Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему: «Проєктування та розробка Android-застосунку для ведення харчового щоденника та аналізу його впливу на досягнення фітнес-цілей»

Виконав: студент групи ІП321-1
спеціальність 121 «Інженерія програмного забезпечення»

Коломоєць С. В.
(прізвище та ініціали)

Керівник: к. ф.-м. н., доц. Лебідь О. Ю.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент: ДМСУ, Спеціалізоване управління розробки та супроводження програмного забезпечення, Департамент з питань цифрового розвитку, цифрових трансформацій та цифровізації
(місце роботи)
головний державний інспектор відділу розробки програмного забезпечення
(посада)
Бахтін О. В.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2025

АНОТАЦІЯ

Коломоєць С. В. Проєктування та розробка Android-застосунку для ведення харчового щоденника та аналізу його впливу на досягнення фітнес-цілей.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро 2025.

Об'єктом дослідження є процеси для забезпечення ведення харчового щоденника та відстеження впливу раціону на фітнес-результати за допомогою мобільного додатку.

Предмет дослідження – мобільний додаток на платформі Android, завданням якого – ведення харчового щоденника з використанням технологій мобільної розробки.

Метою роботи є залучення все більшої кількості людей до зміни їхнього способу життя шляхом розробки програмного забезпечення, що допомагає виправити негативні звички.

Дана робота присвячена проєктуванню та розробці Android-застосунку для ведення харчового щоденника, який надає можливості: контроль харчування, слідкування за водним балансом та пошук, переглядати і додавання рецептів страв або продуктів харчування до раціону. У процесі роботи проведено аналіз вимог до функціональності застосунку, розроблено його архітектуру за шаблоном MVVM, реалізовано кешування даних за допомогою Room, а також забезпечено інтуїтивно зрозумілий інтерфейс користувача. Тестування застосунку відбувалося на різних пристроях, що підтвердило його стабільність та зручність використання. Результатом роботи стало функціональне рішення, яке сприяє формуванню корисних харчових звичок та підтримці здорового способу життя.

Ключові слова: Android-застосунок, харчовий щоденник, Android Studio, Kotlin, MVVM, Room, REST API.

ABSTRACT

Kolomoets S. V. Design and development of an Android application for keeping a food diary and analyzing its impact on achieving fitness goals.

Qualification work for a bachelor's degree in the specialty 121 “Software Engineering.” – University of Customs and Finance, Dnipro 2025.

The object of the study is the processes for keeping a food diary and tracking the impact of diet on fitness results using a mobile application.

The subject of the study is a mobile application on the Android platform, the task of which is to keep a food diary using mobile development technologies.

The aim of the work is to encourage more and more people to change their lifestyle by developing software that helps to correct negative habits.

This work is devoted to the design and development of an Android application for keeping a food diary, which provides the following capabilities: nutrition control, water balance monitoring, and searching, viewing, and adding recipes for dishes or foods to the diet. In the course of the work, the requirements for the functionality of the application were analyzed, its architecture was developed using the MVVM pattern, data caching was implemented using Room, and an intuitive user interface was provided. The application was tested on various devices, confirming its stability and ease of use. The result of the work was a functional solution that promotes the formation of healthy eating habits and supports a healthy lifestyle.

Keywords: Android application, food diary, Android Studio, Kotlin, MVVM, Room, REST API.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення;

ОС – операційна система;

API – Application Programming Interface;

MVVM – Model-View-ViewModel;

МП – мова програмування;

ID – ідентифікатор об’єкту;

UI – візуальний інтерфейс користувача (User interface);

UX – досвід від користувацького інтерфейсу;

КБЖВ – калорії, білки, жири, вуглеводи.

ВСТУП

У сучасному цифровому світі дедалі більше людей ведуть малорухомий спосіб життя, проводячи значну частину часу вдома за екранами комп'ютерів та смартфонів. Оцифрування майже всіх сфер життя призвело до зменшення фізичної активності та соціальної взаємодії. Місцеві парки, театральні вистави, кінотеатри, бібліотеки та інші культурні й громадські місця втратили свою актуальність, адже майже всі ці речі доступні онлайн – фільми, книги, вистави, спілкування. Це зумовлює зниження мотивації до руху, прогулянок та активного дозвілля, що в свою чергу негативно впливає на фізичне та психологічне здоров'я людини.

Але, зважаючи на таку глобальну проблему, все більше людей прагнуть змінити своє життя, надихаючись прикладами з соціальних мереж та інших онлайн-платформ, де користувачі демонструють реальні результати завдяки веденню здорового способу життя. Це породжує попит на цифрові інструменти, які допомагають краще контролювати харчування, фізичну активність та досягнення бажаного результату. Саме тому все більшої популярності набувають сервіси для ведення харчового щоденника та відслідковування спортивної активності, які дозволяють своїм користувачам усвідомлено підходити до свого раціону, аналізувати дані і прогресію та формувати корисні звички. З часом такі сервіси стали більш доступними та зручними, оскільки, окрім веб-версій, з'явилися і мобільні додатки, що дозволяє кожному користувачу зі смартфоном легко користуватися ними будь-де та у будь-який час.

Актуальність роботи – вирішення однієї з ключової проблеми сучасних людей – зайва вага та ведення малорухомого способу життя, які негативно впливають на здоров'я та якість життя людини.

Метою роботи – залучення все більшої кількості людей до зміни їхнього способу життя шляхом розробки програмного забезпечення, що допомагає виправити негативні звички. Для цього буде використано мову

програмування Kotlin та середовище розробки Android Studio. Основні вимоги до ПЗ включають створення зручного та інтуїтивно зрозумілого користувацького інтерфейсу з використанням сучасних візуальних стилів і принципів Material 3 для забезпечення єдності інтерфейсу. Передбачається розробка аналітичної системи для відслідковування прогресу користувача на шляху до досягнення фітнес-цілей, впровадження рекомендаційної системи, яка сприятиме розвитку досягнень користувача. Однією з ключових цілей є забезпечення стабільної та ефективної роботи програмного забезпечення.

Методи дослідження – технології проєктування та реалізації програмного забезпечення, методи інформаційного моделювання, методи аналізу та інтерпретації даних, методи практичного тестування.

Для досягнення поставленої мети в роботі ставились та вирішувались наступні завдання:

1. Провести аналіз літературних джерел та наукових публікацій у галузі мобільних застосунків для фіксації харчування та контролю фітнес-цілей.

2. Дослідити існуючі рішення на ринку Android-додатків, визначити їх функціональні можливості, переваги та обмеження.

3. Виявити потреби потенційних користувачів і на основі цього сформулювати функціональні та нефункціональні вимоги до системи.

4. Сформулювати завдання для розробки мобільного додатку, що дозволяє вести харчовий щоденник та відслідковувати прогрес у досягненні фітнес-цілей.

5. Обґрунтувати вибір технологій, інструментів та платформи для реалізації Android-застосунку.

6. Розробити архітектуру застосунку, визначити структуру локальної бази даних та механізми збереження інформації про раціон.

7. Провести проєктування інтерфейсу користувача відповідно до сучасних принципів UI/UX-дизайну та вимог Material Design 3.

8. Інтегрувати сторонній API для розширення можливостей щодо пошуку та аналізу харчових продуктів.

9. Реалізувати функціональні можливості застосунку, включаючи збір, збереження, обробку та аналіз даних про харчування користувача.

10. Впровадити модуль аналітики, який дозволяє виявляти вплив харчових звичок на досягнення фітнес-цілей користувача.

11. Провести тестування застосунку, оцінити його зручність, стабільність, продуктивність та сформулювати рекомендації щодо подальшого вдосконалення.

Об'єкт дослідження – процеси для забезпечення ведення харчового щоденника та відстеження впливу раціону на фітнес-результати за допомогою мобільного додатку.

Головна увага виділяється реалізації функцій для забезпечення правильного збирання та обробки даних про харчування користувача, включаючи калорійність, вміст білків, жирів, вуглеводів, а також вітамінів і мікроелементів. Це вимагає наявності надійної та актуальної бази продуктів, яка дозволить користувачам швидко знаходити потрібні страви чи інгредієнти. введення харчового щоденника, такі як автоматичне рахування доступних інтеграцій реклами при натиску або запуску програми.

Предмет дослідження – мобільний додаток на платформі Android, завданням якого – ведення харчового щоденника з використанням технологій мобільної розробки. Предмет дослідження включає функціональні та програмні особливості Android-застосунку для фіксації харчування, аналізу даних та візуалізації впливу харчування на досягнення фітнес-цілей.

Предметом мобільного додатку, є його архітектура, модулі, способи і технології розробки таких додатків та компоненти, які визначають його функціональність, зручність використання та загальну ефективність.

В якості результату дослідження кваліфікаційної роботи – розроблено програмне забезпечення, яке дозволяє використовувати наступні функціональні можливості.

1. Створення персонального профілю з урахуванням фізіологічних параметрів користувача.

2. Розрахунок індивідуального плану харчування на основі введених даних.

3. Відображення щоденної норми КБЖВ та прогресу її виконання

4. Порівняння фактичного споживання КБЖВ з рекомендованими значеннями.

5. Автоматична корекція плану при зміні ваги користувача.

6. Щоденне збереження та перегляд харчової статистики за обрані дати.

7. Візуалізація прогресу за допомогою графіків та індикаторів.

8. Прогнозування досягнення цілей на основі динаміки харчування.

9. Введення кількості спалених калорій для точнішого балансу.

10. Нагадування про прийом їжі відповідно до заданого режиму.

11. Пошук і додавання страв та продуктів до щоденника харчування.

12. Фільтрацію рецептів, управління улюбленими продуктами та контроль відхилень.

Важливим аспектом є відповідність даного продукту потребам користувачів та ринковим потребам. У сучасному світі розвиток програмного забезпечення є динамічним процесом, тому важливо проводити постійний моніторинг ринку та аналізувати досвід розробки аналогічних програмних продуктів. Розроблений мобільний додаток має гарні перспективи та актуальність на ринку протягом багатьох років.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновку, списку використаних джерел та додатків. Робота містить 61 сторінку основного тексту, 23 рисунків, 5 таблиць та 30 літературних джерел.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. АНАЛІЗ ПРОБЛЕМАТИКИ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз літературних джерел предметної області

У сучасну цифрову епоху проблема ожиріння набула майже масштабів глобальної загрози для здоров'я населення. Малорухливий спосіб життя, зниження рівня фізичної активності та надмірне споживання висококалорійної їжі стали поширеними явищами у повсякденному житті людей у всьому світі. Разом із розвитком технологій, зокрема цифрових пристроїв і мобільних застосунків, відбулися значні зміни у способі ведення життя, що, з одного боку, сприяють комфорту, а з іншого – провокують розвиток хронічних захворювань, зокрема ожиріння. Незважаючи на глобальний характер цієї проблеми, ситуація в Україні також викликає занепокоєння: високий відсоток населення має надмірну вагу, що негативно впливає на загальний стан здоров'я людей. Реальність проблеми підтверджується результатами низки досліджень, у яких аналізується як загальний стан здоров'я людей, так і роль сучасних цифрових інструментів у зміні способу життя.

Так автори статі [1] наголошують, що дорослих людей в Україні, які мають проблеми з масою тіла приблизно 59% від всього населення, з яких майже 25% підпадають під критерії ожиріння. Окрім цього, у роботі [1] зазначено, що втрата навіть 5-10% ваги, вже може суттєво покращити здоров'я людей, але у разі швидкого схуднення – це може нести негативні наслідки для організму людини.

Також, у роботі [2] провели дослідження, через яке виявили, що близько 81% людей мали намір схуднути, і навіть робили декілька дій, які б змогли допомогти у виконанні цієї цілі. Але тільки близько 11% людей змогли загубити 5% від маси тіла та при цьому вдержати цей результат, не набравши зайвого кілограму.

Крім загроз для здоров'я, ожиріння має й суттєвий економічний вплив. У дослідженні [3] наголошується, що до 2060 року, Україна може втратити понад 21 мільярд доларів США через наслідки ожиріння, що дорівнює близько 5,69% від ВВП. Це обумовлено зниженням продуктивності, збільшенням витрат на охорону здоров'я, вищим рівнем інвалідності та передчасної смертності.

В результаті аналізу робот [1-3] видно, що проблема ожиріння є не лише масштабною, але й критично важливою, як з медичної точки зору, так і з економічною. Виходячи з проаналізованих робіт, постає питання – якими методами можна вирішувати цю проблему?

Одним із перспективних напрямів є використання цифрових технологій, зокрема мобільних додатків. Якщо раніше смартфон часто асоціювався з малорухливим способом життя, то нині він може стати інструментом для формування корисних звичок. У роботі [4] досліджено, як мобільні додатки сприяють мотивації молоді вести здоровий спосіб життя, займатися фізичною активністю та контролювати харчування. Значна частина таких застосунків орієнтована на активності (біг, ходьба, тренування) й одночасно містить функціонал для моніторингу харчування та водного балансу.

Також автори роботи [5] провели 41 дослідження, яке охопило 6348 учасників та показало, що використання мобільних додатків позитивно впливає на харчову поведінку і здоров'я, включаючи зменшення індексу маси тіла, артеріального тиску та рівня ліпідів у крові. Основні ефективні компоненти додатків включали: постановку цілей, моніторинг, надання зворотного зв'язку та соціальну підтримку.

Далі, у роботі [6] наведено дослідження, яке виявило, що мобільні додатки можуть допомогти пацієнтам у зниженні ваги, іноді навіть ефективніше, ніж традиційні методи. Проте успіх залежить від рівня залученості користувача, і існує потреба в більш тривалих дослідженнях, особливо серед різних демографічних груп.

