки споживання макронутрієнтів та досягнення цілей дозволить користувачеві краще відслідковувати свої результати та приймати обґрунтовані рішення щодо харчування.
- Багатомовна підтримка. Реалізація можливості змінювати мову інтерфейсу розширить потенційну аудиторію застосунку та зробить його доступнішим для користувачів з різних країн.
- Доступність для людей з особливими потребами. Покращення навігації, збільшення розміру шрифтів, голосові підказки – усе це сприятиме тому, щоб “AlterMeal” був зручний для максимально широкого кола користувачів.

Таким чином, врахування зазначених напрямків удосконалення дозволить зробити мобільний застосунок “AlterMeal” ще більш функціональним, інтуїтивним і корисним для різних категорій користувачів, а також зміцнити його конкурентні переваги на ринку мобільних рішень для здорового харчування.

### 4.4. Висновки до розділу

У цьому розділі було детально розглянуто інтерфейс користувача мобільного застосунку “AlterMeal”, основні форми для взаємодії з користувачем, а також підходи до тестування розробленого програмного продукту. В результаті реалізації застосунку вдалося створити інтуїтивно зрозумілий та зручний інтерфейс, який відповідає сучасним вимогам до мобільних застосунків та дозволяє користувачам легко виконувати основні дії: додавати продукти, планувати раціон, обирати альтернативи з урахуванням різноманітних обмежень та переглядати статистику.

Особливу увагу було приділено перевірці працездатності застосунку шляхом ручного та автоматизованого тестування. Проведене тестування дозволило виявити і виправити низку недоліків, а також переконатися у надійності основної функціональності продукту. Окремо було впроваджено юніт-тести для ключових частин бізнес-логіки, що підвищило якість коду та спростило подальшу підтримку й розвиток застосунку.

Таким чином, реалізований застосунок демонструє швидкодію, забезпечує високий рівень зручності для кінцевого користувача та має потенціал для подальшого розвитку і вдосконалення.

# ВИСНОВКИ

У дипломному проєкті було розроблено мобільний застосунок “AlterMeal” для КЖБВ і формування раціону з урахуванням індивідуальних потреб та обмежень користувача. В ході виконання роботи було проведено аналіз проблеми інтерактивного підбору харчування, виявлення та аналіз вимог до застосунку, сучасних засобів розробки мобільних застосунків, обґрунтовано вибір технологій та архітектурних рішень, розроблено зручний і функціональний інтерфейс, а також реалізовано основну бізнес-логіку.

Метою дипломного проєкту є створення мобільного застосунку “AlterMeal” для інтерактивного підбору харчування і формуванню харчового раціону, який враховує індивідуальні потреби користувача, медичні, релігійні, інші обмеження, а також забезпечить планування та відстежування раціонну харчування.

Розроблено, ескізний проєкт застосунку “AlterMeal”, що дає уяву про структуру ключових екраних форм (сторінок), які реалізують основну функціональність мобільного застосунку. Відповідно до дерева проблем було створено діаграму прецедентів, яка відображає функціональні можливості застосунку.

Застосунок забезпечує автоматичний розрахунок калорійності продуктів і страв, дозволяє враховувати релігійні, медичні та інші обмеження, а також обирати раціон згідно особистих цілей (схуднення, набір маси, підтримка ваги тощо). Інтерфейс застосунку побудований з урахуванням принципів зручності для користувача. Усі основні сценарії – реєстрація, введення персональних даних, додавання продуктів у раціон чи план, перегляд аналітики та контроль досягнення цілей – були реалізовані й протестовані як вручну, так і за допомогою юніт-тестів.

У мобільному застосунок “AlterMeal” застосовано архітектурний патерн Model-View-Controller. MVC архітектура забезпечує розподіл між

основними компонентами застосунку: Model, View і Controller. Розділення логіки на окремі модулі та компоненти відповідно до принципів MVC дозволяє легко підтримувати й розширювати функціональність застосунку.

Розроблена структура коду та запропонована архітектура застосунку забезпечують гнучкість і можливість подальшого розширення функціональності, зокрема: впровадження функції додавання власних продуктів і страв, розширення бази даних, інтеграція з хмарними сервісами, фітнес-трекерами та розвиток аналітики для користувача.

Отже, результати виконаної роботи повністю відповідають поставленим завданням і підтверджують доцільність впровадження інноваційних ІТ-рішень для полегшення процесу контролю харчування та підтримки здорового способу життя за допомогою мобільних застосунків.

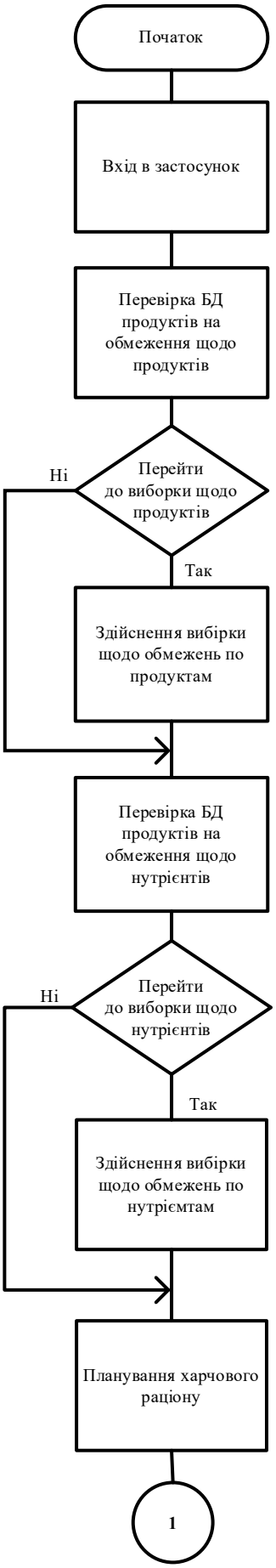
## СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ


1. Дуденко Н. В. Нутриціологія : навч. посіб. / Н. В. Дуденко, А. Ф. Павлоцька, І. В. Цихановська. – К. : Світ книг, 2022. – 527 с.
2. Тележенко Л. М. Здорове харчування: практичні рекомендації / Л. М. Тележенко, Н. А. Дзюба, М. А. Кашкано. – Олді, 2018. – 200 с.
3. Пахомська, О. В.  Харчові продукти – проблеми якості та безпечності. Науковий вісник ТДАТУ, Вип. 11, Т. 2. С. 333-342
4. Yazio [Електронний ресурс]. – Режим доступу: https://www.yazio.com/en – Дата доступу: лютий 2025.
5. MyFitnessPal [Електронний ресурс]. – Режим доступу: https://www.myfitnesspal.com – Дата доступу: лютий 2025.
6. Таблиця калорійності [Електронний ресурс]. – Режим доступу: https://www.tablycjakalorijnosti.com.ua– Дата доступу: лютий 2025.
7. C# language documentation [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/ – Дата доступу: березень 2025.
8. Java [Електронний ресурс]. – Режим доступу: https://docs.oracle.com/en/java/ – Дата доступу: березень 2025.
9. Kotlin [Електронний ресурс] – Режим доступу:  https://kotlinlang.org/# – Дата доступу: березень 2025.
10. Dart [Електронний ресурс] – Режим доступу: https://dart.dev/docs – Дата доступу: березень 2025.
11. JavaScript Tutorial [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/js/default.asp – Дата доступу: березень 2025.
12. Swift. [Електронний ресурс]. – Режим доступу: https://developer.apple.com/swift/PHP – Дата доступу: квітень 2025.
13. C++ [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/ – Дата доступу: квітень 2025.

14. Unity [Електронний ресурс]. –Режим доступу: https://docs.unity3d.com/ – Дата доступу: квітень 2025.
15. Flutter [Електронний ресурс]. – Режим доступу: https://flutter.dev – Дата доступу: квітень 2025.
16. PostgreSQL [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/ – Дата доступу: квітень 2025.
17. SQLite [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/about.html – Дата доступу: квітень 2025.
18. MVC [Електронний ресурс]. – Режим доступу: https://www.yiiframework.com/doc/guide/ – Дата доступу: квітень 2025.
19. Unit Testing – Software Testing [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/unit-testing-software-testing/ – Дата доступу: травень 2025.

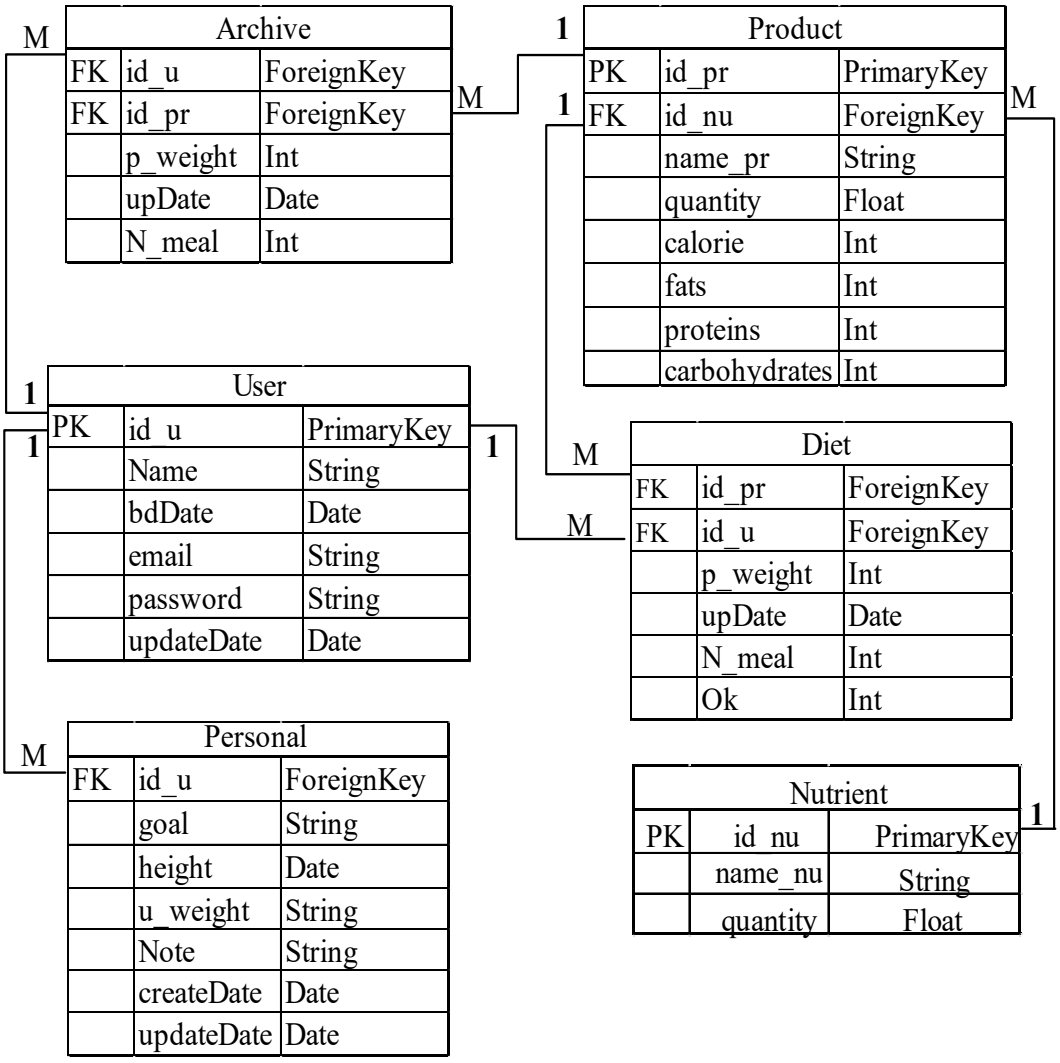
# ДОДАТКИ

**Додаток 1**
**Копії графічних матеріалів**





ДП.045440-06-99
Мобільний застосунок для інтерактивного підбору харчування. Діаграма діяльності основного сценарію використання. Схема алгоритму