Таким чином, джерела [4-6] демонструють, що мобільні додатки вже сьогодні активно використовуються як засіб зміни поведінки користувачів і можуть бути ефективним інструментом у боротьбі з надмірною вагою.

Більше того, мобільні додатки, що спеціалізуються на контролі харчування, показують реальні результати. У роботі [7] автори зазначають, що застосунки для фіксації споживаної їжі можуть бути ефективними навіть без дотримання спеціальних дієт.

Дослідження [8] показало, що до 50% активних користувачів застосунку SIMPLE досягли клінічно значущого зниження ваги (≥5%) всього за 12 тижнів.

Також, дослідження [9] підтверджує ефективність використання мобільних електронних пристроїв у зниженні ваги серед людей із надмірною вагою та ожирінням. Автори підкреслюють важливість розвитку "mHealth-рішень" у контексті загальносвітової боротьби з ожирінням.

Після проведеного аналізу літературних джерел, з результатів випливає проблема неконтрольованого харчування користувачів та невмотивованість цим займатися. Мобільні додатки можуть бути корисним інструментом у контролі харчування та зниженні ваги, особливо коли вони поєднують самостійний моніторинг, постановку цілей, зворотний зв'язок та соціальну підтримку. Однак їх ефективність значною мірою залежить від активності та мотивації користувача.

Для досягнення найкращих результатів рекомендується використовувати додатки як частину комплексного підходу до здорового способу життя, включаючи фізичну активність, збалансоване харчування та, за потреби, консультації з фахівцями.

Отже, результати досліджень [7,8,9] підтверджують доцільність і ефективність розробки мобільних додатків для контролю харчування та підтримки здоров'я. Такі інструменти вже демонструють позитивну динаміку серед користувачів і мають високий потенціал на ринку цифрового здоров'я.

1.2Аналіз існуючих рішень та їх функціональних можливостей

На ринку мобільних застосунків представлено безліч рішень, що допомагають користувачам контролювати різні аспекти здоров'я – від харчування та водного балансу до емоційного стану й менструального циклу. Серед них є як прості додатки, орієнтовані на виконання однієї конкретної функції, так і більш комплексні системи. До останніх належать такі застосунки, як «Eatfit», «YAZIO», «Lifesum», «Track – Calorie Counter», «Nutrilio» та інші.

Ці багатофункціональні рішення поєднують у собі можливості відстеження добової норми харчування та води, формування списків продуктів і рецептів, надання персоналізованих рекомендацій, а також аналітики результатів для подальшого коригування харчового плану. Саме ці можливості роблять їх потужним інструментом у боротьбі з неконтрольованим харчуванням.

«EatFit» – простий та легкий у використанні застосунок, призначений для контроля ваги, щоденного харчування та калорійності раціону. Його основна мета – допомогти користувачеві досягати фітнес-цілей шляхом базового моніторингу харчування. Розробники наголошують на доступності функціоналу та інтуїтивному підході до щоденного відстеження харчування, що робить додаток привабливим для широкого кола користувачів, зокрема новачків.

До основних функцій EatFit належать калорійний трекер, аналіз співвідношення білків, жирів і вуглеводів, а також база продуктів, яка дозволяє фіксувати прийоми їжі. Фокус зроблено саме на швидкість та простоту введення даних – користувачеві не потрібно витрачати багато часу на налаштування чи пошук функцій.

Користувачі позитивно відгукуються про легкість інтерфейсу, що дозволяє одразу почати користуватися застосунком без тривалого ознайомлення. Також вагомою перевагою є можливість роботи в офлайн-

режимі – навіть без підключення до інтернету користувач має повноцінний доступ до основного функціоналу, що особливо зручно під час подорожей або перебування у місцях із нестабільним зв'язком.

Втім, простота EatFit має і зворотний бік. Користувачі часто вказують на обмежену базу продуктів, що змушує витрачати додатковий час на створення власних позицій вручну. Попри те, що така функція доступна, її регулярне використання знижує загальний комфорт від роботи з додатком. Крім того, відсутність гнучкої персоналізації та інтеграції з іншими сервісами, зокрема Google Fit чи пристроями на зразок фітнес-годинників, робить EatFit менш ефективним для тих, хто хоче об'єднувати дані про активність і харчування в одній екосистемі. Додаток доступний на платформі Google Play за посиланням [10].

«YAZIO» – це один із найпопулярніших мобільних додатків для харчового трекінгу в Європі від розробника "YAZIO", орієнтований на допомогу в досягненні фітнес-цілей: зниженні ваги, її наборі або просто підтримці збалансованого раціону. Його основна мета – надати користувачеві простий, але ефективний інструмент для контролю харчування та самоспостереження. Розробники позиціонують YAZIO як персонального асистента зі здорового харчування та схуднення.

До основного функціоналу входить калорійний трекер, який дозволяє фіксувати спожиті продукти та аналізувати денне співвідношення білків, жирів і вуглеводів. Крім того, додаток має велику базу харчових продуктів, що значно полегшує внесення даних про їжу, а також інструменти для створення персональних планів харчування.

Серед переваг YAZIO користувачі часто відзначають зручний та естетично приємний інтерфейс, що робить процес ведення харчового щоденника швидким і не обтяжливим. Велика база продуктів дозволяє легко знаходити навіть специфічні позиції, а підтримка інтеграцій з популярними сервісами на кшталт Google Fit чи Fitbit сприяє більш комплексному підходу до контролю здоров'я.

Однак одним із головних недоліків додатку є те, що більшість розширених функцій – таких як готові плани харчування, детальна аналітика або деякі види звітів – доступні лише у преміум-версії. Це створює відчутну різницю між базовим і повноцінним досвідом користування, що може бути бар'єром для тих, хто шукає безкоштовне повнофункціональне рішення. Крім того, деякі користувачі вказують, що через велику кількість функцій інтерфейс потребує певного часу для звикання, особливо на старті користування. Застосунок доступний за посиланням [11].

«Lifesum» – фітнес-застосунок від компанії "Lifesum AB", який поєднує функції контролю харчування, фізичної активності та турботи про ментальне здоров'я. Його головна мета – допомогти користувачу побудувати збалансований спосіб життя, що охоплює не лише фізичні, а й психологічні аспекти здоров'я. Автори подають додаток як цифрового помічника для формування корисних звичок і тривалих змін у стилі життя.

Серед ключових функцій – трекінг харчування, водного балансу та ваги, а також можливість створення індивідуальних планів харчування на основі персональних цілей і потреб. Додаток також пропонує тести, які допомагають краще зрозуміти свій стиль харчування, та функції, спрямовані на підтримку ментального здоров'я, що робить Lifesum більш комплексним рішенням порівняно з класичними калорійними трекерами.

Однією з головних переваг цього застосунку користувачі вважають зрозумілий та зручний інтерфейс, завдяки якому додаток легко освоїти навіть новачкам. Важливою також є наявність персоналізованих порад, які допомагають орієнтуватися в процесі змін і адаптувати програму під власний стиль життя. Ще однією сильною стороною є кросплатформеність, що дозволяє користуватися додатком з різних пристроїв незалежно від операційної системи.

Проте, попри свій функціонал, застосунок має і певні недоліки. Через велику кількість функцій та інформації інтерфейс місцями виглядає перевантаженим, що ускладнює навігацію та може створювати бар'єри для

користувача при першому знайомстві з додатком. Окрім цього, більшість дієтичних програм і розширених можливостей доступні лише за платною підпискою, що знижує цінність базової безкоштовної версії для тих, хто шукає повноцінне рішення без фінансових витрат. Додаток доступний за посиланням [12].

«Track – Calorie Counter» – це безкоштовний мобільний застосунок, призначений для відстеження калорій, макронутрієнтів та контролю харчування. Його основна мета – надати користувачам простий та ефективний інструмент для щоденного моніторингу раціону, особливо в контексті схуднення або підтримання здорового способу життя. Застосунок позиціонується як інтуїтивне рішення без зайвих ускладнень, орієнтоване на широку аудиторію користувачів.

До ключових можливостей входить зручний пошук продуктів у великій базі даних Nutritionix, яка містить як стандартні продукти харчування, так і страви з популярних ресторанів. Також реалізовано функцію сканування штрих-кодів, яка дозволяє швидко додавати продукти у щоденник харчування. Додатковою перевагою є можливість введення власних рецептів і створення повторюваних страв для швидкого доступу.

Однак, незважаючи на свої сильні сторони, застосунок має певні недоліки. Користувачі відзначають обмежений функціонал персоналізації: наприклад, недостатню гнучкість у налаштуванні індивідуальних дієтичних цілей або макронутрієнтних співвідношень. Також відсутні адаптивні рекомендації чи поради на основі особистих даних, що змушує користувачів самостійно приймати рішення щодо змін у харчуванні. Іноді виникають зауваження щодо точності бази даних або складності навігації у певних розділах інтерфейсу.

Застосунок доступний безкоштовно на платформах Android та iOS. Додаткові функції, зокрема автоматичне розпізнавання голосового введення або інтеграція з фітнес-трекерами, можуть бути доступні у платній версії або з обмеженнями. Додаток доступний за посиланням [13].

«Nutrilio» – Застосунок для трекінгу харчування, настрою та водного балансу. Мета цього застосунку полягає у створенні усвідомленості щодо звичок харчування та впливу на самопочуття. Також автор подає цей застосунок як інтуїтивне рішення для самоспостереження за своїм здоров'ям.

З основних функцій, додаток надає щоденник їжі для ведення загальної харчової норми, можливість фіксування та будування календарю настрою, а також контролю водного балансу користувача.

З додаткових функцій, застосунок надає аналітичні інструменти для ведення статистики корисних звичок, також для будування графіку настрою ґрунтуючись на календар настрою, а також причини прийомів їжі.

З переваг користувачі зазначають що застосунок має фокус на психологічному аспекті, що не звично для такого роду застосунків, окрім цього, застосунок має мінімалістичний інтерфейс виконаний за сучасними рекомендаціями стосовно Flat дизайну, а також застосунок не потребує реєстрації користувача, тобто застосунок завантажується і одразу готовий до роботи.

Попри свої переваги, Nutrilio має також і низку недоліків. Користувачі відзначають слабку нутрітивну аналітику – застосунок не дозволяє детально відстежувати споживання КБЖВ. Також йому бракує інтеграції з популярними фітнес-сервісами, такими як Google Fit або Fitbit, що обмежує можливість об'єднання даних у межах екосистеми здоров'я. Для користувачів, які прагнуть більш повної аналітики та автоматизованого збору даних, ці обмеження можуть бути критичними. Додаток доступний за посиланням [14].

1.3 Визначення потреб користувачів та формування вимог до системи

Виходячи з проаналізованих існуючих рішень, можна сформувати такі базові потреби користувача:

– простота та швидкий старт без реєстрації;

– персоналізований контроль макронутрієнтів та води;
– велика й актуальна база продуктів, рецепти та повторювані страви;
– інтеграція з фітнес-сервісами та пристроями;
– адаптивні поради та гнучкі цілі;
– можливість офлайн-використання;
– інтерфейс, який не перевантажений, але дає доступ до глибокої функціональності при потребі.

Для розробки ефективної системи, яка дійсно задовольняє потреби кінцевих користувачів, важливо не лише проаналізувати існуючі рішення, але й чітко окреслити цільову аудиторію майбутнього застосунку. Розуміння типових сценаріїв використання, очікувань користувачів та їхніх основних вимог дає змогу сформувати обґрунтовані функціональні та нефункціональні вимоги до системи. У цьому підрозділі буде розглянуто характеристику цільової аудиторії, а також сформульовано ключові вимоги до Android-застосунку, що базуються як на аналізі популярних аналогів, так і на вивченні зворотного зв'язку від користувачів.

Розроблювана система має певні обмеження, пов'язані з віком та станом здоров'я користувача. Вона орієнтована на осіб віком від 10 до 80 років. Для дітей до 10 років, а також осіб старшого віку, які можуть мати вікові або хронічні захворювання, доцільніше користуватись спеціалізованими медичними програмами або звертатися до лікарів. Також система не враховує специфічні медичні стани, що впливають на харчування, тому призначена виключно для здорових користувачів без потреби в індивідуальному клінічному підході.

Цільовою аудиторією є середньостатистичні здорові особи, які прагнуть покращити свій раціон, контролювати харчування або досягти певних фітнес-цілей. Застосунок має відповідати типовим очікуванням цієї категорії користувачів – бути зручним, інтуїтивно зрозумілим, персоналізованим і водночас достатньо гнучким, аби покривати різні стилі харчування та щоденні звички.

Метою створення специфікації вимог є чітке окреслення функціоналу, необхідного для виконання основних завдань застосунку, а також визначення нефункціональних характеристик, таких як сумісність, продуктивність, безпечність і зручність інтерфейсу. Це дозволить обрати відповідні інструменти розробки, забезпечити якісну реалізацію системи та уникнути застосування застарілих або невиправдано складних технологій.

У рамках функціональних вимог розглянуто ключові групи функцій, які забезпечують роботу системи. Спочатку, увагу приділено персоналізації користувацького досвіду – від збору первинних даних до формування індивідуального плану харчування. Цей функціональний блок наведено у таблиці А.1

Далі, індивідуальний план харчування має бути не лише сформований, а й динамічно підтримуваний у процесі користування застосунком. Саме тому наступна група вимог зосереджена на зборі щоденних даних, їх візуалізації, аналітиці та взаємодії з користувачем через нагадування. Ці функції допомагають відстежувати прогрес, вчасно реагувати на відхилення та прогнозувати досягнення поставлених цілей. Відповідні вимоги наведено у таблиці А.2.

Для того щоб користувач міг повноцінно реалізовувати індивідуальний план, система повинна надати зручні інструменти для пошуку, фільтрації та вибору їжі й рецептів. Також важливо мати можливість швидко взаємодіяти з улюбленими продуктами. Третій функціональний блок зосереджений саме на цих аспектах роботи застосунку. Вимоги наведено у таблиці А.3.

Нефункціональні вимоги, натомість, визначають якість виконання функцій. Вони стосуються продуктивності, зручності, надійності та безпеки роботи застосунку. Хоча вони не впливають безпосередньо на функціональність, саме вони формують користувацький досвід і довіру до продукту. Крім того, нефункціональні вимоги допомагають визначити композицію всієї системи – який вид інтерфейсу варто використовувати, чи потрібна багаторівнева структура з розділенням логіки, можливість

зберігання даних, чи передбачено роботу в офлайн-режимі. Вони також задають очікувану поведінку застосунку: як він має поводитися при втраті підключення до мережі, при зміні орієнтації екрана, при довготривалому використанні або в умовах багатозадачності. Усі нефункціональні вимоги, які допоможуть наведені у таблиці 1.1.

Таблиця 1.1

Нефункціональні вимоги до системи

Назва вимоги	Опис вимоги
Робота без доступу до інтернет мережі	система повинна забезпечувати функціонал у офлайн режимі
Використання мінімалістичного інтерфейсу відповідно до принципів Material 3	система повинна мати дизайн інтерфейсу вико-ристовуючи компоненти та відповідаючи принципам Material 3 від Google
Адаптивність інтерфейсу до різних екранів пристроїв	система повинна мати адаптивність інтерфейсу до різних форматів екранів
Підтримка перемикання світлої і темної теми	застосунок повинен підтримувати автоматичне перемикання світлої та темної теми
Надійність системи	застосунок повинен правильно оброблювати виняткові ситуації, запобігаючи неочікуваному завершенню роботи та втраті даних
Продуктивність роботи	час запуску застосунку не повинен перевищувати 500 мс. Додаток повинен ефективно використовувати ресурси пристрою, не створюючи надмірного навантаження
Версія ОС Android	мінімальна підтримка ОС Android повинна бути не нижче версії 9.0 (API 28)
Кешування даних	система повинна кешувати завантажувальні дані, для відтворення їх у випадках відсутності доступу до мережі інтернет
Інтеграція з сервісом Google Fit	застосунок повинен мати можливість інтеграції з Google Fit для зчитування активності користувача
Можливість фонової роботи	система повинна мати можливість працювати у фоновому режимі для забезпечення синхронізації між Google Fit та застосунку

На основі визначених потреб користувачів і системних обмежень, було сформульовано перелік нефункціональних вимог, що визначають очікувану поведінку застосунку, доцільні інструменти реалізації та принципи взаємодії з користувачем. Вони створюють фундамент для розробки сучасного мобільного застосунку, який відповідає актуальним стандартам якості,

забезпечує надійний та інтуїтивно зрозумілий користувацький досвід, а також підтримує повноцінну інтеграцію з сервісом Google Fit.

Для кращого розуміння логіки взаємодії користувача із системою та загальної структури її функціоналу, було побудовано діаграму варіантів використання. Вона відображає основні сценарії роботи та реакції системи на дії користувача. Діаграма представлена на рисунку 1.1.

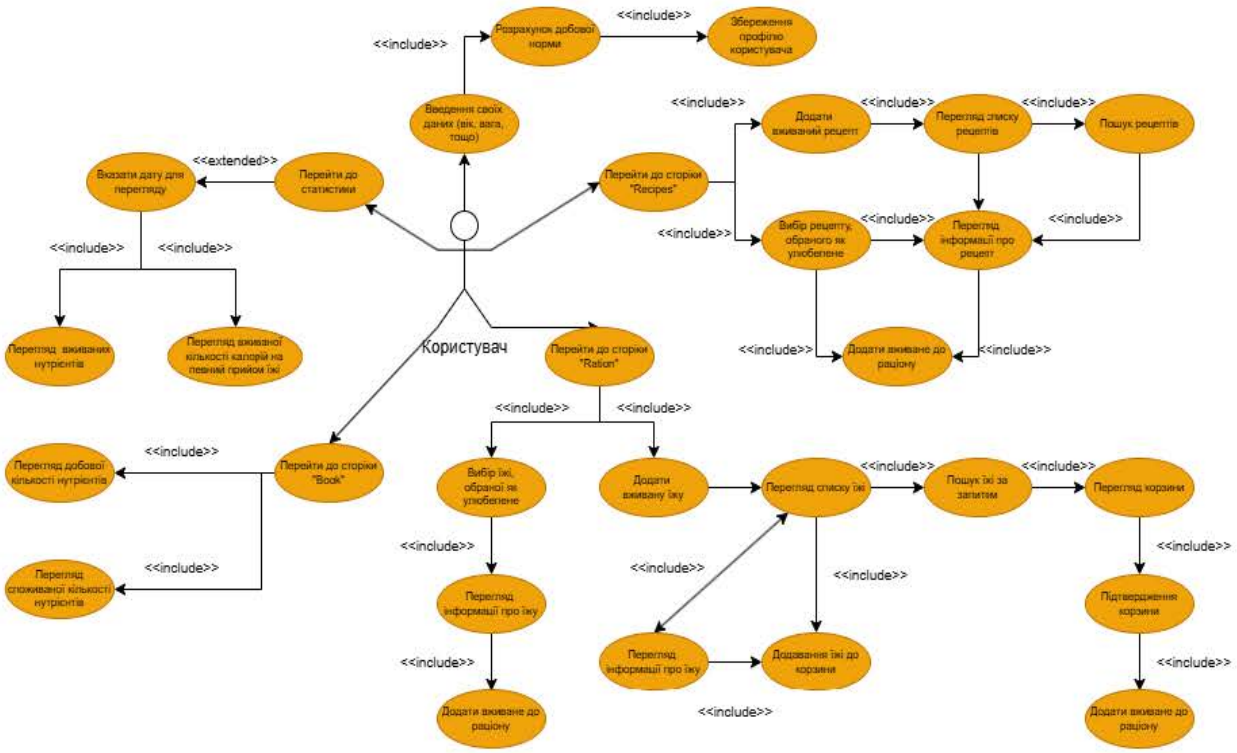

Рисунок 1.1 – Діаграма варіантів використання

1.4 Постановка завдання для розробки Android-застосунку

На основі аналізу літературних джерел та результатів дослідження предметної області можна стверджувати, що проблема незбалансованого харчування залишається актуальною як в Україні, так і у світі загалом. Низька обізнаність щодо принципів здорового раціону, відсутність системного підходу до контролю споживаних нутрієнтів та малорухливий спосіб життя сприяють поширенню ожиріння та дієтичних порушень серед населення.

Попри наявність великої кількості мобільних застосунків для фіксації харчування, багато користувачів все ще не знаходять оптимального інструменту для досягнення своїх фітнес- або оздоровчих цілей. У розділі 1.2 було розглянуто кілька популярних рішень на ринку, які дійсно мають попит, однак у них виявлено низку суттєвих недоліків – складний або перевантажений інтерфейс, відсутність персоналізації, обмежений функціонал або потреба в постійному підключенні до Інтернету. Ці зауваження були підтверджені у відгуках користувачів на платформі Google Play.

Метою розробки Android-застосунку є усунення виявлених під час аналізу недоліків існуючих рішень, а також створення ефективного інструменту для щоденного контролю харчування. Застосунок має не лише виконувати функцію харчового щоденника, а й сприяти досягненню особистих цілей користувача – зниженню або набору ваги, підтримці фізичної форми, контролю споживання КБЖВ та контролю водного балансу.

Розробка такого програмного продукту передбачає реалізацію усіх функціональних та не функціональних вимог до системи, які забезпечать:

- зручне ведення записів про прийоми їжі;
- підрахунок основних нутрієнтів та калорійності;
- візуалізацію даних у вигляді графіків;
- простий, інтуїтивно зрозумілий інтерфейс з дизайном Material 3.

Очікується, що результатом розробки стане стабільний, функціональний і користувацько-орієнтований мобільний застосунок, який дозволить ефективно відстежувати харчову поведінку та допомагатиме користувачеві у формуванні здорового стилю життя.

1.5 Висновки до першого розділу

Спочатку, у підрозділі 1.1 було проведено аналіз літературних джерел. Це дало змогу оцінити масштаби проблеми з зайвою вагою та ожирінням не

тільки в України, а і у всьому світі. Також були проаналізовані джерела, які розповідають про ситуацію на ринку мобільних застосунків щодо вже існуючих рішень, які частково намагаються виправити ситуацію, а також дійсно допомагають більшості людям.

Після аналізу літературних джерел предметної області, було проведено аналіз існуючих рішень у підрозділі 1.2, в рамках чого були оглянуті п'ять застосунків для цієї проблеми, а саме: «EatFit», «YAZIO», «Lifesum», «Tracke – Calorie Counter» та «Nutrilio». Було розглянуто їхнє призначення, тобто для чого вони створювалися, функціональні особливості, переваги та недоліки, які відмічають користувачі.

На основі зібраних даних, у підрозділі 1.3 було визначено загальні потреби користувачів, що стали основою для формування специфікації системи. Специфікація охоплює функціональні та нефункціональні вимоги, варіанти використання, а також загальні вимоги, яких система повинна дотримуватись. Сформульовані вимоги базуються на виявлених недоліках існуючих рішень на ринку та ключових потребах цільової аудиторії.

Враховуючи визначені вимоги та потреби користувачів, було сформульовано постановку завдання на розробку Android-застосунку у підрозділі 1.4. Він описує реалізацію основних етапів та очікуваних результатів.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ANDROID-ЗАСТОСУНКУ ДЛЯ ВЕДЕННЯ ХАРЧОВОГО ЩОДЕННИКА

2.1 Обґрунтування вибору платформи, інструментів та технології реалізації

З огляду на результати проведеного аналізу літературних джерел та існуючих рішень, можна зробити висновок, що формат мобільного застосунку є найбільш зручним для використання та повсякденного ведення харчового щоденника. Мобільний додаток забезпечує постійний доступ до функціоналу, підтримку офлайн-режиму та кращу інтеграцію з можливостями пристрою, що суттєво підвищує зручність для користувача порівняно з веб-сервісами.

В якості цільової мобільної платформи було обрано ОС Android, що зумовлено її відкритістю, доступністю серед великої кількості пристроїв різних цінових категорій та домінуванням на ринку мобільних ОС. Такий вибір дозволяє орієнтуватися на широку аудиторію з мінімальними стартовими витратами. Незважаючи на фрагментованість екосистеми, Android дає змогу швидко протестувати продукт, отримати користувацький фідбек і поступово вдосконалювати функціонал.

Для розробки програмного забезпечення орієнтованого на Android платформу, також треба обрати інструменти та середовища розробки. Для розробки такого застосунку, Google рекомендує використовувати Android Studio. «Android Studio» – є офіційним середовищем розробки для розробки Android-додатків від Google [16]. Воно побудоване на базі IntelliJ IDEA від JetBrains, що забезпечує високу продуктивність, інтелектуальні підказки, зручну навігацію, гнучке налаштування та потужні засоби для проведення рефакторингу коду. Android Studio також включає спеціалізовані інструменти для створення інтерфейсів, емуляцію пристроїв AVD (Android Virtual Device), моніторинг продуктивності та засоби налагодження, що дозволяє

охопити весь цикл розробки Android-застосунків. Така інтеграція всіх необхідних компонентів робить Android Studio оптимальним вибором для ефективної та професійної роботи над мобільними застосунками. Зовнішній вигляд Android Studio наведено на рисунку 2.1.

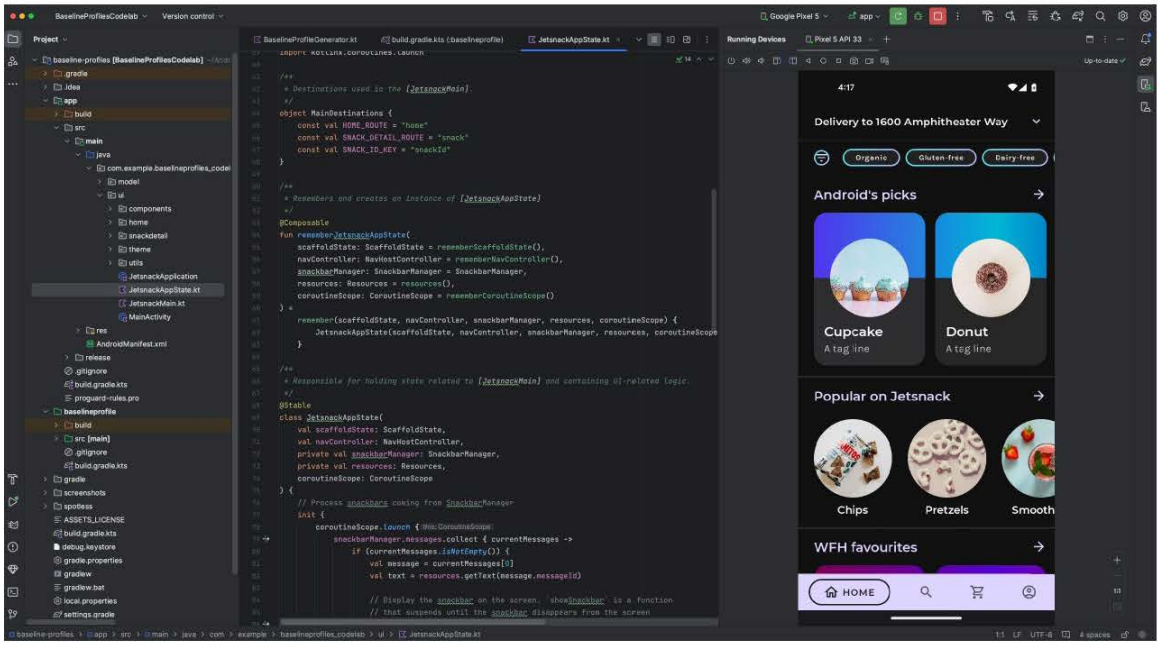

Рисунок 2.1 – Зовнішній вигляд середовища розробки «Android Studio»

Одним із недоліків цього середовища розробки є відносно високі системні вимоги до апаратної конфігурації системи, на якій виконується робота. На офіційному сайті Android Developers наведено такі мінімальні системні вимоги для роботи середовища на ОС Windows [17]:

– 64-bit Microsoft® Windows® 8/10/11;

– x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor;

– 8 GB RAM or more;

– 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator);

– 1280 x 800 minimum screen resolution.