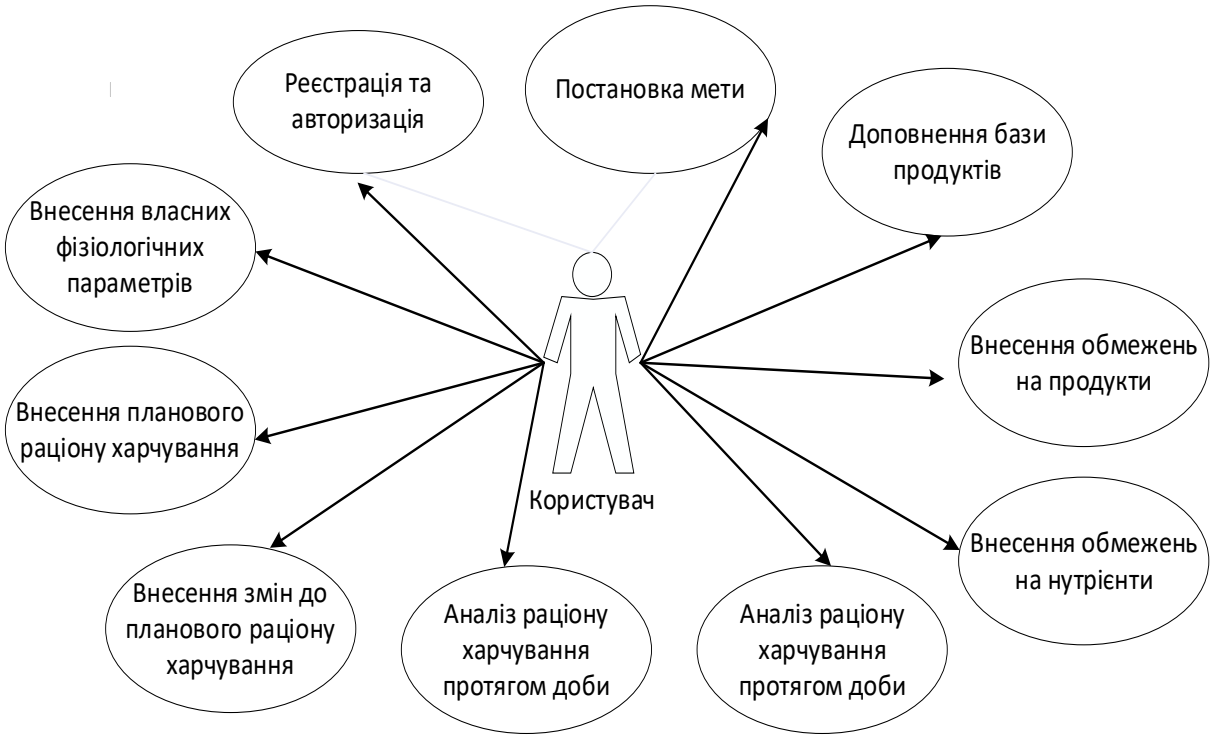



ДП.045420-07-99

Мобільний застосунок для інтерактивного підбору харчування.

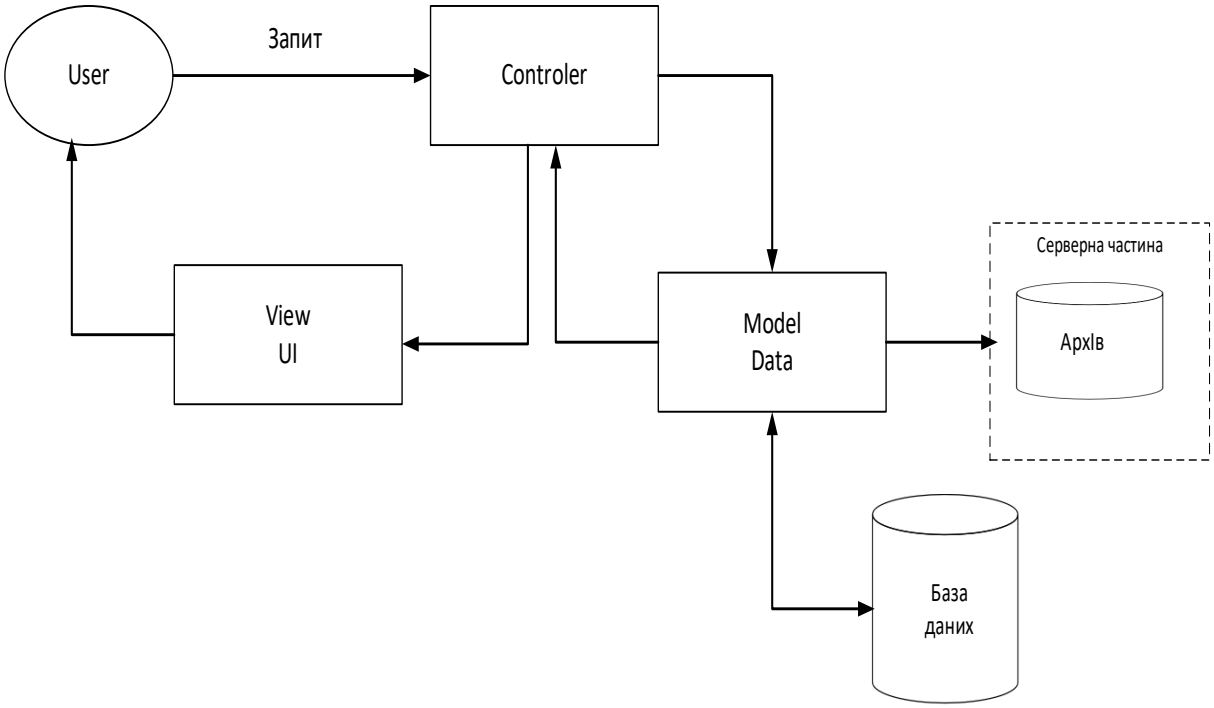
Логічна модель бази даних. Схема даних

**Діаграма прецедентів. “Мобільний застосунок для інтерактивного підбору харчування”**




Єрошин Богдан, група КП-13

**Архітектура мобільного застосунку для інтерактивного підбору харчування**




Єрошин Богдан, група КП-13

**Додаток 2**
**Лістинг програми**

## Фрагмент коду 1. Реалізація моделі продукту

```
class Product {
  final int id;
  final String name;
  final int categoryId;
  final double calories, protein, fat, carbs, fiber, salt, sugar;
  final String restrictions;
  final String source;
  Product({
    required this.id,
    required this.name,
    required this.categoryId,
    required this.calories,
    required this.protein,
    required this.fat,
    required this.carbs,
    required this.fiber,
    required this.salt,
    required this.sugar,
    required this.restrictions,
    required this.source,
  });
  factory Product.fromMap(Map<String, dynamic> map) => Product(
    id: map['id'],
    name: map['name'],
    categoryId: map['category_id'],
    calories: (map['calories'] as num).toDouble(),
    protein: (map['protein'] as num?)?.toDouble() ?? 0,
    fat: (map['fat'] as num?)?.toDouble() ?? 0,
    carbs: (map['carbs'] as num?)?.toDouble() ?? 0,
    fiber: (map['fiber'] as num?)?.toDouble() ?? 0,
    salt: (map['salt'] as num?)?.toDouble() ?? 0,
    sugar: (map['sugar'] as num?)?.toDouble() ?? 0,
    restrictions: map['restrictions'] ?? '',
    source: map['source'] ?? '',
  );
}
```

## Фрагмент коду 2. Обрахунок добової норми калорій

```
double calcBMR(UserSettings user) {
  final weight = user.weight;
  final height = user.height;
  final age = user.age;
  double bmr = user.sex == «Чоловіча»
      ? 10 * weight + 6.25 * height - 5 * age + 5
      : 10 * weight + 6.25 * height - 5 * age - 161;
  double factor = 1.2;
  if (user.activity == «Низька») factor = 1.2;
  if (user.activity == «Середня») factor = 1.4;
  if (user.activity == «Висока») factor = 1.7;
  bmr *= factor;
  if (user.plan == «Схуднення») bmr *= 0.85;
  if (user.plan == «Екстримальне схуднення») bmr *= 0.75;
  if (user.plan == «Набір маси») bmr *= 1.15;
  return bmr;
}
```

## Фрагмент коду 3. Додавання продукту у раціон

```
void addToRation(Product p, double grams) {
  ration.add(RationItem(p, grams));
  notifyListeners();
}
```

**Фрагмент коду 4. Перевірка на обмеження**

```
bool isRestricted(Product p, UserSettings user) {
  if (p.restrictions.trim().isEmpty) return false;
  for (final r in user.allRestrictionKeys) {
    if (r.isNotEmpty &&
p.restrictions.toLowerCase().contains(r.toLowerCase())) {
      return true;
    }
  }
  return false;
}
```