Для своїх додатків, Android підтримує декілька мов програмування, а саме: Java, Kotlin та C++.

В якості мови програмування, для розробки застосунку було обрано Kotlin [18], який є сучасним та лаконічним рішенням, офіційно рекомендоване Google як основна мова для Android-розробки. На відміну від Java – Kotlin має значні переваги: читабельний синтаксис, компактність, знижена потреба у шаблонному коді та вбудована “null-безпека” [19], що допомагає уникати типових помилок під час виконання коду. Крім того, Kotlin має повну сумісність з Java, що дозволяє використовувати існуючі Java-бібліотеки та поступово інтегрувати Kotlin у вже наявні проєкти. Важливою перевагою Kotlin є підтримка корутин [20] – легковагових потоків, що спрощують реалізацію асинхронної логіки без складних зворотних викликів. Саме завдяки корутинам можлива ефективна та читабельна робота з мережевими запитами, базою даних та іншими фоновими операціями у застосунку.

Для реалізації локального середовища, яке зберігає дані у застосунку було обрано технологію «Room» – офіційну бібліотеку, яка є частиною “Android Jetpack” та спрощує роботу з вбудованою базою даних “SQLite” [21]. На відміну від застарілого підходу з використанням “SQLiteOpenHelper” – Room дозволяє працювати з базою даних через об’єктно-орієнтований інтерфейс. Замість написання SQL-запитів, розробник використовує анотації (@Entity, @Dao, @Query та інші), які автоматично генерують необхідний SQL-код під час збірки проєкту, підвищуючи безпеку та читабельність коду.

Room органічно інтегрується в архітектуру MVVM: DAO-класи (Data Access Objects) надають доступ до даних, ViewModel взаємодіє з ними й передає оновлення до View через LiveData або Flow. Такий підхід забезпечує реактивне оновлення інтерфейсу без зайвої логіки, що особливо важливо для мобільного додатку. Ще однією перевагою є підтримка GSON у поєднанні з “@TypeConverter”, що дозволяє зберігати складні типи – наприклад, списки або JSON-структури – у вигляді рядків у БД. Це робить Room зручним для збереження структурованих і вкладених даних, які часто зустрічаються у застосунках для харчового контролю.

Для перетворення даних у формат JSON – система відповідно до вимог, повинна використовувати бібліотеку Gson. «Gson» – це легковагова бібліотека від Google, призначена для серіалізації та десеріалізації об'єктів у формат JSON і навпаки [22]. Вона підтримує як прості, так і вкладені структури даних, забезпечує гнучке перетворення об'єктів без потреби у складній конфігурації та дозволяє працювати навіть із частково структурованими JSON-моделями.

Основною причиною вибору цієї бібліотеки – є її використання в інших бібліотеках та стабільність, надійність і офіційна підтримка з боку Google. Бібліотека добре документована, легко інтегрується у проєкт, має низький поріг входу для використання та не потребує додаткових залежностей. Gson дозволяє перетворювати складні об'єкти або списки об'єктів у JSON-рядки та зворотно, що є необхідним для зберігання даних у локальних базах, кешування або передачі між компонентами застосунку.

Також Gson використовується у інших популярних бібліотеках, що є одною з його переваг. Наприклад: Retrofit (для обробки мережевих запитів), Room (через @TypeConverter), Firebase та інші. Це забезпечує більшу сумісність між компонентами Android-застосунку та зменшує дублювання функціональності. Завдяки цьому Gson виконує роль універсального засобу для роботи з JSON у межах усієї архітектури застосунку.

Для взаємодії між сервером та клієнтом за архітектурним стилем REST API у застосунку, буде гарним вибором бібліотека «Retrofit», яка відзначається простотою, ефективністю та широкою підтримкою в Android-розробці [23]. На відміну від альтернатив, таких як Volley чи HttpUrlConnection – Retrofit забезпечує декларативний підхід до опису запитів, підтримує асинхронну роботу, обробку помилок та легко інтегрується в сучасну архітектуру. Бібліотека тісно сумісна з Gson, що дозволяє автоматично перетворювати JSON-дані у Kotlin-об'єкти. В основі Retrofit працює клієнт OkHttp, який відповідає за стабільні з'єднання, логування та кешування [24]. Завдяки використанню корутин Kotlin [20],

виклики API реалізуються у вигляді “suspend” функцій, що значно спрощує асинхронну обробку запитів.

2.2 Архітектура застосунку та проєктування структури локальної бази даних Room

Для успішного проєктування архітектури Android-застосунку важливо дотримуватися сучасних принципів і рекомендацій, зокрема тих, що надає Google у межах концепції App Architecture, представленої на порталі Android Developers [15]. Вони спрямовані на створення стійкої, масштабованої та легко підтримуваної структури, яка дозволяє зменшити технічні ризики, розділити обов’язки між компонентами та забезпечити зрозумілу взаємодію між рівнями візуального інтерфейсу, логіки та доступу до даних. Згідно з цієї архітектурної моделі, система поділяється на окремі шари: UI Layer, Domain Layer (опціонально) та Data Layer. Кожен із них має чітко визначену відповідальність і дозволяє будувати ізольовані, незалежні та зручні для тестування компоненти. Сторінка «Guide to app architecture» [15] надає візуальну схему взаємодії цих шарів, яка представлена на рисунку 2.2.

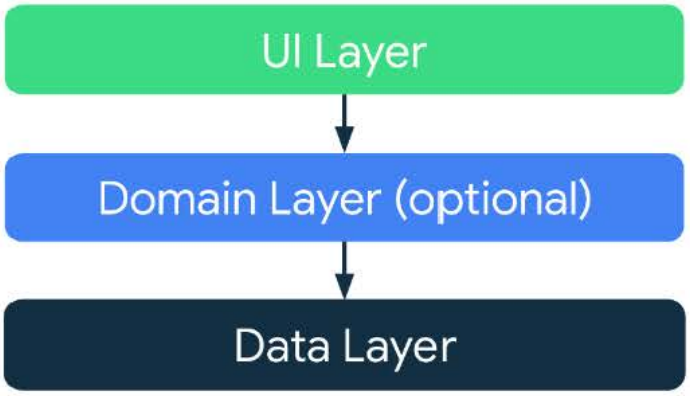

Рисунок 2.2 – Діаграма типової архітектури застосунку

«UI Layer» (Presentation Layer) у App Architecture відповідає за відображення інтерфейсу користувача та обробку взаємодії з ним. Тут

розміщуються всі View-елементи (екрани, фрагменти, класичні View), а також логіка зміни інтерфейсу – наприклад, активація кнопок чи показ помилок. Цей шар працює зі станами з ViewModel і не містить бізнес-логіки, що дозволяє чітко розділити відповідальність у застосунку.

«Domain Layer» – опціональний шар, який містить бізнес-логіку у вигляді use case-ів і описує інтерфейси репозиторіїв, що взаємодіють із Data Layer. Це забезпечує незалежність логіки від реалізації, покращує тестування та повторне використання коду.

«Data Layer» відповідає за доступ до даних із зовнішніх джерел – API, баз даних чи файлів. Тут розміщуються джерела даних та репозиторії, які реалізують інтерфейси з "Domain Layer". Репозиторії слугують посередниками й надають єдиний спосіб доступу до інформації в застосунку.

Для App Architecture рекомендується обирати архітектурний шаблон MVVM тому це буде одним із ефективніших рішень при розробці застосунку. Така комбінація дозволяє чітко розмежувати відповідальність між різними рівнями застосунку, полегшує внесення змін у функціонал без порушення цілісності системи, а також забезпечує можливість легкого тестування окремих компонентів.

Архітектурний шаблон MVVM передбачає поділ логіки застосунку на три основні частини:

- View – інтерфейс користувача;
- ViewModel – посередник між View та Model;
- Model – дані й бізнес-логіка.

ViewModel виступає незалежним компонентом, який не має прямої прив'язки до інтерфейсу, що дозволяє повторно використовувати логіку, легше її тестувати та керувати життєвим циклом. Важливо, що фрагменти або активності не взаємодіють безпосередньо з джерелами даних, а звертаються до ViewModel, яка вже виконує відповідні запити та обробку. Завдяки цьому досягається чистота коду, його модульність та спрощується підтримка навіть у довготривалих проєктах.

Для кращого розуміння взаємодії застосунку з зовнішнім середовищем було створено контекстну діаграму за методологією IDEF0. Вона подає застосунок як «чорну скриньку», яка приймає вхідні дані, обробляє їх відповідно до внутрішньої логіки та повертає результати обробки. Такий підхід дозволяє наочно окреслити межі системи та її взаємодію з зовнішніми елементами. Діаграма представлена на рисунку 2.3.

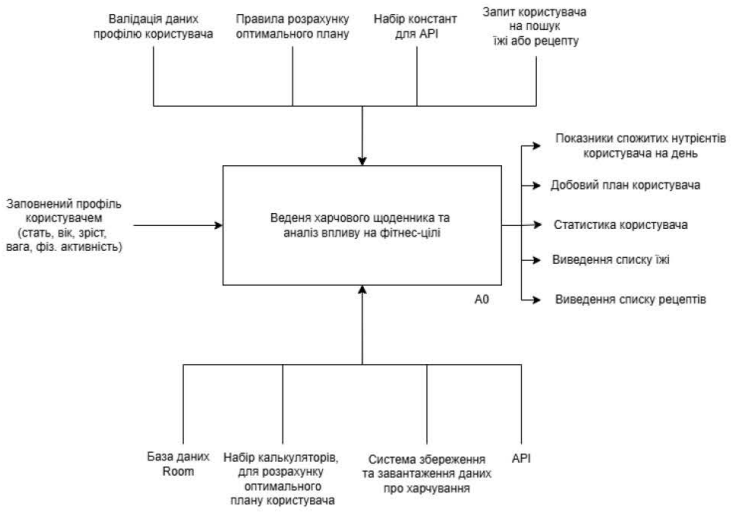

Рисунок 2.3 – Контекстна діаграма за методологією IDEF0

Для глибшого розуміння архітектури системи та взаємозв'язків між її основними компонентами було побудовано діаграму першого рівня за методологією IDEF0. Діаграма 1-го рівня наведена на рисунку 2.4.

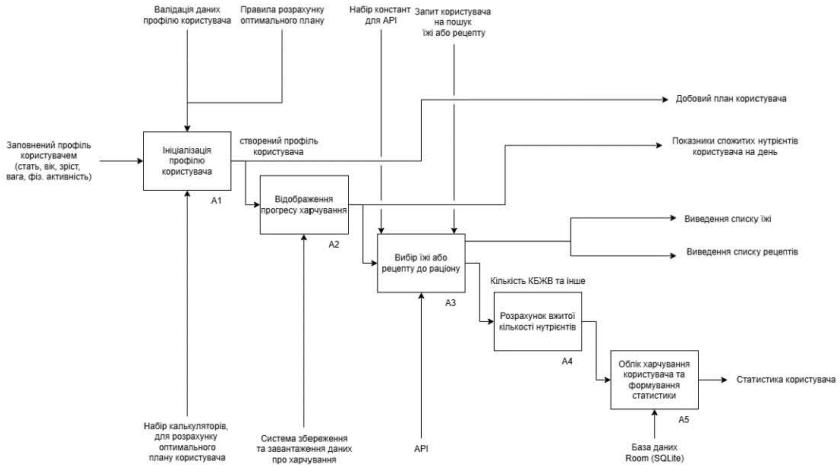

Рисунок 2.4 – Діаграма 1-го рівня за методологією IDEF0

Для моделювання послідовності подій та динаміки виконання основних процесів у системі було побудовано діаграму IDEF3. Вона дозволяє відобразити логіку виконання сценарію з урахуванням часових залежностей, умов переходу та варіативності дій. У даному випадку діаграма демонструє типовий процес додавання користувачем продукту до раціону – від ініціації дії до збереження результатів у системі. Діаграма наведена на рисунку 2.5.

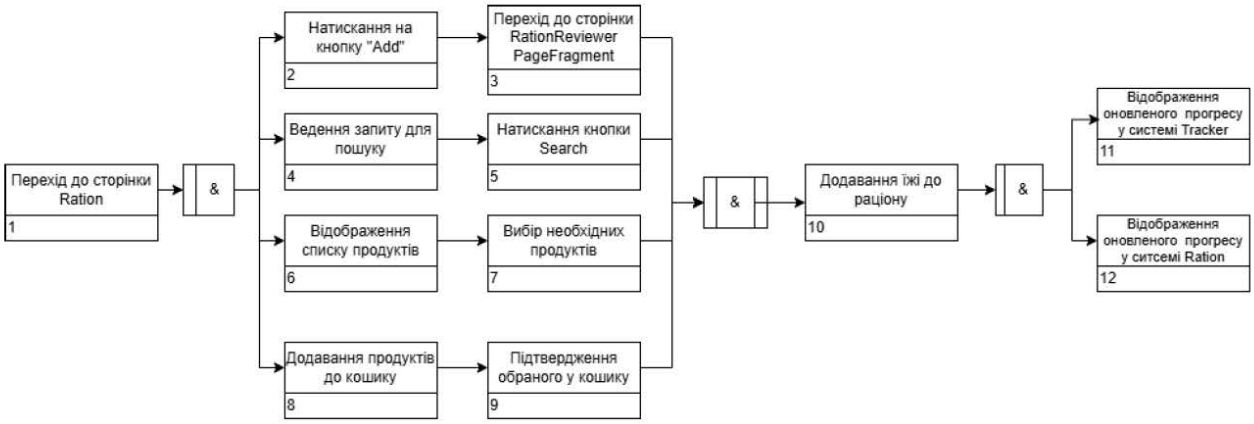

Рисунок 2.5 – Діаграма за методологією IDEF3

Далі було створено діаграму послідовностей, яка ілюструє етапи взаємодії між об’єктами під час роботи користувача з основними системи застосунку. Діаграма наведена на рисунку 2.6.

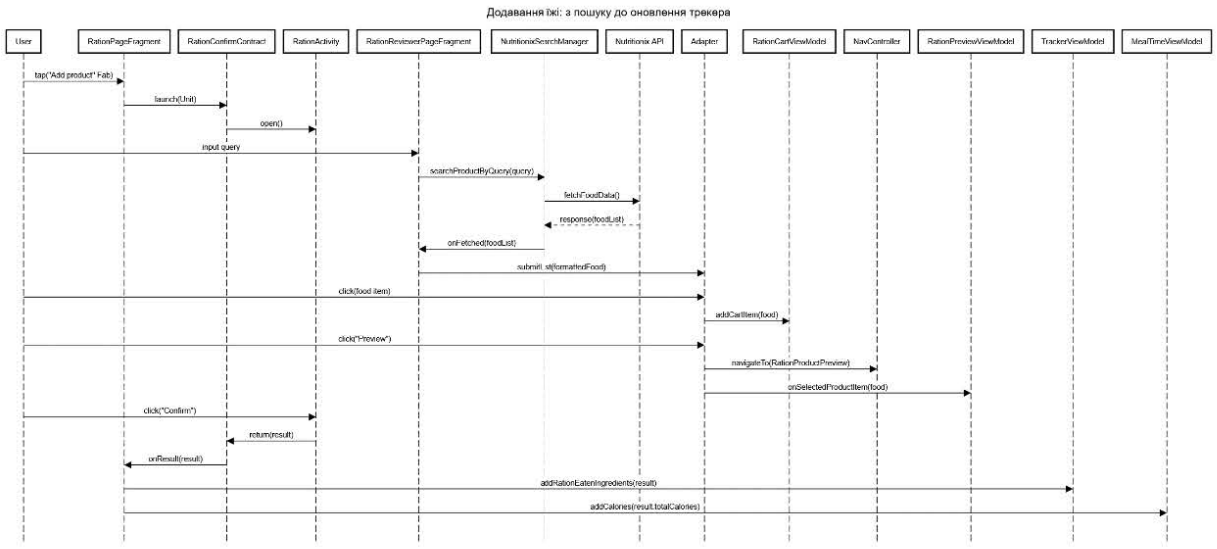

Рисунок 2.6 – Діаграма послідовностей

Виходячи з наведених діаграм, можна визначити загальну архітектуру застосунку та принципи його функціонування. Окремі компоненти системи використовують локальне середовище для збереження і обробки даних користувача. У цьому контексті було обрано використання локальної бази даних, реалізованої за допомогою бібліотеки Room, яка є частиною Android Jetpack. Room надає зручний ORM шар поверх SQLite, що спрощує роботу з базою даних та забезпечує безпечний доступ до неї через об'єктну модель.

Застосунок використовує кілька таблиць спільної бази даних QCB_IntakesDatabase, кожна з яких має чітко визначену функцію у контексті збереження та аналізу даних, пов'язаних із харчуванням користувача.

Мета використання бази даних – забезпечення збереження, структурування та доступу до інформації, необхідної для щоденного аналізу харчування, формування звітності та персоналізованих рекомендацій. Серед збережених даних – історія спожитих продуктів, статистика нутрієнтів за кожен день та інформація про улюблені продукти.

Відповідно до вимог системи, база даних охоплює всі ключові дані, необхідні для повноцінного аналізу харчової поведінки користувача. Зокрема, зберігається інформація про кількість і склад спожитих макронутрієнтів, дата та контекст споживання, що дозволяє у будь-який момент отримати точну картину харчування користувача за будь-який обраний день з моменту початку використання застосунку.

Однією з ключових таблиць є QCB_MealEntries, яка реєструє кожен продукт, що був позначений користувачем як спожитий. Коли користувач знаходить необхідні продукти, додає їх до кошика та підтверджує їх вживання – система передає дані до системи Tracker. Tracker, у свою чергу, оновлює візуальний інтерфейс і зберігає записи до таблиці QCB_MealEntries.

Ця таблиця включає такі поля, як унікальний індекс, назва продукту, дата та час вживання, а також детальний склад нутрієнтів. Дані з неї активно використовуються модулем Statistics для формування добових звітів про спожиті продукти. Структуру таблиці QCB_MealEntries подано на рис. 2.7.

Рисунок 2.7 – Структура таблиці QCB_MealEntries

Збереження щоденної статистики є необхідним для коректної роботи сторінки Statistics, яка відповідає за відображення детального звіту про харчування користувача за обраний день та системи аналітики Analytics, яка аналізує прогресію користувача. Після вибору конкретної дати, сторінка Statistics завантажує відповідні дані з бази та формує узагальнену картину спожитих КБЖВ. У звіті, користувач отримує інформацію про загальну кількість спожитих калорій, співвідношення макронутрієнтів, а також індивідуальні характеристики, що були враховані під час аналізу

Організація та збереження цих даних реалізована у таблиці QCB_DailyNutritionHistory, яка акумулює усі необхідні значення в розрізі добового споживання. Структура таблиці зображена на рисунку 2.8.

Рисунок 2.8 – Структура таблиці QCB_DailyNutritionHistory

Застосунок також підтримує функціональність формування списку улюблених продуктів, що дає змогу користувачу швидко додавати популярні

або часто вживані позиції до щоденного раціону без повторного пошуку. Ця можливість реалізується за допомогою таблиці QCB_FavoriteProducts .

Під час натискання кнопки “Favorite” у картці продукту, відповідна інформація зберігається у таблиці, що дозволяє згодом швидко отримати доступ до обраних елементів. Повторне натискання цієї ж кнопки призводить до видалення продукту зі списку. Таблиця QCB_FavoriteProducts зберігає ключову інформацію про кожен обраний продукт, а саме: його назву, дату додавання до списку, а також унікальний індекс, що дозволяє однозначно ідентифікувати запис. Структура таблиці наведена на рисунку 2.9.

Рисунок 2.9 – Структура таблиці QCB_FavoriteProducts

Також, окрім харчування застосунок підтримує функцію контролю водного балансу користувача. Ця функція аналогічна до додавання продуктів до раціону, тому було для зберігання даних щодо споживання води, потрібно дві таблиці: QCB_WaterEntries та QCB_DailyWaterHistory.

Таблиця QCB_WaterEntries відповідає за зберігання списку вживаних склянок води. Таблиця містить дані про: час вживання та об’єм води. Структура таблиці наведена на рисунку 2.10:

Рисунок 2.10 – Структура таблиці QCB_WaterEntries

Таблиця QCB_DailyWaterHistory, аналогічно до таблиці QCB_Daily-NutritionHistory – зберігає дату запису, сумарний об'єм вживаної води та необхідну добову кількість для досягнення норми на добу. Структура таблиці наведена на рисунку 2.11.

Рисунок 2.11 – Структура таблиці QCB_DailyWaterHistory

В результаті, було спроектовано базу даних QCB_IntakesDatabase, яка має таблиці: QCB_MealEntries, QCB_DailyNutritionHistory, QCB_Favorite-Products, QCB_DailyWaterHistory та QCB_WaterEntries. Усі таблиці цієї бази даних не мають зв'язків через те, що база даних використовується як самостійний UI-журнал. Схема бази даних наведена на рисунку 2.12.

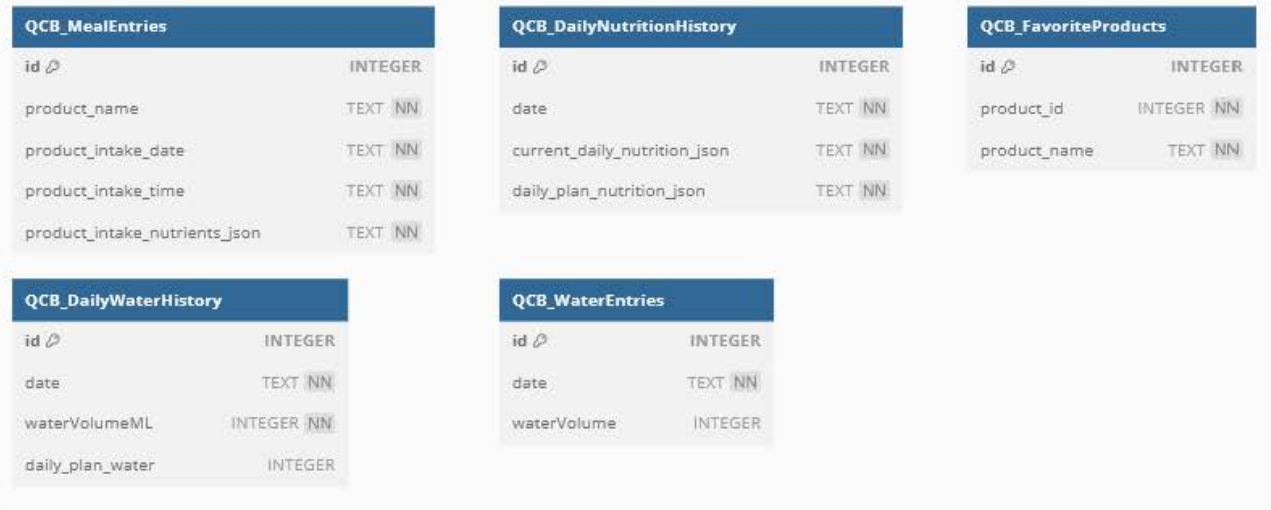

Рисунок 2.12 – Структура таблиці QCB_DailyWaterHistory

2.3 Проєктування інтерфейсу користувача UI/UX

Проєктування інтерфейсу користувача (UI) та досвіду взаємодії (UX) є один з ключових етапів у розробці Android-застосунку, адже саме через інтерфейс відбувається основна комунікація користувача з системою. Основна мета цього процесу – створити інтуїтивно зрозумілий, зручний і привабливий інтерфейс відповідно до вимог та підходів, який сприятиме ефективному виконанню завдань користувача.

На початку, було визначено основну кольорову палітру застосунку орієнтуючись на палітру «кольорів настрою». Візуальний інтерфейс повинен відповідати цій палітрі, оскільки такі кольори створюють емоційний зв'язок із призначенням застосунку та його основною метою – підтримка позитивного користувацького досвіду. В якості основного кольору було обрано колір з назвою "Pink" (код кольору: FFDBD2), який створює акцент на енергійності та дружньої атмосфери, крім цього, наведений колір лягає в основу кольорової схеми застосунку.

Далі, для проєктування інтерфейсу користувача існує декілька підходів, які мають певну мету та свої переваги і недоліки. Але конкретно для Android-застосунку буде правильно використовувати підхід Material Design, який є досить поширений на мобільному ринку. Його мета полягає в тому, щоб використовувати вже готову дизайн-систему від компаній Google, Microsoft, Apple та інших. Для Android-застосунків рекомендовано використовувати більш сучасний варіант – Material Design 3 [26], розроблений компанією Google. Враховуючи сучасні вимоги до дизайну, доцільно обрати саме дизайн-систему Material Design версії 3, оскільки він вирізняється оновленим візуальним стилем, розширеними можливостями персоналізації та активною підтримкою з боку Google. З можливостей цієї дизайн-системи, можна виділити швидкість розробки за рахунок готових UI-компонентів, послідовність і узгодженість інтерфейсу та сучасний візуальний стиль. Завдяки наявності готових компонентів немає потреби створювати власні

графічні елементи з нуля, що суттєво пришвидшує процес розробки. Крім того, сучасний візуальний стиль Material Design є приємним і добре знайомим звичайному користувачу, адже саме його Google використовує у своїх сервісах, які зазвичай уже встановлені на кожному Android-пристрої за замовчуванням.