**Фрагмент коду 4. Весь код файлу main.dart**

```
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:path/path.dart' as p;
import 'package:sqflite/sqflite.dart';

//-----App Entry Point-----
void main() {
  runApp(CalorieAssistantApp());
}

//-----DBHelper: Database logic for productsDB.db-----
class DBHelper {
  static Database? _db;
  static Future<Database> get database async {
    if (_db != null) return _db!;
    var databasesPath = await getDatabasesPath();
    String path = p.join(databasesPath, «productsDB.db»);
    if (!(await File(path).exists())) {
      ByteData data = await
rootBundle.load(«assets/database/productsDB.db»);
      List<int> bytes = data.buffer.asUint8List(data.offsetInBytes,
data.lengthInBytes);
      await File(path).writeAsBytes(bytes, flush: true);
    }
    _db = await openDatabase(path, readOnly: false);
    return _db!;
  }

  static Future<List<Product>> getProducts() async {
    final db = await database;
    final prodList = await db.query(«products»);
    return prodList.map((e) => Product.fromMap(e)).toList();
  }

  static Future<List<Product>> getProductsByCategory(int categoryId)
async {
    final db = await database;
    final prodList = await db.query(«products», where: «category_id
= ?», whereArgs: [categoryId]);
```

```
    return prodList.map((e) => Product.fromMap(e)).toList();
  }

  static Future<Map<int, double>> getAvgCaloriesForAllCategories()
async {
    final db = await database;
    final result = await db.rawQuery(«SELECT category_id,
AVG(calories) as avg_cal FROM products GROUP BY category_id»);
    return { for (var row in result) row['category_id'] as int :
(row['avg_cal'] as num).toDouble() };
  }
}

//-----Product Model-----
class Product {
  final int id;
  final String name;
  final int categoryId;
  final double calories, protein, fat, carbs, fiber, salt, sugar;
  final String restrictions;
  final String source;
  Product({
    required this.id,
    required this.name,
    required this.categoryId,
    required this.calories,
    required this.protein,
    required this.fat,
    required this.carbs,
    required this.fiber,
    required this.salt,
    required this.sugar,
    required this.restrictions,
    required this.source,
  });
  factory Product.fromMap(Map<String, dynamic> map) => Product(
    id: map['id'],
    name: map['name'],
    categoryId: map['category_id'],
    calories: (map['calories'] as num).toDouble(),
    protein: (map['protein'] as num?)?.toDouble() ?? 0,
    fat: (map['fat'] as num?)?.toDouble() ?? 0,
    carbs: (map['carbs'] as num?)?.toDouble() ?? 0,
    fiber: (map['fiber'] as num?)?.toDouble() ?? 0,
    salt: (map['salt'] as num?)?.toDouble() ?? 0,
    sugar: (map['sugar'] as num?)?.toDouble() ?? 0,
    restrictions: map['restrictions'] ?? '',
    source: map['source'] ?? '',
  );
}
```

```
class RationItem {
  final Product product;
  final double grams;
  RationItem(this.product, this.grams);
}

class PlanItem {
  final Product product;
  final double grams;
  PlanItem(this.product, this.grams);
}

//-----UserSettings Model-----
class UserSettings {
  String email;
  String plan;
  List<String> religiousRestrictions;
  List<String> allergyRestrictions;
  double weight;
  double height;
  int age;
  String sex;
  String activity;

  UserSettings({
    required this.email,
    required this.plan,
    required this.religiousRestrictions,
    required this.allergyRestrictions,
    required this.weight,
    required this.height,
    required this.age,
    required this.sex,
    required this.activity,
  });

  List<String> get allRestrictions => [...religiousRestrictions,
...allergyRestrictions];

  List<String> get allRestrictionKeys => [
    ...religiousRestrictions.map((r) => _religiousMap[r] ?? ««),
    ...allergyRestrictions.map((a) => _allergyMap[a] ?? ««),
  ];
  static const Map<String, String> _religiousMap = {
    «Піст»: «religion»,
    «Халяль»: «pork»,
    «Веганство»: «vegan»,
    «Вегетаріанство»: «vegetarian»
  };
```

```
  static const Map<String, String> _allergyMap = {
    «Лактоза»: «lactose»,
    «Глютен»: «gluten»,
    «Горіхи»: «nut»,
    «Інше»: «other»
  };
}

//-----Global State (for session)-----
class AppData extends ChangeNotifier {
  UserSettings? user;
  List<RationItem> ration = [];
  List<PlanItem> plan = [];
  void setUser(UserSettings u) {
    user = u;
    notifyListeners();
  }
  void addToRation(Product p, double grams) {
    ration.add(RationItem(p, grams));
    notifyListeners();
  }
  void addToPlan(Product p, double grams) {
    plan.add(PlanItem(p, grams));
    notifyListeners();
  }
  void removeRationAt(int idx) {
    ration.removeAt(idx);
    notifyListeners();
  }
  void removePlanAt(int idx) {
    plan.removeAt(idx);
    notifyListeners();
  }
  void clearRation() {
    ration.clear();
    notifyListeners();
  }
  void clearPlan() {
    plan.clear();
    notifyListeners();
  }
}

//-----Check if Product is restricted for User-----
bool isRestricted(Product p, UserSettings user) {
  if (p.restrictions.trim().isEmpty) return false;
  for (final r in user.allRestrictionKeys) {
    if (r.isNotEmpty &&
p.restrictions.toLowerCase().contains(r.toLowerCase())) {
      return true;
```

```
    }
  }
  return false;
}

//-----Color for product by category's avg calories (from db)-----
Color getProductColor(Product p, double avgCalories, UserSettings
user) {
  if (isRestricted(p, user)) return Colors.red.shade100;
  if (user.plan == «Схудненння» || user.plan == «Екстримальне
схудненння») {
    return p.calories <= avgCalories ? Colors.green.shade100 :
Colors.yellow.shade100;
  }
  if (user.plan == «Набір маси») {
    return p.calories >= avgCalories ? Colors.green.shade100 :
Colors.yellow.shade100;
  }
  return Colors.white;
}

//-----BMR Calculation-----
double calcBMR(UserSettings user) {
  final weight = user.weight;
  final height = user.height;
  final age = user.age;
  double bmr = user.sex == «Чоловіча»
      ? 10 * weight + 6.25 * height - 5 * age + 5
      : 10 * weight + 6.25 * height - 5 * age - 161;
  double factor = 1.2;
  if (user.activity == «Низька») factor = 1.2;
  if (user.activity == «Середня») factor = 1.4;
  if (user.activity == «Висока») factor = 1.7;
  bmr *= factor;
  if (user.plan == «Схудненння») bmr *= 0.85;
  if (user.plan == «Екстримальне схудненння») bmr *= 0.75;
  if (user.plan == «Набір маси») bmr *= 1.15;
  // Підтримання ваги - без змін
  return bmr;
}

//-----dailyCaloriesNorm function for tests-----
double dailyCaloriesNorm(UserSettings user) {
  return calcBMR(user);
}

//-----CalorieAssistantApp: State provider-----
class CalorieAssistantApp extends StatelessWidget {
  final appData = AppData();
  @override
```

```
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Калорійний Асистент',
      theme: ThemeData(
        primarySwatch: Colors.green,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: ChangeNotifierProvider<AppData>(
        data: appData,
        child: AuthGate(),
      ),
    );
  }
}

//-----AUTH GATE (login/register)-----
class AuthGate extends StatefulWidget {
  @override
  State<AuthGate> createState() => _AuthGateState();
}

class _AuthGateState extends State<AuthGate> {
  bool isRegister = false;
  final emailController = TextEditingController();
  final passwordController = TextEditingController();
  final planController = TextEditingController();
  final weightController = TextEditingController();
  final heightController = TextEditingController();
  final ageController = TextEditingController();

  List<String> religious = [];
  List<String> allergy = [];

  String plan = «Схудnення»;
  String sex = «Чоловіча»;
  String activity = «Середня»;

  final plans = [
    «Схудnення»,
    «Підтримання ваги»,
    «Набір маси»,
    «Екстримальне схудnення»
  ];

  final sexList = [«Чоловіча», «Жіноча»];
  final activityList = [«Низька», «Середня», «Висока»];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
      appBar: AppBar(title: Text(isRegister ? «Реєстрація» :
«Вхід»)),
      body: Padding(
        padding: EdgeInsets.all(16),
        child: ListView(
          children: [
            TextField(
              controller: emailController,
              keyboardType: TextInputType.emailAddress,
              decoration: InputDecoration(labelText: «Email»),
            ),
            if (isRegister)
              Column(
                children: [
                  TextField(
                    controller: passwordController,
                    obscureText: true,
                    decoration: InputDecoration(labelText:
«Пароль»),
                  ),
                  DropdownButtonFormField(
                    value: plan,
                    items: plans
                        .map((e) =>
                        DropdownMenuItem(value: e, child: Text(e)))
                        .toList(),
                    onChanged: (val) {
                      setState(() {
                        plan = val as String;
                      });
                    },
                    decoration: InputDecoration(labelText: «Ціль»),
                  ),
                  TextField(
                    controller: weightController,
                    keyboardType: TextInputType.number,
                    decoration: InputDecoration(labelText: «Вага
(кг)»),
                  ),
                  TextField(
                    controller: heightController,
                    keyboardType: TextInputType.number,
                    decoration: InputDecoration(labelText: «Зріст
(см)»),
                  ),
                  TextField(
                    controller: ageController,
                    keyboardType: TextInputType.number,
                    decoration: InputDecoration(labelText: «Вік»),
                  ),
```

```
                    DropdownButtonFormField(
                      value: sex,
                      items: sexList
                          .map((e) =>
                          DropdownMenuItem(value: e, child: Text(e)))
                          .toList(),
                      onChanged: (val) {
                        setState(() {
                          sex = val as String;
                        });
                      },
                      decoration: InputDecoration(labelText: «Стать»),
                    ),
                    DropdownButtonFormField(
                      value: activity,
                      items: activityList
                          .map((e) =>
                          DropdownMenuItem(value: e, child: Text(e)))
                          .toList(),
                      onChanged: (val) {
                        setState(() {
                          activity = val as String;
                        });
                      },
                      decoration: InputDecoration(labelText:
«Активність»),
                    ),
                    SizedBox(height: 8),
                    Text(«Релігійні обмеження»),
                    MultiSelectChip(
                      items: [
                        «Піст»,
                        «Халяль»,
                        «Веганство»,
                        «Вегетаріанство»,
                      ],
                      selected: religious,
                      onSelectionChanged: (val) {
                        setState(() => religious = val);
                      },
                    ),
                    SizedBox(height: 8),
                    Text(«Алергії»),
                    MultiSelectChip(
                      items: [
                        «Лактоза»,
                        «Глютен»,
                        «Горіхи»,
                        «Інше»
                      ],
```

```
                    selected: allergy,
                    onSelectionChanged: (val) {
                      setState(() => allergy = val);
                    },
                  ),
                ],
              ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () {
                if (isRegister) {
                  if (emailController.text.isNotEmpty &&
                      passwordController.text.isNotEmpty) {
                    final appData =

context.findAncestorWidgetOfExactType<CalorieAssistantApp>()?.appDat
a
                            ?? (context as Element)

.findAncestorWidgetOfExactType<CalorieAssistantApp>()
                            ?.appData;
                    appData?.setUser(UserSettings(
                      email: emailController.text,
                      plan: plan,
                      religiousRestrictions: religious,
                      allergyRestrictions: allergy,
                      weight: double.tryParse(weightController.text)
?? 70,
                      height: double.tryParse(heightController.text)
?? 170,
                      age: int.tryParse(ageController.text) ?? 25,
                      sex: sex,
                      activity: activity,
                    ));
                    Navigator.pushReplacement(
                        context, MaterialPageRoute(builder: (_) =>
MainPage()));
                  }
                } else {
                  final appData =

context.findAncestorWidgetOfExactType<CalorieAssistantApp>()?.appDat
a
                          ?? (context as Element)

.findAncestorWidgetOfExactType<CalorieAssistantApp>()
                          ?.appData;
                  appData?.setUser(UserSettings(
                    email: emailController.text,
                    plan: «Схуднення»,
```

```
                    religiousRestrictions: [],
                    allergyRestrictions: [],
                    weight: 70,
                    height: 170,
                    age: 25,
                    sex: «Чоловіча»,
                    activity: «Середня»,
                  ));
                  Navigator.pushReplacement(
                      context, MaterialPageRoute(builder: (_) =>
MainPage()));
                }
              },
              child: Text(isRegister ? «Зареєструватися» :
«Увійти»),
            ),
            SizedBox(height: 10),
            TextButton(
              onPressed: () => setState(() => isRegister =
!isRegister),
              child: Text(isRegister
                  ? «Вже є акаунт? Увійти»
                  : «Немає акаунту? Зареєструватися»),
            )
          ],
        ),
      ),
    );
  }
}

//-----MULTISELECT CHIP-----
class MultiSelectChip extends StatefulWidget {
  final List<String> items;
  final List<String> selected;
  final Function(List<String>) onSelectionChanged;

  MultiSelectChip(
      {required this.items, required this.selected, required
this.onSelectionChanged});

  @override
  _MultiSelectChipState createState() => _MultiSelectChipState();
}

class _MultiSelectChipState extends State<MultiSelectChip> {
  late List<String> selectedChoices;

  @override
  void initState() {
```

```
    super.initState();
    selectedChoices = List.from(widget.selected);
  }

  @override
  Widget build(BuildContext context) {
    return Wrap(
      children: widget.items.map((item) {
        final sel = selectedChoices.contains(item);
        return Padding(
          padding: const EdgeInsets.symmetric(horizontal: 4.0),
          child: ChoiceChip(
            label: Text(item),
            selected: sel,
            onSelected: (selected) {
              setState(() {
                sel
                    ? selectedChoices.remove(item)
                    : selectedChoices.add(item);
                widget.onSelectionChanged(selectedChoices);
              });
            },
          ),
        );
      }).toList(),
    );
  }
}

//-----MAIN PAGE with tabs-----
class MainPage extends StatefulWidget {
  @override
  State<MainPage> createState() => _MainPageState();
}

class _MainPageState extends State<MainPage> {
  int _selectedIndex = 0;

  @override
  Widget build(BuildContext context) {
    final appData =
(context.findAncestorWidgetOfExactType<CalorieAssistantApp>()?.appDa
ta
        ?? (context as
Element).findAncestorWidgetOfExactType<CalorieAssistantApp>()?.appDa
ta)!;

    return Scaffold(
      body: IndexedStack(
        index: _selectedIndex,
```

```
        children: [
          RationPage(appData: appData),
          PlanPage(appData: appData),
          ProductsPage(appData: appData),
          ProfilePage(appData: appData),
          AnalysisPage(appData: appData),
        ],
      ),
      bottomNavigationBar: BottomNavigationBar(
        type: BottomNavigationBarType.fixed,
        items: [
          BottomNavigationBarItem(icon: Icon(Icons.restaurant),
label: «Раціон»),
          BottomNavigationBarItem(icon: Icon(Icons.calendar_today),
label: «План»),
          BottomNavigationBarItem(icon:
Icon(Icons.local_grocery_store), label: «Продукти»),
          BottomNavigationBarItem(icon: Icon(Icons.person), label:
«Параметри»),
          BottomNavigationBarItem(icon: Icon(Icons.analytics),
label: «Аналіз»),
        ],
        currentIndex: _selectedIndex,
        onTap: (i) => setState(() => _selectedIndex = i),
      ),
    );
  }
}

//-----Products Tab-----
class ProductsPage extends StatefulWidget {
  final AppData appData;
  ProductsPage({required this.appData});
  @override
  State<ProductsPage> createState() => _ProductsPageState();
}

class _ProductsPageState extends State<ProductsPage> {
  List<Product> products = [];
  Map<int, double> avgCaloriesByCategory = {};

  @override
  void initState() {
    super.initState();
    _loadProductsAndAvgs();
  }

  Future<void> _loadProductsAndAvgs() async {
    final allProds = await DBHelper.getProducts();
    final avgs = await DBHelper.getAvgCaloriesForAllCategories();
```

```
    setState(() {
      products = allProds;
      avgCaloriesByCategory = avgs;
    });
  }

  void addWithDialog(Product prod, bool toRation) async {
    final gramsController = TextEditingController();
    await showDialog(
        context: context,
        builder: (BuildContext context) {
          return AlertDialog(
            title: Text('Скільки грамів?'),
            content: TextField(
              controller: gramsController,
              keyboardType: TextInputType.number,
              decoration: InputDecoration(hintText: «Введіть
грамовку»),
            ),
            actions: [
              TextButton(
                onPressed: () {
                  Navigator.pop(context);
                },
                child: Text(«Скасувати»),
              ),
              TextButton(
                onPressed: () {
                  final grams =
double.tryParse(gramsController.text) ?? 0;
                  if (grams > 0) {
                    if (toRation) {
                      widget.appData.addToRation(prod, grams);
                    } else {
                      widget.appData.addToPlan(prod, grams);
                    }
                    Navigator.pop(context);
                    ScaffoldMessenger.of(context).showSnackBar(
                        SnackBar(content: Text('${prod.name}
додано!')));
                  }
                },
                child: Text(«Додати»),
              ),
            ],
          );
        });
  }

  @override
```

```
  Widget build(BuildContext context) {
    final user = widget.appData.user!;
    return Scaffold(
      appBar: AppBar(title: Text(«Продукти»)),
      body: products.isEmpty
          ? Center(child: Text(«Немає продуктів»))
          : ListView.builder(
        itemCount: products.length,
        itemBuilder: (ctx, idx) {
          final prod = products[idx];
          final avgCal = avgCaloriesByCategory[prod.categoryId] ??
0;
          final color = getProductColor(prod, avgCal, user);
          return Card(
            color: color,
            child: ListTile(
              title: Text(prod.name),
              subtitle: Text(
                  «Ккал: ${prod.calories} | Б: ${prod.protein} | Ж:
${prod.fat} | В: ${prod.carbs}»),
              trailing: Row(
                mainAxisSize: MainAxisSize.min,
                children: [
                  IconButton(
                    icon: Icon(Icons.restaurant),
                    tooltip: «Додати у раціон»,
                    onPressed: () => addWithDialog(prod, true),
                  ),
                  IconButton(
                    icon: Icon(Icons.calendar_today),
                    tooltip: «Додати у план»,
                    onPressed: () => addWithDialog(prod, false),
                  ),
                ],
              ),
            ),
          );
        },
      ),
    );
  }
}

//-----Ration Tab-----
class RationPage extends StatelessWidget {
  final AppData appData;
  RationPage({required this.appData});

  @override
  Widget build(BuildContext context) {
```

```
    final user = appData.user!;
    double totalKcal = appData.ration.fold(0.0, (sum, item) =>
    sum + (item.product.calories * item.grams / 100));
    double dailyNorm = calcBMR(user);
    double percent = dailyNorm == 0 ? 0 : (totalKcal / dailyNorm) *
100;
    Color infoColor = Colors.green.shade100;

    if (user.plan == «Набір маси») {
      if (percent < 70) infoColor = Colors.red.shade100;
      else if (percent < 90) infoColor = Colors.yellow.shade100;
      else if (percent > 140) infoColor = Colors.red.shade100;
      else infoColor = Colors.green.shade100;
    } else {
      if (percent > 120) infoColor = Colors.red.shade100;
      else if (percent > 105) infoColor = Colors.yellow.shade100;
      else infoColor = Colors.green.shade100;
    }

    return Scaffold(
      appBar: AppBar(
        title: Text('Раціон'),
      ),
      body: Column(
        children: [
          Expanded(
            child: appData.ration.isEmpty
                ? Center(child: Text(«Раціон порожній»))
                : ListView.builder(
              itemCount: appData.ration.length,
              itemBuilder: (ctx, idx) {
                final item = appData.ration[idx];
                return FutureBuilder<double>(
                    future:
DBHelper.getProductsByCategory(item.product.categoryId)
                        .then((list) => list.isEmpty
                        ? 0
                        : list.map((e) => e.calories).reduce((a, b)
=> a + b) / list.length),
                    builder: (context, snap) {
                      final avg = snap.data ?? 0;
                      final color = getProductColor(item.product,
avg, user);
                      return Dismissible(
                        key: ValueKey(item.product.id.toString() +
item.grams.toString()),
                        background: Container(color:
Colors.red.shade100),
                        onDismissed: (_) =>
appData.removeRationAt(idx),
```

```
                        child: Card(
                          color: color,
                          child: ListTile(
                            title: Text(item.product.name),
                            subtitle: Text(
                                «${item.grams} г | Ккал:
${(item.product.calories * item.grams / 100).toStringAsFixed(1)}»),
                          ),
                        ),
                      );
                    });
              },
            ),
          ),
          Container(
            width: double.infinity,
            padding: EdgeInsets.all(16),
            color: infoColor,
            child: Text(
              «Калорій за день: ${totalKcal.toStringAsFixed(0)} /
${dailyNorm.toStringAsFixed(0)} «
                  «(${percent.toStringAsFixed(0)}%)»,
              textAlign: TextAlign.center,
              style: TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
            ),
          ),
        ],
      ),
    );
  }
}

//-----Plan Tab-----
class PlanPage extends StatelessWidget {
  final AppData appData;
  PlanPage({required this.appData});

  @override
  Widget build(BuildContext context) {
    final user = appData.user!;
    return Scaffold(
      appBar: AppBar(
        title: Text('План'),
      ),
      body: appData.plan.isEmpty
          ? Center(child: Text(«План порожній»))
          : ListView.builder(
        itemCount: appData.plan.length,
        itemBuilder: (ctx, idx) {
```

```
          final item = appData.plan[idx];
          return FutureBuilder<double>(
              future:
DBHelper.getProductsByCategory(item.product.categoryId)
                  .then((list) => list.isEmpty
                  ? 0
                  : list.map((e) => e.calories).reduce((a, b) => a +
b) / list.length),
              builder: (context, snap) {
                final avg = snap.data ?? 0;
                final color = getProductColor(item.product, avg,
user);
                return Dismissible(
                  key: ValueKey(item.product.id.toString() +
item.grams.toString()),
                  background: Container(color: Colors.red.shade100),
                  onDismissed: (_) => appData.removePlanAt(idx),
                  child: Card(
                    color: color,
                    child: ListTile(
                      title: Text(item.product.name),
                      subtitle: Text(
                          «${item.grams} г | Ккал:
${(item.product.calories * item.grams / 100).toStringAsFixed(1)}»),
                      trailing: IconButton(
                        icon: Icon(Icons.check_circle_outline),
                        color: Colors.green,
                        tooltip: «Додати у раціон»,
                        onPressed: () {
                          appData.addToRation(item.product,
item.grams);
                          appData.removePlanAt(idx);

ScaffoldMessenger.of(context).showSnackBar(
                            SnackBar(content:
Text('${item.product.name} додано в раціон!')),
                          );
                        },
                      ),
                    ),
                  ),
                );
              });
        },
      ),
    );
  }
}

//-----Profile Tab-----
```

```
class ProfilePage extends StatefulWidget {
  final AppData appData;
  ProfilePage({required this.appData});
  @override
  State<ProfilePage> createState() => _ProfilePageState();
}

class _ProfilePageState extends State<ProfilePage> {
  late TextEditingController weightController;
  late TextEditingController heightController;
  late TextEditingController ageController;
  String plan = "";
  List<String> religious = [];
  List<String> allergy = [];
  String sex = "Чоловіча";
  String activity = "Середня";

  final sexList = ["Чоловіча", "Жіноча"];
  final activityList = ["Низька", "Середня", "Висока"];

  @override
  void initState() {
    super.initState();
    weightController =
        TextEditingController(text:
widget.appData.user?.weight.toString() ?? "");
    heightController =
        TextEditingController(text:
widget.appData.user?.height.toString() ?? "");
    ageController =
        TextEditingController(text:
widget.appData.user?.age.toString() ?? "");
    plan = widget.appData.user?.plan ?? "Схудненння";
    religious = List.from(widget.appData.user?.religiousRestrictions
?? []);
    allergy = List.from(widget.appData.user?.allergyRestrictions ??
[]);
    sex = widget.appData.user?.sex ?? "Чоловіча";
    activity = widget.appData.user?.activity ?? "Середня";
  }

  @override
  Widget build(BuildContext context) {
    final plans = [
      "Схудненння",
      "Підтримання ваги",
      "Набір маси",
      "Екстримальне схудненння"
    ];
    return Scaffold(
```

```
      appBar: AppBar(title: Text(«Параметри»)),
      body: Padding(
        padding: EdgeInsets.all(16.0),
        child: ListView(
          children: [
            TextField(
              controller: weightController,
              keyboardType: TextInputType.number,
              decoration: InputDecoration(labelText: «Вага (кг)»),
            ),
            TextField(
              controller: heightController,
              keyboardType: TextInputType.number,
              decoration: InputDecoration(labelText: «Зріст (см)»),
            ),
            TextField(
              controller: ageController,
              keyboardType: TextInputType.number,
              decoration: InputDecoration(labelText: «Вік»),
            ),
            DropdownButtonFormField(
              value: plan,
              items: plans
                  .map((e) =>
                  DropdownMenuItem(value: e, child: Text(e)))
                  .toList(),
              onChanged: (val) {
                setState(() {
                  plan = val as String;
                });
              },
              decoration: InputDecoration(labelText: «Ціль»),
            ),
            DropdownButtonFormField(
              value: sex,
              items: sexList
                  .map((e) => DropdownMenuItem(value: e, child:
Text(e)))
                  .toList(),
              onChanged: (val) {
                setState(() {
                  sex = val as String;
                });
              },
              decoration: InputDecoration(labelText: «Стать»),
            ),
            DropdownButtonFormField(
              value: activity,
              items: activityList
                  .map((e) => DropdownMenuItem(value: e, child:
```

```
Text(e)))
                .toList(),
            onChanged: (val) {
              setState(() {
                activity = val as String;
              });
            },
            decoration: InputDecoration(labelText: «Активність»),
          ),
          SizedBox(height: 8),
          Text(«Релігійні обмеження»),
          MultiSelectChip(
            items: [
              «Піст»,
              «Халяль»,
              «Веганство»,
              «Вегетаріанство»,
            ],
            selected: religious,
            onSelectionChanged: (val) {
              setState(() => religious = val);
            },
          ),
          SizedBox(height: 8),
          Text(«Алергії»),
          MultiSelectChip(
            items: [
              «Лактоза»,
              «Глютен»,
              «Горіхи»,
              «Інше»
            ],
            selected: allergy,
            onSelectionChanged: (val) {
              setState(() => allergy = val);
            },
          ),
          SizedBox(height: 16),
          ElevatedButton(
            onPressed: () {
              widget.appData.user = UserSettings(
                email: widget.appData.user?.email ?? ««,
                plan: plan,
                religiousRestrictions: religious,
                allergyRestrictions: allergy,
                weight: double.tryParse(weightController.text) ??
70,
                height: double.tryParse(heightController.text) ??
170,
                age: int.tryParse(ageController.text) ?? 25,
```

```
                  sex: sex,
                  activity: activity,
                );
                ScaffoldMessenger.of(context).showSnackBar(
                    SnackBar(content: Text(«Параметри
збережено!»)));
              },
              child: Text(«Зберегти»),
            )
          ],
        ),
      ),
    );
  }
}

//-----Analysis Tab-----
class AnalysisPage extends StatelessWidget {
  final AppData appData;
  AnalysisPage({required this.appData});
  @override
  Widget build(BuildContext context) {
    final user = appData.user!;
    final ration = appData.ration;
    final plan = appData.plan;

    List<Widget> analysisList = [];

    for (var item in ration) {
      if (isRestricted(item.product, user)) {
        analysisList.add(ListTile(
          leading: Icon(Icons.warning, color: Colors.red),
          title: Text(
              «У раціоні: ${item.product.name} порушує ваші
обмеження! Це може бути небезпечно для здоров’я.»),
        ));
      }
    }
    for (var item in plan) {
      if (isRestricted(item.product, user)) {
        analysisList.add(ListTile(
          leading: Icon(Icons.warning, color: Colors.red),
          title: Text(
              «У плані: ${item.product.name} порушує ваші обмеження!
Це може бути небезпечно для здоров’я.»),
        ));
      }
    }

    void checkOptimal(List list, String where) async {
```

```
      for (var item in list) {
        final avg = await
DBHelper.getProductsByCategory(item.product.categoryId)
            .then((products) => products.isEmpty
            ? 0
            : products.map((e) => e.calories).reduce((a, b) => a +
b) / products.length);
        if ((user.plan == «Схуднення» || user.plan == «Екстримальне
схуднення») && item.product.calories > avg) {
          analysisList.add(ListTile(
            leading: Icon(Icons.tips_and_updates, color:
Colors.orange),
            title: Text(«У $where: ${item.product.name} – не
оптимальний вибір для схуднення (завелика калорійність). Обирайте
продукти позначені зеленим.»),
          ));
        }
        if (user.plan == «Набір маси» && item.product.calories <
avg) {
          analysisList.add(ListTile(
            leading: Icon(Icons.tips_and_updates, color:
Colors.orange),
            title: Text(«У $where: ${item.product.name} – не
оптимальний вибір для набору маси (замала калорійність). Обирайте
продукти позначені зеленим.»),
          ));
        }
      }
    }

    return FutureBuilder(
        future: () async {
          checkOptimal(ration, «раціоні»);
          checkOptimal(plan, «плані»);
          await Future.delayed(Duration(milliseconds: 100));
          return;
        }(),
        builder: (context, snapshot) {
          if (snapshot.connectionState != ConnectionState.done)
            return Scaffold(
              appBar: AppBar(title: Text(«Аналіз»)),
              body: Center(child: CircularProgressIndicator()),
            );
          if (analysisList.isEmpty) {
            analysisList.add(ListTile(
              title: Text(«Зараз у раціоні та плані відсутні
проблеми або не оптимальні продукти!»),
            ));
          }
          return Scaffold(
```

```
            appBar: AppBar(title: Text(«Аналіз»)),
            body: ListView(
              padding: EdgeInsets.all(16),
              children: analysisList,
            ),
          );
        });
  }
}

//-----Provider (simple ChangeNotifier replacement)-----
class ChangeNotifierProvider<T extends ChangeNotifier> extends
InheritedWidget {
  final T data;
  ChangeNotifierProvider({required Widget child, required
this.data})
      : super(child: child);

  static T? of<T extends ChangeNotifier>(BuildContext context) {
    final provider =

context.dependOnInheritedWidgetOfExactType<ChangeNotifierProvider<T>
>();
    return provider?.data;
  }

  @override
  bool updateShouldNotify(ChangeNotifierProvider<T> oldWidget) =>
true;
}
```