Застосунок повинен мати обмежену кількість екранів для зручної навігації. Враховуючи основні функції – вибір продуктів, рецептів і перегляд харчової статистики – головна сторінка містить три основні фрагменти: TrackerPageFragment, RationPageFragment та RecipesPageFragment. Tracker-PageFragment відповідає за відображення добового харчування: шкала калорій, спожиті макронутрієнти, а також поточний прийом їжі відповідно до часу. Також звідси можна перейти до статистики. RationPageFragment показує всі прийоми їжі, калорійність кожного з них і список улюблених продуктів. Є кнопка додавання продукту, яка веде до сторінки пошуку їжі. RecipesPageFragment аналогічно відображає улюблені рецепти та дозволяє додавати нові, переходячи на сторінку їх пошуку.

Усі основні екрани головної сторінки MainActivity представлені на рисунку 2.13.

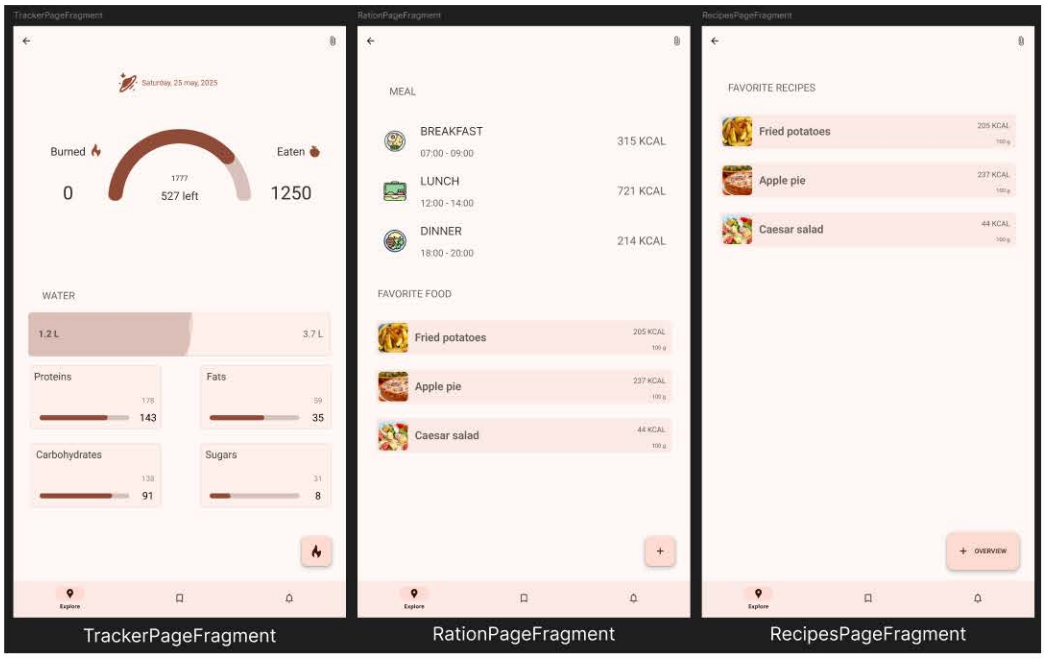

Рисунок 2.13 – Структура основних екранів головної сторінки

Під час проєктування інтерфейсу взаємодії користувача з функціоналом вибору продуктів харчування, було розроблено послідовний цикл, реалізований навколо трьох основних екранах: RationReviewerPage-Fragment, RationOverviewPageFragment та RationCartPageFragment.

Основна мета – забезпечити інтуїтивно зрозумілий і зручний шлях від пошуку продукту до його додавання в щоденний раціон.

Інтерфейс RationReviewerPageFragment розроблений з фокусом на простоту пошуку. Поле введення запиту доступне одразу при відкритті, а виконання пошуку ініціюється натисканням кнопки “Search” на клавіатурі.

Отримані результати виводяться у вигляді скролюваного списку карток, де кожна картка містить коротку, але інформативну візуалізацію: назву продукту, калорійність, розмір порції, зображення та кнопку “Add”. Кнопка виконує швидку дію – додавання продукту до кошика з типовою порцією у 100 г, без потреби переходу на інші екрани.

Для користувачів, які бажають ознайомитися з деталями, передбачено навігацію до RationOverviewPageFragment.

Дизайн цього екрану орієнтований на глибше занурення в інформацію про продукт: відображаються КБЖВ, можливість змінити розмір порції, додати до улюбленого чи одразу до кошика. Основний акцент – зрозуміле структурування даних і доступність основних дій.

Завершує взаємодію RationCartPageFragment – інтерфейс для перевірки і фінального підтвердження вибору. У списку користувач може видаляти продукти або переглядати їх знову.

Нижня панель надає миттєвий зворотний зв’язок у вигляді сумарної калорійності та розміру порцій.

Кнопка “Confirm” завершує цикл, підтверджуючи вибір і повертаючи користувача до головної сторінки.

Дизайн інтерфейсу усіх екранів в рамках активності RationActivity представлено на рисунку 2.14.

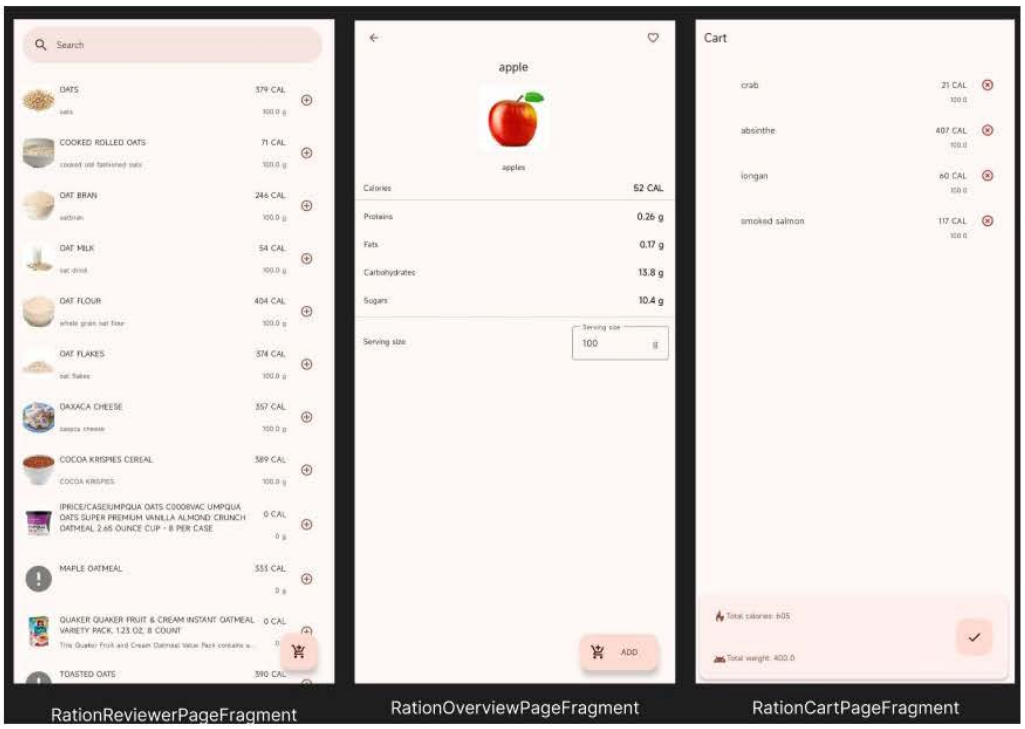

Рисунок 2.14 – Інтерфейс фрагментів системи Ration

Система вибору рецептів має схожий принцип роботи та навігації, як і у випадку з продуктами, проте дещо відрізняється у візуальній подачі інформації про рецепт. Основні етапи взаємодії – пошук, перегляд та додавання рецепту до раціону – реалізуються за допомогою фрагментів RecipeReviewerPageFragment та RecipeOverviewPageFragment, які обслуговуються активністю RecipesActivity.

Фрагмент RecipeReviewerPageFragment, аналогічно до RationReviewerPageFragment, дозволяє користувачеві вводити запит у поле пошуку, після чого система виконує пошук рецептів за API та відображає відповідні результати у вигляді списку.

В свою чергу, фрагмент RecipeOverviewPageFragment відповідає за детальний перегляд обраного рецепту. Інтерфейс цього екрану містить повну інформацію про рецепт: назву, загальний опис, час приготування, перелік інгредієнтів, необхідне кухонне обладнання, а також покрокову інструкцію приготування. Візуальне оформлення спрямоване на зручне сприйняття контенту та легкий доступ до основних функцій. Структура екранів, що реалізуються у RecipesActivity, представлена на рисунку 2.15.

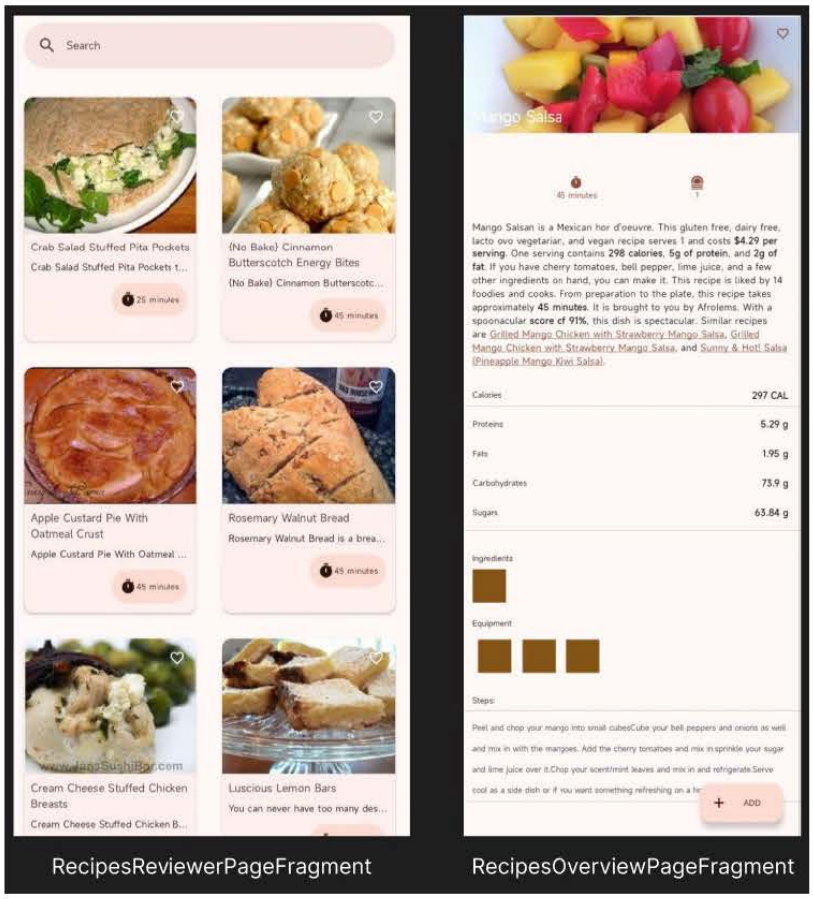

Рисунок 2.15 – Інтерфейс екранів для роботи з рецептами у системі Recipes

Окрім основних екранів, застосунок містить допоміжні сторінки, доступні через NavigationDrawer, що реалізує Android SDK. Всі сторінки – Settings, Water, Profile, Analytics та About – відкриваються у межах ExtraActivity, яка динамічно завантажує відповідні фрагменти.

– на сторінці Settings, користувач може змінювати одиниці вимірювання та керувати інтеграцією з Google Fit;

– сторінка Water відображає інформацію про план та прогрес користувача у водному балансу;

– сторінка Profile містить дані профілю, який користувач створив на початку при першому завантажені застосунку. У разі потреби, на цій сторінці можна відредагувати конкретні дані профілю;

– сторінка Analytics показує аналіз прогресії користувача, де аналізуються відхилення від поточного плану, критичність цього відхилення та прогноз досягнення мети;

– на сторінці About представлена інформація про застосунок, його опис та підтримку.

Візуальний інтерфейс фрагментів активності ExtraActivity: SettingsFragment, ProfileFragment, WaterBalanceFragment, AnalyticsFragment, AboutFragment, а також активність статистики – StatisticsActivity, наведено на рисунках А.1, А.2 та А.3.

2.4 Інтеграція з Spoonacular API

Для повного забезпечення функціоналу, є необхідність у великий базі даних про харчові продукти, що містять загальну інформацію та детальний склад макронутрієнтів. Крім того, система потребує доступ до бази кулінарних рецептів, яка включає опис страви, час його приготування, покрокові інструкції та харчову цінність.

З метою задоволення цих вимог у застосунок було інтегровано Spoonacular API – зовнішній сервіс, який надає доступ до великої харчової бази даних і тисячі рецептів із детальними характеристиками макронутрієнтів. Завдяки цьому, під час запиту користувача за назвою продукту або рецепту – застосунок звертається до бази даних Spoonacular API за REST API та отримує результати даних у вигляді JSON форматі. Потім додаток оброблює ці дані і виводить їх у зручному візуальному інтерфейсі, де окремо представлена калорійність їжі та БЖВ.