**Додаток 3**

**Копія презентації**

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

# МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ІНТЕРАКТИВНОГО ПІДБОРУ ХАРЧУВАННЯ

## «AlterMeal»

Виконала: студентка групи КП-13 Єрошин Богдан Дмитрович

Керівник: доцент кафедри ПЗКС, к.т.н., доцент Люшенко Леся Анатоліївна

Київ – 2025


# ПОСТАНОВКА ЗАДАЧІ

**Мета проєкту:** Метою дипломного проєкту є створення мобільного застосунку «AlterMeal» для інтерактивного підбору харчування і формуванню харчового раціону, який враховує індивідуальні потреби користувача, медичні, релігійні, інші обмеження, а також забезпечить планування та відстежування раціонну харчування.

**Завдання:**

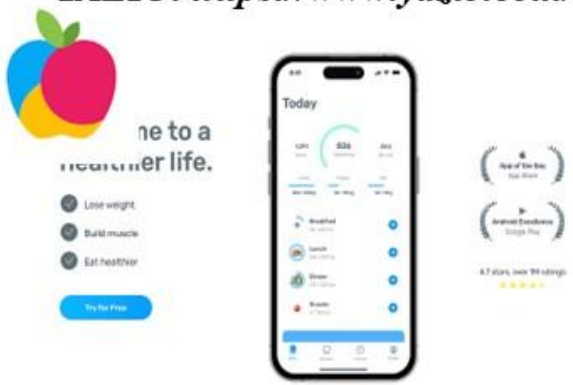
1. Проаналізувати існуючі програмні рішення
2. Визначити вимоги до розробки
3. Обрати засоби реалізації
4. Розробити вебзастосунок відповідно до вимог
5. Протестувати розроблений вебзастосунок
6. Визначити шляхи подальшого розвитку

# АКТУАЛЬНІСТЬ

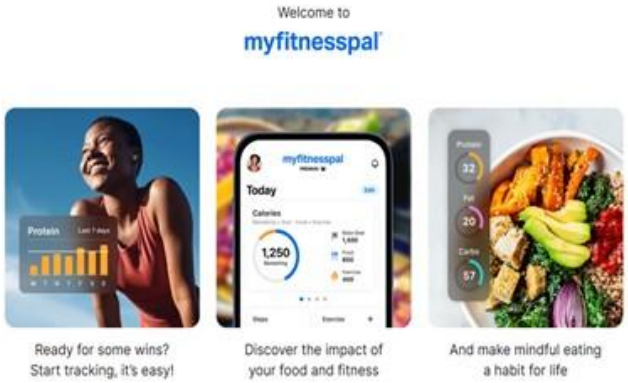
Застосунок «AlterMeal» забезпечує індивідуальний підхід до кожного користувача згідно з його цілями, підбір продуктів на основі смаків та звичок користувача, забезпечує м`який перехід до нових харчових звичок.

# ІСНУЮЧІ АНАЛОГИ

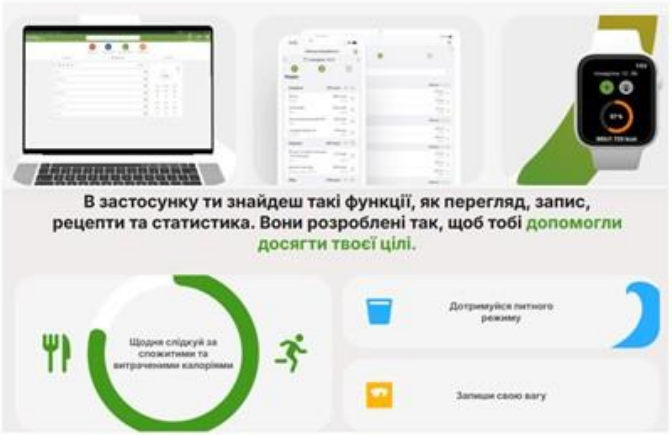
***YAZIO. https://www.yazio.com/en***




***MyFitnessPal. https://www.myfitnesspal.com***




***Таблиця калорійності. https://www.tablycjakalorijnosti.com.ua***



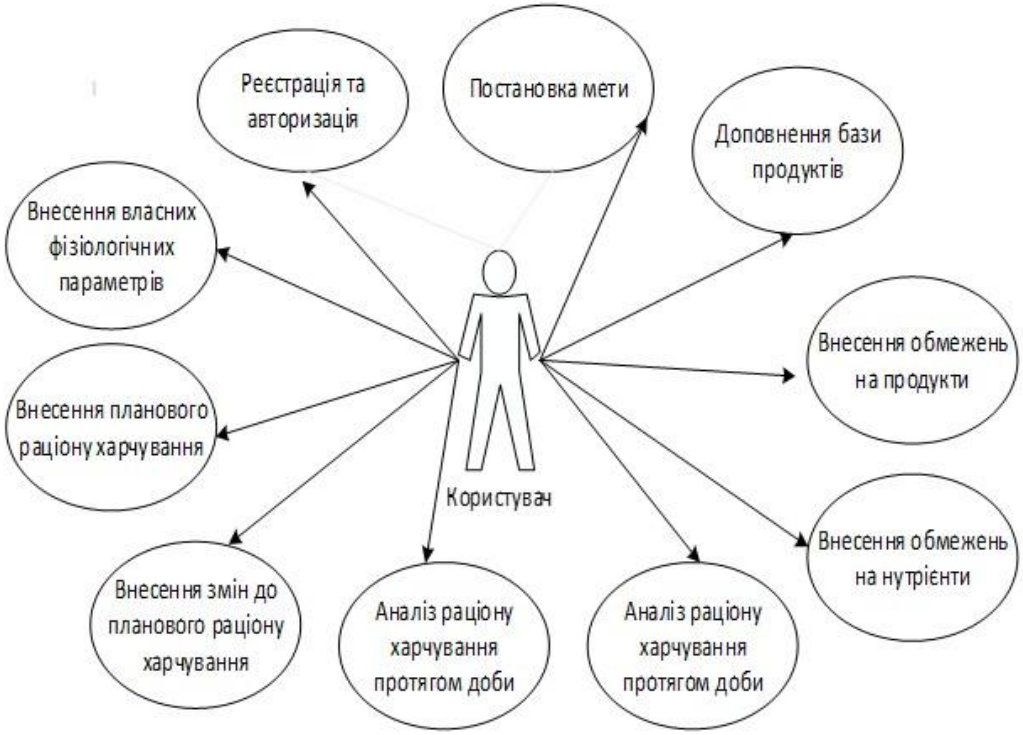
# ПРОБЛЕМА ХАРЧОВИХ ОБМЕЖЕНЬ

## ОПИТУВАННЯ ПОКАЗАЛО:


# ВИМОГИ ДО ЗАСТОСУНКУ

- реєстрація та авторизація за допомогою електронної пошти або номеру телефону користувача;
- формування рекомендацій;
- адаптивний дизайн;
- зручний, естетичний інтерфейс користувача;
- пошук та фільтрація записів за різними критеріями;
- формування налаштування профілю користувача (вік, стать, мета тощо);
- відстеження КЖБВ та наявність нутрієнтів;
- планування харчування;
- формування обмежень .

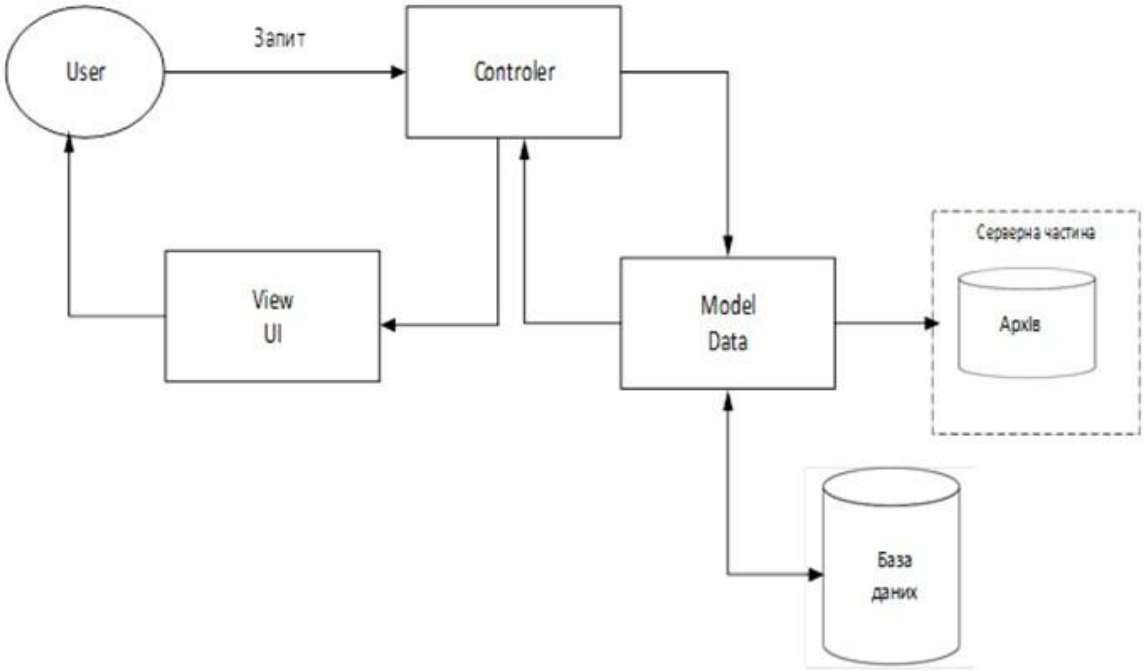
# ДІАГРАМА ПРИЦЕДЕНТІВ



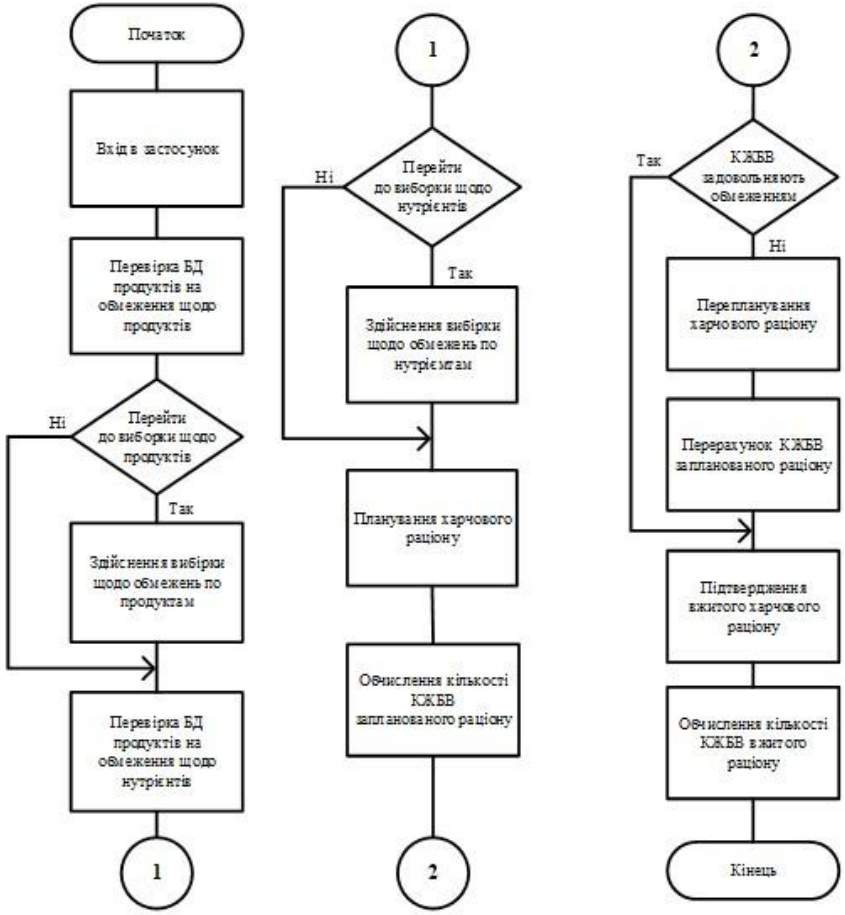
# ЗАСОБИ РОЗРОБЛЕННЯ


Тип ПЗ — **мобільний застосування**
Мова програмування — **Dart**
Фреймворк — **Flutter**
База даних —