З переваг наведеного API, можна явно вважати велику кількість записів харчових продуктів та кулінарних страв у базі, які можна приготувати. Окрім цього, API розподіляє харчові продукти на дві групи: інгредієнти та продукти. До інгредієнтів входять звичайні, не оброблені продукти харчування, наприклад: яблуко, груша, огірок, цибуля та інше, тобто те з чого складається продукт. До продуктів входять вже та їжа, яка була приготована з інгредієнтів, наприклад: бургер, яблучне пюре, салат та інше. Крім даних про окремі продукти, API надає доступ до великої кількості

різноманітних рецептів. Відповідь сервера має уніфіковану структуру, що включає інформацію про нутрієнтний склад страви. Окрім цього, повертається розширений набір даних: перелік інгредієнтів, час приготування, етапи приготування, показники оцінки рецепту від Spoonacular, індекс корисності (HealthScore), а також посилання на оригінальне джерело рецепту з відповідною ліцензією. Це робить API потужним інструментом для інтеграції кулінарного функціоналу в мобільний застосунок.

Але попри зручність і широкі можливості Spoonacular API, існує суттєвий недолік, що може обмежити його використання для більшості користувачів – це вартість, яка напряму залежить від кількості запитів до сервера. Безкоштовний тариф має значні обмеження, а для повноцінного використання (наприклад, частих пошуків продуктів або генерації рецептів) необхідно перейти на платний план.

2.5 Висновки до другого розділу

У розділі 2 було детально обґрунтовано технічні рішення, прийняті при проєктуванні та реалізації Android-застосунку.

У підрозділі 2.1 розглянуто вибір платформи, інструментів та технологій. Як цільову платформу обрано Android завдяки її популярності серед користувачів та широким можливостям налаштування. Для реалізації застосунку використано Android Studio як основне середовище розробки, мову програмування Kotlin, а також сучасні технології, зокрема Room для роботи з локальною базою даних, Gson для обробки JSON-даних і Retrofit для взаємодії з API. Вибір цих інструментів обумовлений їхньою зручністю, продуктивністю та активною підтримкою спільноти.

У підрозділі 2.2 було розглянуто архітектуру застосунку, що базується на шаблоні MVVM, який забезпечує чітке розділення логіки, зручність тестування та масштабованість. Також наведено діаграми IDEF0 різних

рівнів, IDEF3 і діаграму послідовності, які ілюструють логіку роботи системи та взаємодію між її компонентами. Окремо увагу приділено проєктуванню локальної бази даних QCB_IntakesDatabase, зокрема було описано структуру таблиць QCB_MealEntries, QCB_DailyNutritionHistory, QCB_Favorite-Products, QCB_DailyWaterHistory та QCB_WaterEntries, що зберігають історію споживання продуктів та обрані користувачем продукти.

У підрозділі 2.3 було здійснено проєктування користувацького інтерфейсу з дотриманням принципів Material Design від Google. Основним кольором застосунку обрано відтінок рожево-червоного, що відповідає темі мотивації та емоційної залученості. Розглянуто основні екрани головної активності MainActivtiy застосунку: TrackerPageFragment, RationPage-Fragment та RecipesPageFragment. Окрім цього, були розглянуті додаткові екрани активності ExtraActivity: SettingsFragment, ProfileFragment, WaterBalanceFragment, AnalyticsFragment, AboutFragment.

У підрозділі 2.4 описано інтеграцію з API Spoonacular, яка дозволяє розширити функціональність застосунку за рахунок отримання рецептів та інформації про продукти з великої онлайн-бази. Було проаналізовано мету інтеграції, а також переваги у вигляді автоматизованого пошуку рецептів і недоліки, пов'язані з обмеженнями API та залежністю від стороннього сервісу. Таким чином, у результаті проєктування було сформовано цілісне бачення архітектури, бази даних, інтерфейсу та зовнішніх інтеграцій, що дозволяє забезпечити надійність, зручність та ефективність майбутнього Android-застосунку.

РОЗДІЛ 3. РОЗРОБКА ANDROID-ЗАСТОСУНКУ ДЛЯ ВЕДЕННЯ ХАРЧОВОГО ЩОДЕННИКА ТА АНАЛІЗУ ЙОГО ВПЛИВУ НА ДОСЯГНЕННЯ ФІТНЕС-ЦІЛЕЙ

3.1 Ініціалізація проєкту та налаштування середовища розробки

На початковому етапі, було створено проєкт з назвою «Quick Calorie Book», головним пакетом якого є “com.cobaltumapps.quickcaloriebook” у середовищі Android Studio. Також перед роботою проєкту, було впроваджено певну ієрархію пакетів для задоволення архітектури відповідно до рекомендацій Google з App Architecture, тобто створено пакети, які мають в собі файли певного функціоналу згідно до шару: UI, Domain або Data. Структуру проєкту наведено на рисунку 3.1.

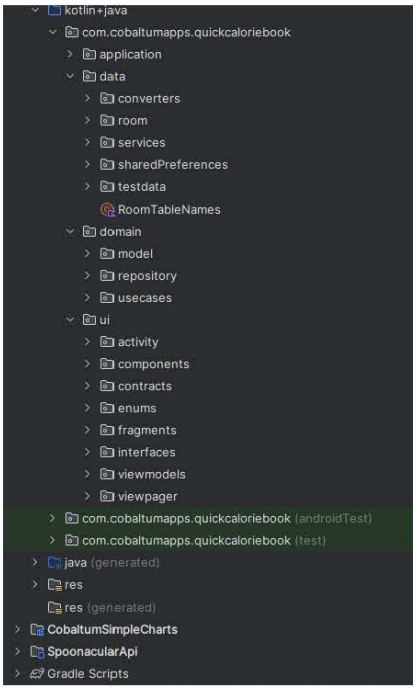

Рисунок 3.1 – Структура проєкту «Quick Calorie Book»

Далі, було проведено налаштування проєкту у файлі build.gradle, який визначає конфігурацію для системи збірки проєкту – Gradle. Проєкт повинен підтримувати мінімальну версію ОС – Android 9.0 (API 26) та новішу, на

момент 20 травня 2025 року – Android 15.0 (API 35). Саме такий діапазон версій дозволяє використовувати більшість зручних функцій при побудові візуальних інтерфейсів та компонентів ОС. Також, за рахунок визначення мінімальної версії системи, не треба піклуватися про можливі помилки на застарілі системи.

Після проведення загальних налаштувань проєкту, було впроваджено кольорову тему, яка була визначення при проєктуванні інтерфейсу. Для цього було використано інструмент Material Theme Builder – офіційний генератор теми від Google [27]. Завдяки цьому інструменту, кольорова палітра була інтегрована у проєкт застосунку не тільки з основним кольором, а й відтінками та іншими кольорами, які необхідні для View–компонентів Android. Загальний вигляд палітри, утворену в результаті налаштувань інструменту і була експортована до проєкту, наведено на рисунку 3.2.

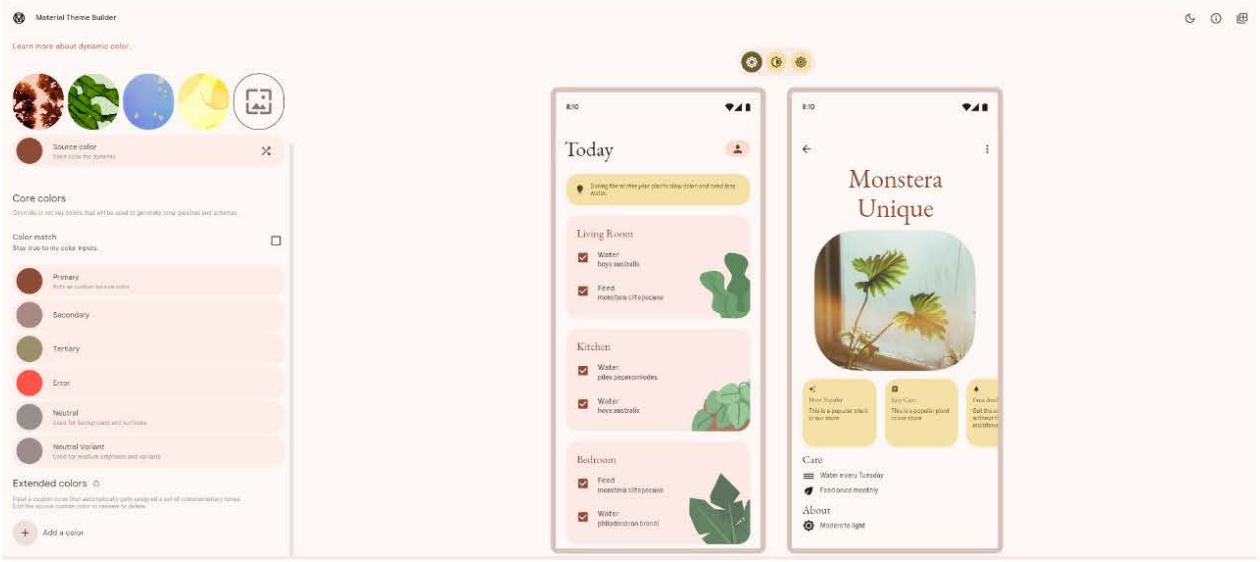

Рисунок 3.2 – Вікно Material Theme Builder з обраною палітрою

Далі, для зручності розробки та забезпечення модульності, деякі компоненти проєкту були винесені в окремі модулі. Такий підхід підвищує гнучкість системи, дозволяє повторно використовувати код, а також спрощує його тестування незалежно від Android-платформи. Зокрема, модулі можна тестувати за допомогою автоматизованих тестів, використовуючи лише мову

програмування Kotlin без необхідності підключення Android SDK. Крім того, окрема збірка модулів дозволяє уникати повної перекомпіляції всього проєкту, що значно економить час під час розробки.

Окрім цього, у проєкті використовується система контролю версій Git, що забезпечує зручне керування змінами в коді, а також платформа GitHub для зберігання репозиторію, спільної роботи та резервного копіювання.

Для узгодженості та зручності в аналізі історії змін застосовується підхід «Conventional Commits» – система найменування комітів за стандартизованою структурою, яка включає тип зміни та короткий опис [28]. Це спрощує читання журналу змін, автоматизацію збірки та генерацію змін до релізів. Для перевірки працездатності застосунку використовувався інструмент Android Virtual Device, що входить до складу Android Studio.

Було створено кілька віртуальних пристроїв із різними версіями ОС, розмірами екранів і конфігураціями пам'яті. Це дало змогу протестувати роботу застосунку в умовах, наближених до реальних, та перевірити адаптивність інтерфейсу і стабільність функцій. Крім того, тестування проводилося і на фізичних пристроях для оцінки поведінки застосунку у реальному середовищі. Характеристики пристроїв наведено в таблиці 3.1.

Таблиця 3.1

Характеристика текстуальних пристроїв

Назва	Версія ОС	Роздільна здатність екрану	Об'єм RAM	Тип
Motorola Moto X	Android 8.0	1280x720	3 GB	Віртуальний
Redmi Note 9 Pro	Android 12	2400x1080	6 GB	Фізичний
Redmi Note 9	Android 13	2340x1080	4 GB	Віртуальний
HTC One	Android 10 Q	1920x1080	6 GB	Віртуальний
Pixel 9	Android 15	1920x1080	8 GB	Віртуальний
Pixel 9 pro	Android 15	2856x1280	16 GB	Фізичний

3.2 Реалізація функціональних можливостей додатку та їх тестування

Застосунок для ведення харчового щоденника реалізує повноцінну систему моніторингу харчування користувача з індивідуальним підходом. Основу архітектури становлять кілька ключових підсистем. Система профілю користувача ініціалізується при першому запуску та зберігає персональні дані: стать, вік, зріст, вага, цільова вага, ціль та активність. Дані використовуються для формування добового плану харчування, який включає цільову кількість калорій та нутрієнтів,, а також розрахунок водного балансу. Архітектурно, профіль реалізовано через шаблон Singleton для централізованого доступу до UserProfileManager, який завантажує профіль з сховища SharedPrefernece та зберігає у собі екземпляр протягом життєвого циклу застосунку.

Система відстеження харчування реалізована з використанням TrackerViewModel та TrackerSessionViewModel у TrackerPageFragment, які відповідають за фіксацію щоденного прогресу, збереження та відтворення сесій при завантаженні застосунку. Також, фрагмент реалізує IntakeNutritionProcessor, який займається розрахунком кількості КБЖВ та надсиланням оновленої інформації через TrackerViewModel.

Для реалізації функціоналу роботи з продуктами харчування, у проєкті було реалізовано активність RationActivtiy, яка координує роботу фрагментів: RationReviewerPageFragment, RationOverviewPageFragment та RationCart-PageFragment. Взаємодія між цими фрагментами забезпечується за допомогою моделей: RationCartViewModel та RationOverviewViewModel. Кожен фрагмент виконує окрему логічну функцію. Ration-ReviewerPageFragment відповідає за пошук та відображення списку доступних продуктів. Користувач може додавати продукти до кошика за допомогою спеціальної кнопки. Додані продукти передаються до RationCartViewModel методом addCartItem(), який, у свою чергу, передає

інформацію до RationCartPageFragment, де відображається вміст кошика, загальна калорійність та сумарна вага обраних продуктів.

Крім того, користувач має можливість переглянути детальну інформацію про обраний продукт, натиснувши на його елемент у списку. Це відкриває фрагмент RationOverviewPageFragment, у якому виводиться повна інформація про продукт: його макронутрієнти, зображення, назва, опис та та можливість налаштувати розмір порції. У цьому фрагменті, продукт також можна додати до раціону або до списку улюбленого.

Для реалізації функціоналу роботи з рецептами у застосунку було створено активність RecipesActivity, яка керує двома фрагментами: RecipesReviewerPageFragment та RecipesOverviewPageFragment. Для обміну даними між цими фрагментами використовується модель RecipesOverview-ViewModel, що дозволяє підтримувати єдиний стан рецепту під час навігації. На відміну від системи звичайних продуктів харчування, у роботі з рецептами не передбачено окремого кошика – додавання відбувається безпосередньо зі сторінки перегляду рецепту.