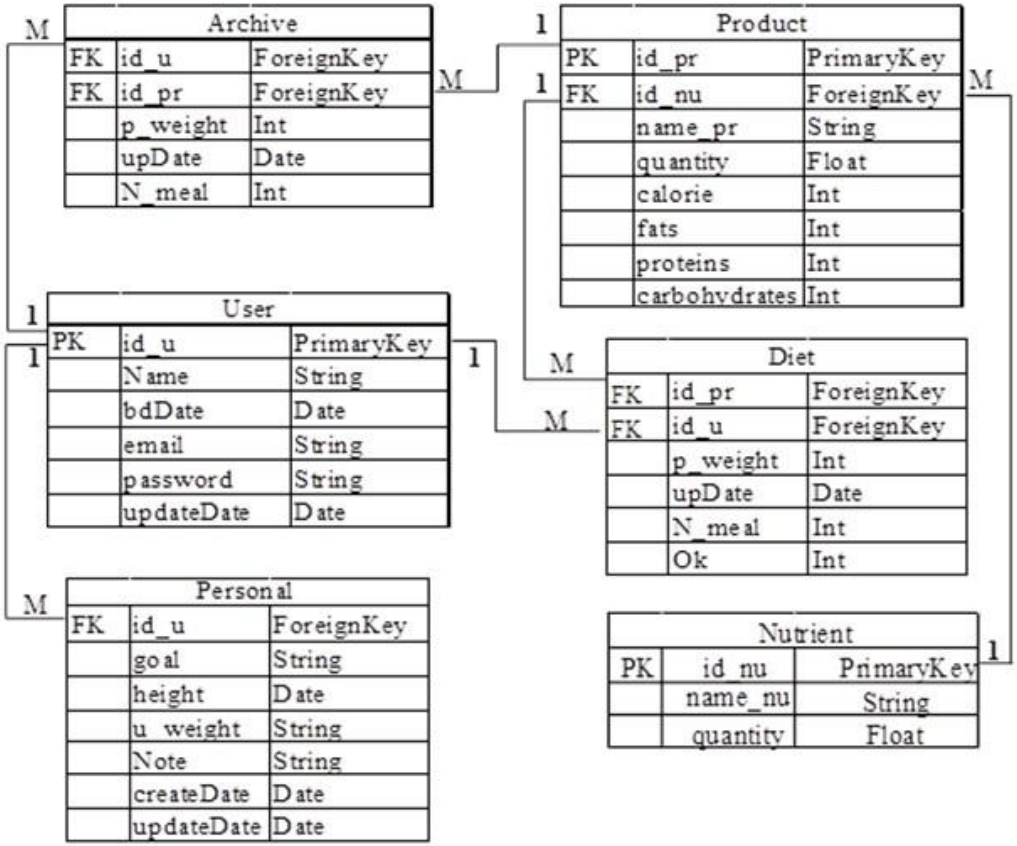

# АРХІТЕКТУРА СИСТЕМИ



# ДІАГРАМА ДІЙ ОСНОВНОГО СЦЕНАРІЮ РОБОТИ ЗАСТОСУНКУ



# СТРУКТУРА БАЗИ ДАНИХ



# ТЕСТУВАННЯ ЗАСТОСУНКУ


Ручне тестування проводилося на різних фізичних пристроях і емуляторах з Android та iOS, а саме:

- Перевірка коректності авторизації та реєстрації
- Тестування роботи з раціоном
- Введення обмежень
- Перевірка підрахунку калорій та нутрієнтів
- Тестування аналізування харчового раціону.
- Збереження архіву на сервері
- Перевірка адаптивності інтерфейсу

# ТЕСТУВАННЯ ЗАСТОСУНКУ


**Unit тестування**. Використання розроблених юніт-тести на Dart з пакетом flutter_test, а спме:

- Перевірка коректності роботи з даними продуктів (створення, пошук, обробка обмежень, підрахунок калорій).
- Тести функцій фільтрації продуктів за алергенами, релігійними та смаковими обмеженнями.
- Тести CRUD-операцій для локальної бази даних SQLite – додавання, редагування, видалення записів про прийоми їжі.
- Тести на валідацію введених даних користувача.

**Верифікація UX та UI:**

- Тестування сценаріїв користувача.
- Тестування usability інтерфейсу.
- Тестування реакції на помилки.

# ШЛЯХИ ПОДАЛЬШОГО РОЗВИТКУ


- Можливості додавання власних продуктів.
- Розширення бази рецептів та автоматичний підбір альтернатив.
- Синхронізація даних із хмарними сервісами.
- Інтеграція з фітнес-трекерами та іншими застосунками здоров'я.
- Додаткові аналітичні інструменти
- Багатомовна підтримка.
- Доступність для людей з особливими потребами.

# Апробація застосунку

- Фіналіст конкурсу «XII Міжнародний Фестиваль інноваційних проєктів "Sikorsky Challenge 2023» Назва проєкту: ALTERMEAL №100

- Фіналіст конкурсу стартапів проєкту “Resilience of Education: Sustainability and Cooperation for Ukrainian Universities”, RESCUU.

# ВИСНОВКИ

Створений мобільний застосунок зберігає значний перелік продуктів харчування з КЖБВ та нутрієнтами, а також має функції планування харчового раціону з урахуванням харчових звичок споживача, визначенням обмежень та збереженням спожитого раціону з його аналізуванням.

Для цього:

1. Проаналізовано існуючі програмні рішення.
2. Спроєктовано архітектуру вебзастосунку та структуру БД
3. Розроблено серверну та клієнтську частини вебзастосунку.
4. Протестовано розробку шляхом ручного і автоматизованого тестування.
5. Визначено шляхи подальшого розвитку.

Розробка виконана у повному обсязі відповідно до технічного завдання, тестування проведено згідно з затвердженою програмою та методикою тестування.


# Дякую за увагу!

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп'ютерних систем**

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2024 р

# "МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ІНТЕРАКТИВНОГО ПІДБОРУ ХАРЧУВАННЯ"

## Програма та методика тестування

ДП.045420-04-51

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Леся ЛЮШЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Богдан ЄРОШИН

2024

**ЗМІСТ**

1. Об’єкт випробувань ..... 3

2. Мета тестування ..... 3

3. Методи тестування ..... 3

4. Засоби та порядок тестування ..... 4

## 1. ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є мобільний застосунок для інтерактивного підбору харчування “AlterMeal”. Цей застосунок реалізований мовою програмування Dart в роєднанні з фреймворком Flutter.

## 2. МЕТА ТЕСТУВАННЯ

Метою тестування є перевірка наступного:

- функціональна працездатність елементів мобільного застосунку: додавання продуктів харчування до раціону, підрахунок показників раціону, введення користувацьких обмежень, перегляд історії споживання, налаштування профілю користувача, збереження та обробка даних у локальній базі даних;
- коректне перенесення архіву раціону на сервер;
- адаптивність інтерфейсу для різних розмірів екранів і платформ (Android/iOS);
- коректна робота екранів і форм (вхід, реєстрація, головна форма, перегляд раціону, планування тощо);
- коректна обробка помилок;
- забезпечення належного рівня безпеки даних;
- відповідність розробки вимогам Технічного завдання.

## 3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується за допомогою автоматизованого юніт-тестування і ручного наскрізного тестування. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Використовуються наступні методи:

- функціональне тестування;

- інтеграційне тестування;
- крос-платформне тестування;
- тестування продуктивності;
- тестування сумісності;
- тестування інтерфейсу.

## 4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність мобільного застосунку перевіряється шляхом:

- динамічного ручного тестування – введенням недопустимих значень в поля введення даних;
- динамічного ручного тестування на відповідність функціональним вимогам;
- статичного тестування коду;
- автоматизованого тестування компонентів коду;
- тестування зручності використання;
- тестування інтерфейсу.

Ручне тестування на різних фізичних пристроях і емуляторах з Android та iOS:

- Перевірка коректності авторизації та реєстрації – введення валідних та невалідних даних, обробка помилок при невірному логіні або паролі, відображення відповідних повідомлень.
- Тестування роботи з харчовим раціоном – додавання, редагування і видалення продуктів із щоденного раціону, перевірка збереження та відображення даних після перезапуску застосунку.
- Введення обмежень – додавання обмежень, перевірка фільтрації продуктів відповідно до встановлених правил.
- Перевірка підрахунку параметрів харчового раціону.

- Коректність надання альтернатив – тестування рекомендацій щодо заміни небажаних продуктів або тих, що підпадають під обмеження користувача.
- Збереження архіву на сервер – ручна перевірка процедури архівації даних та отримання підтвердження від користувача.
- Перевірка адаптивності інтерфейсу – зміна орієнтації екрана, тестування на різних розмірах дисплеїв, перевірка масштабування UI-елементів.

Unit тестування. Використання розроблених юніт-тести на Dart з пакетом flutter_test, а спме:

- Перевірка коректності роботи з даними продуктів (створення, пошук, обробка обмежень, підрахунок калорій).
- Тести функцій фільтрації продуктів за алергенами, релігійними та смаковими обмеженнями.
- Тести CRUD-операцій для локальної бази даних SQLite – додавання, редагування, видалення записів про прийоми їжі.
- Тести на валідацію введених даних користувача.

Верифікація UX та UI:

- Тестування сценаріїв користувача.
- Тестування usability інтерфейсу.
- Тестування реакції на помилки.

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп'ютерних систем**

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2025

# МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ІНТЕРАКТИВНОГО ПІДБОРУ ХАРЧУВАННЯ

**Керівництво користувача**

ДП.045420-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Леся ЛЮШЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Богдан ЄРОШИН

2025

**ЗМІСТ**

1. Опис структури мобільного застосунку “AlterMeal”……3
2. Реєстрація та вхід……..3
3. Введення та оновлення даних профілю……..4
4. Додавання продуктів до раціону…….….4
5. Перегляд і редагування раціону…….……5
6. Аналіз та рекомендації…….…7

## 1. Опис структури мобільного застосунку “altermeal”

Мобільний застосунок для інтерактивного підбору харчування “AlterMeal” створений для того, щоб допомогти користувачам контролювати своє харчування, вести щоденник харчового раціонуі, автоматично рахувати КЖБУ та основні нутрієнти, а також отримувати корисні поради й альтернативи відповідно до індивідуальних обмежень і цілей. Далі подано покрокову інструкцію з використання основних функцій застосунку.

## 2. Реєстрація та вхід

Перший запуск: на стартовому екрані оберіть “Реєстрація”, якщо ви новий користувач, або “Вхід” для вже зареєстрованих (рис. 1).

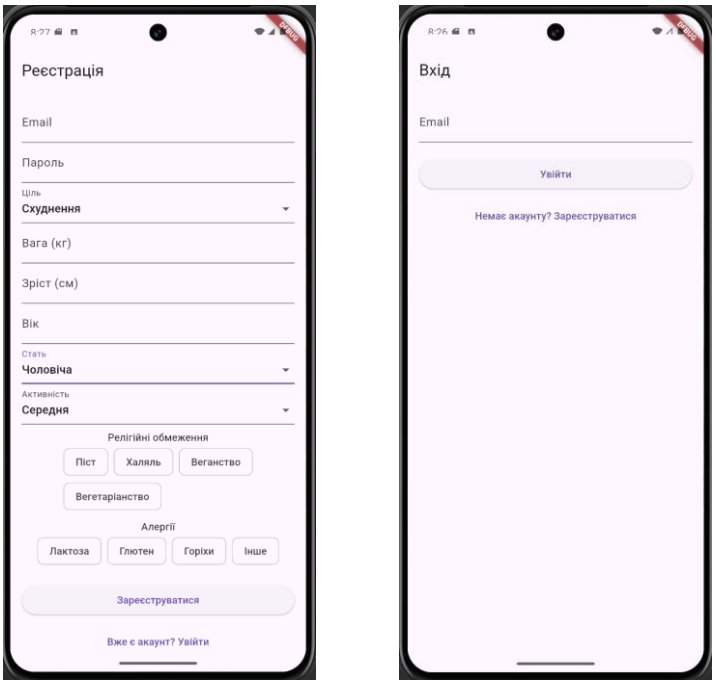



Рис. 1. Форма “Реєстрація” та форма “Вхід”

Реєстрація:

- Введіть ваш e-mail, придумайте пароль.

- Заповніть основні параметри: вік, зріст, вагу, стать, рівень фізичної активності, виберіть ціль (схуднення, підтримка ваги, набір маси, усунення швидких вуглеводів).
- За бажанням додайте алергії або релігійні обмеження, що впливатимуть на рекомендації продуктів.
- Натисніть кнопку “Зареєструватися”.

Вхід:

- Введіть свої дані (e-mail та пароль) для авторизації.
- Натисніть “Увійти”.

### 3. Введення та оновлення даних профілю

У формі “Параметри” вводяться, переглядаються та змінюються особисті дані: вагу, зріст, вік, ціль, рівень активності, стать (рис. 2). За необхідності можна змінити список обмежень, що автоматично впливатиме на підбір дозволених продуктів. Зберігання введеної інформації відбувається натисканням відповідної кнопки.

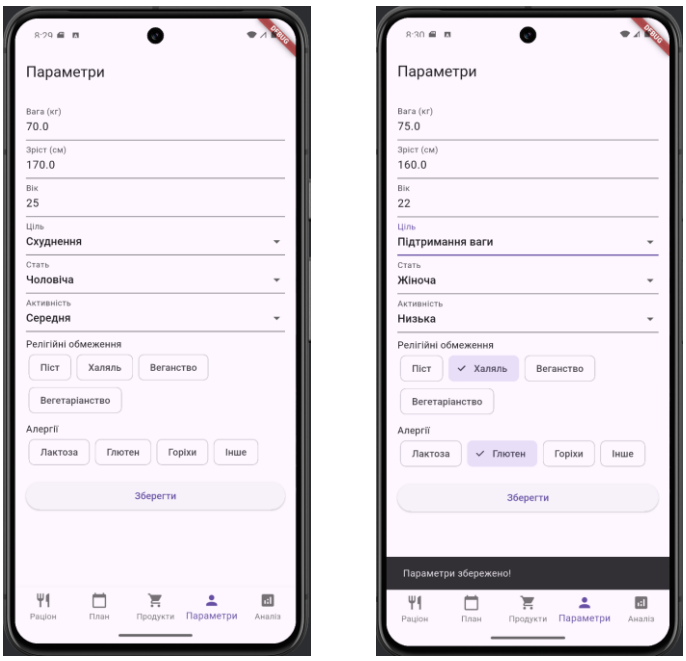



Рис. 3. Форма “Параметри”. Редагування профілю.

## 4. Додавання продуктів до раціону

В формі “Продукти” надається повний перелік продуктів з КЖБУ та їх складом. Колір картки продукту підкаже, чи підходить продукт під вашу мету та обмеження (рис. 4).

Для додавання продукту необхідно натисніти на відповідну на позначку “Додати до раціону”. Вкажіть вагу продукту у грамах та підтвердьте дію. Обраний продукт буде відображений у вкладці “Раціон”.

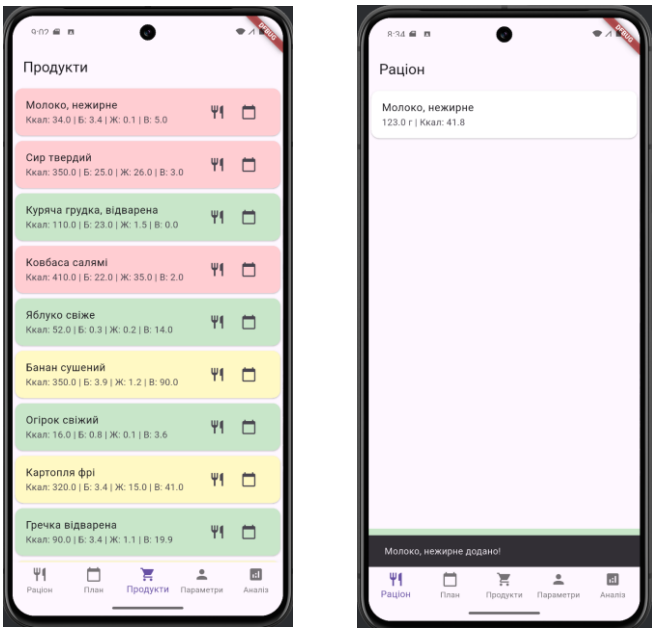



Рис. 4. Форма “Продукти”. Додавання продуктів до раціону харчування

## 5. Перегляд і редагування раціону

В формі “Раціон” відображаються всі продукти, які додаються протягом дня, зі зазначенням ваги КБВЖ та основних нутрієнтів (рис. 5). Для видалення продукту проведіть пальцем по екрану в будь-яку спорону на місці визначеного для виділення продукту. Внизу відображається ваш денний підсумок калорій. При цьому порівнюється фактичні значення

параметрів із рекомендованою нормою відповідно до персональних даних і цілі.

Колір блоку підсумку змінюється залежно від того, наскільки наблизились або перевищили норму споживання КЖБУ (рис. 6).

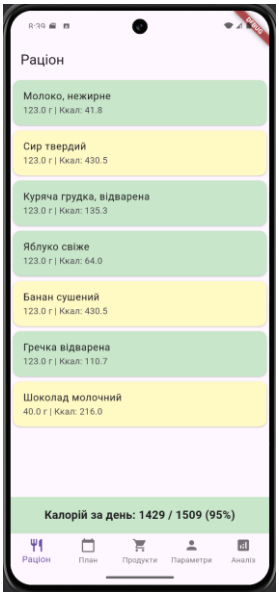



Рис. 7. Форма “Раціон”. Підрахунок калорій

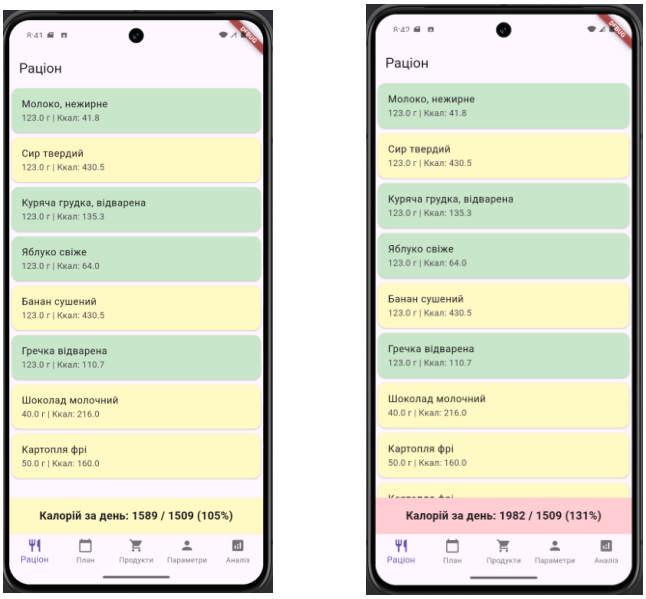



Рис. 8. Форма “Раціон”. Зміна кольору відповідно до перебільшення калорій.

Якщо використовувати функцію планування раціону, то спочатку заповнюється форма “План”. Після вживання їжі з форми “План” користувач переносить продукт натиснувши відповідну позначку в форму “Раціон” (рис. 9).

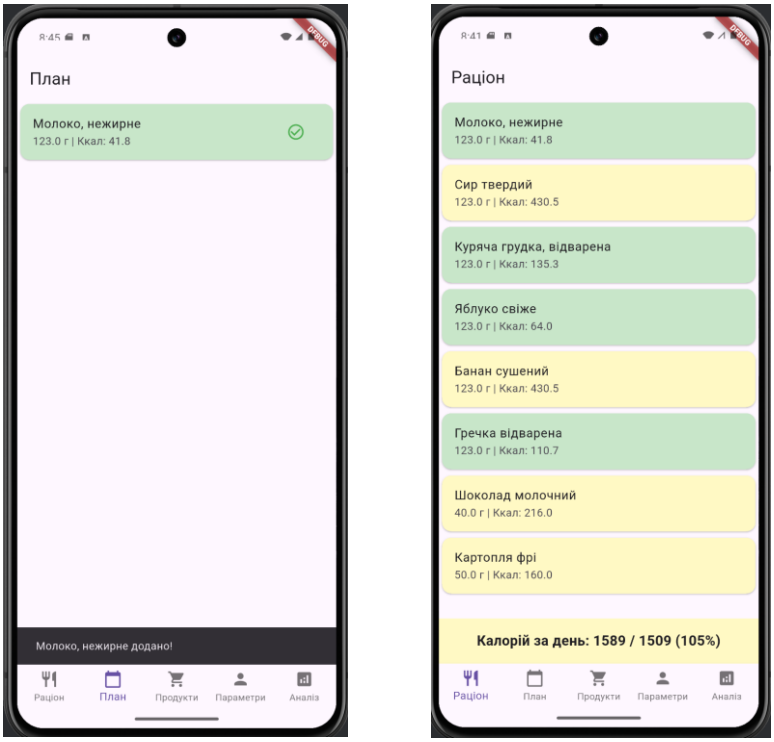



Рис. 9. Форма “План”. Перенесення запланованого продукту в форму “Раціон”

## 6. Аналіз та рекомендації

У вкладці “Аналіз” ( рис.10) надається інформація щодо відповідності складу вашого раціону до заявлених обмежень та цілей, а саме:

- Продукти, які порушують обмеження.
- Продукти, вживання яких не відповідає невідповідає досягненню мети.
- Пропозиція переглянути альтернативи для небажаних продуктів.

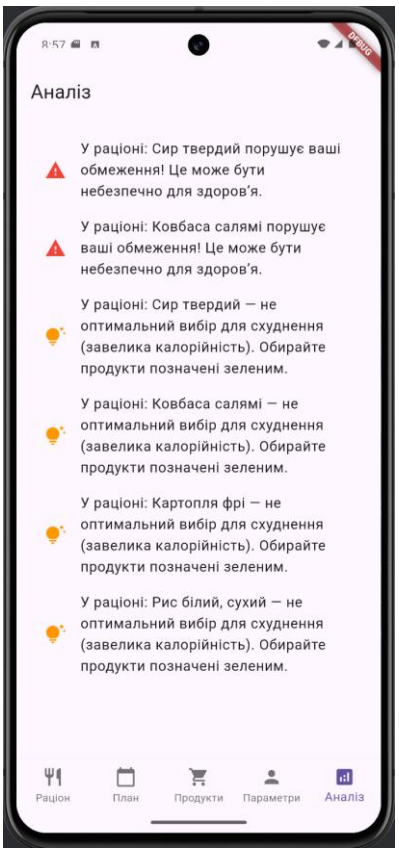



Рис. 10. Аналіз списку продуктів