Користувач може шукати рецепти за допомогою фрагменту RecipesReviewerPageFragment, використовуючи текстовий запит. Після вибору рецепту, відбувається перехід до RecipesOverviewPageFragment, де відображається повна інформація про страву: зображення, назву, опис, час приготування, кількість порцій калорійність, складники та етапи приготування. Також у фрагменті передбачено можливість позначити рецепт як улюблений для швидкого доступу в майбутньому.

Застосунок зберігає дані локально, використовуючи внутрішнє сховище пристрою через Room – бібліотеку для роботи з базами даних у Android. Усі дані, пов’язані з профілем користувача, щоденним раціоном, обраними продуктами та історією вживання, записуються у відповідні таблиці бази даних. Після кожного підтвердження додавання продуктів або рецепту до раціону система оновлює записи: зберігає дату, тип прийому їжі, кількість порцій, калорійність, а також БЖВ (білки, жири, вуглеводи).

Для подальшого аналізу ці дані використовуються у StatisticsActivity. При відкритті цієї активності застосунок звертається до бази даних і витягує історію харчування користувача за останні дні чи тижні. Після обробки даних вони відображаються у вигляді зручних графіків і діаграм: зокрема, користувач може побачити щоденні коливання калорій, співвідношення нутрієнтів, а також загальну динаміку прогресу у досягненні поставленої цілі.

Важливою частиною є система нагадувань NotificationSender, реалізована через WorkManager, яка надсилає push-повідомлення про прийоми їжі, споживання води або порушення плану харчування.

У рамках розширення функціональності застосунку, також були реалізовані аналітичні підсистеми, які забезпечують зручну оцінку прогресу користувача. Зокрема, NutrientComparisonAnalyzer зіставляє споживані нутрієнти з рекомендованими, і у разі суттєвого відхилення ±20% виводить повідомлення на головному екрані. Також реалізовано GoalForecaster, що на основі динаміки змін ваги прогнозує дату досягнення цілі користувача та надає рекомендації.

Контроль відхилень реалізовано у вигляді підсистеми IntakeDeviationMonitor. Якщо користувач протягом трьох днів не досягає 80% плану калорій, система інформує його про це за допомогою push-сповіщення та опису попередження у Analytics. Додатково, щотижневе введення ваги активує AdaptivePlanAdjuster, яка, у разі виявлення невідповідності очікуваному графіку, пропонує оновлений план харчування.

Також для застосунку було реалізовано низку допоміжних фрагментів, що забезпечують користувачу додаткові функції для повноцінного керування своїм профілем, налаштуваннями та аналізом прогресу.

Сторінка Settings реалізує фрагмент SettingsFragment, який надає доступ до базових налаштувань застосунку. Тут користувач може змінити одиниці вимірювання, активувати або вимкнути інтеграцію з Google Fit, налаштувати push-сповіщення.

Сторінка “Profile” реалізує фрагмент ProfileFragment, метою якого є відображення особистих параметри користувача, які були задані під час створення профілю, зокрема вік, стать, зріст, поточну та цільову вагу, рівень фізичної активності та обрану ціль. У разі потреби користувач може переглянути ці дані для повторного аналізу або змінити ці дані для перерахування добового плану.

Для перегляду інформації про водний баланс, було реалізовано фрагмент WaterFragment на сторінці “Water” відповідає за моніторинг водного балансу. У цьому фрагменті відображається щоденна норма споживання води, фактичний обсяг спожитої рідини та індикатор прогресу. Дані можна додавати вручну або автоматично – через інтеграцію з Google Fit.

Сторінка “Analytics” має фрагмент AnalyticsFragment, який містить загальний звіт про прогресію користувача. Користувач може переглянути графіки зміни ваги протягом тижня, аналіз макронутрієнтів за тиждень або місяць, а також відстежувати ефективність виконання фітнес-цілі. Цей фрагмент інтегрує дані з аналітичних підсистем для зручного візуального представлення.

На сторінці “About” представлено фрагмент AboutFragment, який забезпечує відображення загальної інформації про застосунок, його версію та розробника. Також сторінка має адресу електронної пошти для підтримки у разі виникнення проблем.

Після реалізації усіх екранів і функціональних можливостей системи згідно функціональних та нефункціональних вимог, було проведено тестування застосунку на пристроях, які наведені у таблиці 3.1 підрозділу 3.1. Тестування відбувалося поетапно та охоплювало всі основні й допоміжні підсистеми, що були описані вище. На першому етапі було перевірено ключову функціональність: створення профілю, додавання продуктів до раціону, роботу кошика, відстеження споживання КБЖВ, а також збереження та відновлення сесій. Особлива увага приділялася коректності обробки користувацьких даних і взаємодії між фрагментами, зокрема

RationReviewerPageFragment, RationOverviewPageFragment і RationCart-PageFragment.

Усі аналітичні підсистеми було протестовано з урахуванням різних сценаріїв, щоб упевнитися в точності їхньої роботи. Для системи NutrientComparisonAnalyzer було проведено тестування на здатність правильно реагувати на відхилення споживаних КБЖВ, а GoalForecaster – на правильність прогнозування дати досягнення цілі за різних моделей змін ваги. Для IntakeDeviationMonitor і AdaptivePlanAdjuster моделювалися ситуації з порушенням калорійного плану та різким зниженням темпу схуднення, аби переконатися, що система вчасно генерує повідомлення й адаптує плани.

Функціональність push-сповіщень та нагадувань перевірялась як під час активного використання застосунку, так і в фоновому режимі, аби гарантувати надійне надсилання повідомлень користувачу. Окремо тестувався візуальний інтерфейс – на відповідність принципам UX-дизайну, легкість навігації між основними сторінками: Book, Ration, Recipes та зручність користування допоміжними фрагментами, як Settings, Profile, Water, Analytics та Help.

Загалом застосунок пройшов повний цикл модульного й інтеграційного тестування, що дозволило досягти високої стабільності в роботі навіть за інтенсивного використання. Особливу увагу було приділено точності розрахунків, рекомендацій та коректному збереженню і контролю всіх користувацьких дій упродовж дня.

3.3 Інтеграція з Spoonacular API для пошуку та аналізу продуктів

Однією з основних функцій харчового щоденника є зручний пошук продуктів та перегляд їх харчової цінності. Для реалізації цієї можливості в Android-застосунку було інтегровано Spoonacular API – сторонній сервіс, що надає широкий набір даних про продукти, інгредієнти, рецепти та страви.

Щоб реалізувати запити до API, було створено інтерфейс SpoonacularApiService з використанням бібліотеки Retrofit. Кожен метод інтерфейсу відповідає відповідній кінцевій точці API, а параметри передаються через анотації @Query або @Path. Усі запити обов'язково містять API-ключ для авторизації на сервісі Spoonacular. Основними запитами, які використовуються у застосунку, є:

– пошук інгредієнтів і продуктів за ключовим словом;

– отримання повної інформації про інгредієнт або продукт за його ідентифікатором;

– отримання декількох випадкових рецептів;

– пошук рецептів за запитом та дієтичними фільтрами користувача.

Процес пошуку інгредієнтів чи продуктів побудований у два етапи. Спочатку виконується запит searchIngredients() або searchProducts(), що повертає список об'єктів з їхніми ID. Далі за кожним ID надсилається запит getIngredientInformation() або getProductInformation() для отримання повної інформації про кожен елемент.

Для роботи з рецептами передбачено два методи: getRecipeRandom() – для випадкових рецептів, та complexSearchRecipe() – для пошуку за текстовим запитом. Вони також приймають параметр number (кількість результатів) та apiKey, а метод complexSearchRecipe() додатково підтримує параметр offset для посторінкового завантаження.

Щоб зменшити кількість запитів до Spoonacular API та підвищити швидкість роботи застосунку, було впроваджено локальне кешування результатів пошуку. Після кожного пошуку за допомогою searchIngredients() або searchProducts(), застосунок зберігає у базу даних запит користувача та список отриманих ID. Якщо той самий запит повторюється, ID завантажуються з кешу, без повторного звернення до API. Це дозволяє швидко перейти до наступного етапу – отримання детальної інформації.

Додатково кешуються й самі об'єкти відповіді з сервера. Коли користувач відкриває деталі інгредієнта або продукту, отриманий JSON

об'єкт серіалізується та зберігається у базі даних разом із відповідним ID. При повторному зверненні застосунок спочатку перевіряє наявність об’єкта в кеші – і, якщо він знайдений, відображає дані без доступу до мережі.

Цей підхід значно підвищує швидкодію, зменшує використання API-квоти та дає можливість працювати навіть у офлайн-режимі. Користувач може переглядати раніше знайдену інформацію без доступу до інтернету, що особливо зручно у дорозі або за нестабільного з’єднання. Таким чином, локальне кешування стало ключовим елементом застосунку, який поєднує продуктивність, стабільність і зручність для користувача. Процес кешування та отримання даних з кешу наведено на рисунку 3.2.

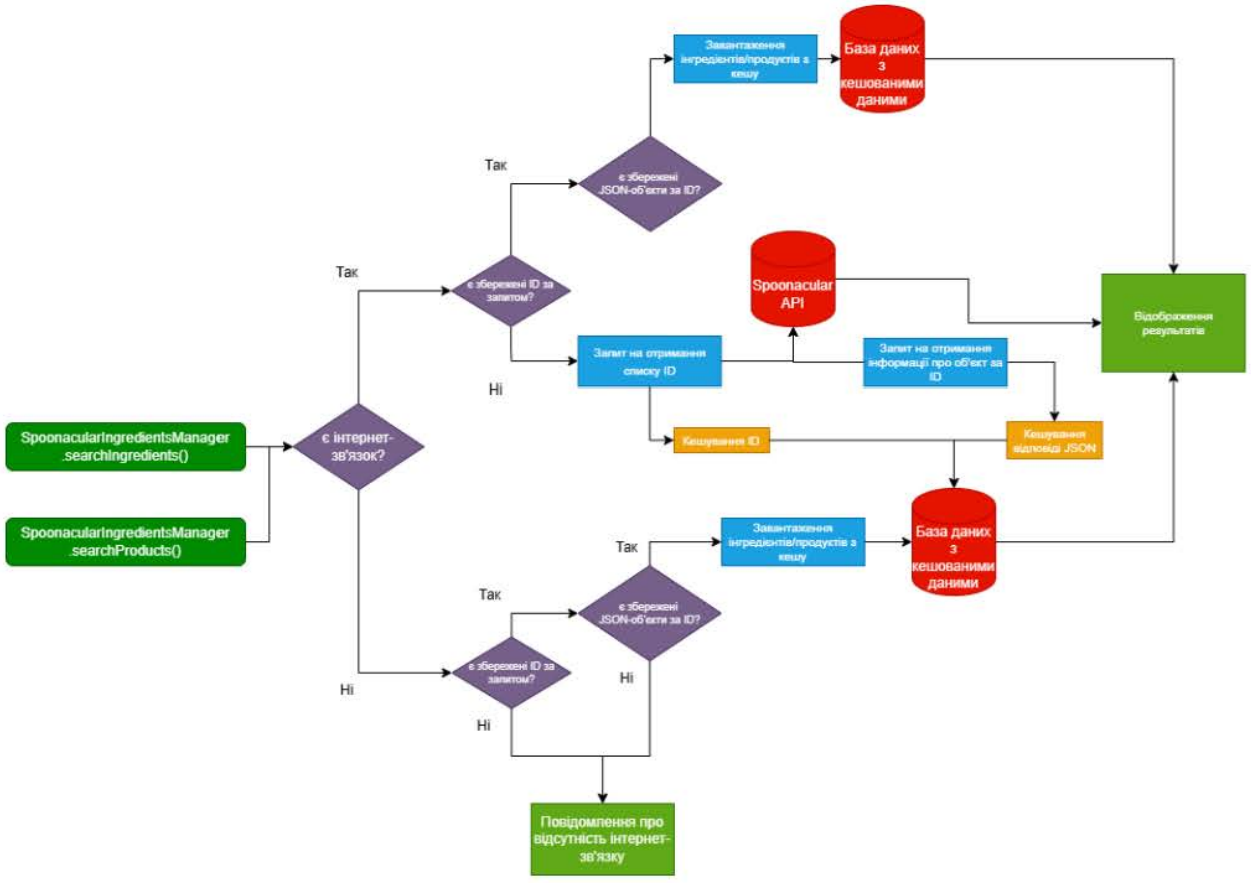

Рисунок 3.2 – Діаграма потоків процесу кешування даних

Процес пошуку рецептів у застосунку відбувається аналогічно до інгредієнтів та продуктів. Спочатку виконується запит до кінцевої точки “recipes/complexSearch”, яка повертає перелік рецептів із базовою

інформацією та унікальними ідентифікаторами. Далі, використовуючи ці ID, надсилаються окремі запити до API для отримання повної інформації про кожен рецепт – включно з описом, списком інгредієнтів, інструкціями з приготування та, за потреби, харчовими характеристиками. У запиті до complexSearch передаються такі параметри: API-ключ, текстовий запит користувача, кількість потрібних результатів та параметр offset для реалізації посторінкового завантаження. Механізм кешування, реалізований для інгредієнтів і продуктів, також застосовується і для рецептів, що дозволяє зберігати результати пошуку та працювати з ними навіть у режимі офлайн. Процес роботи з кешованими даними представлено на рисунку 3.3.

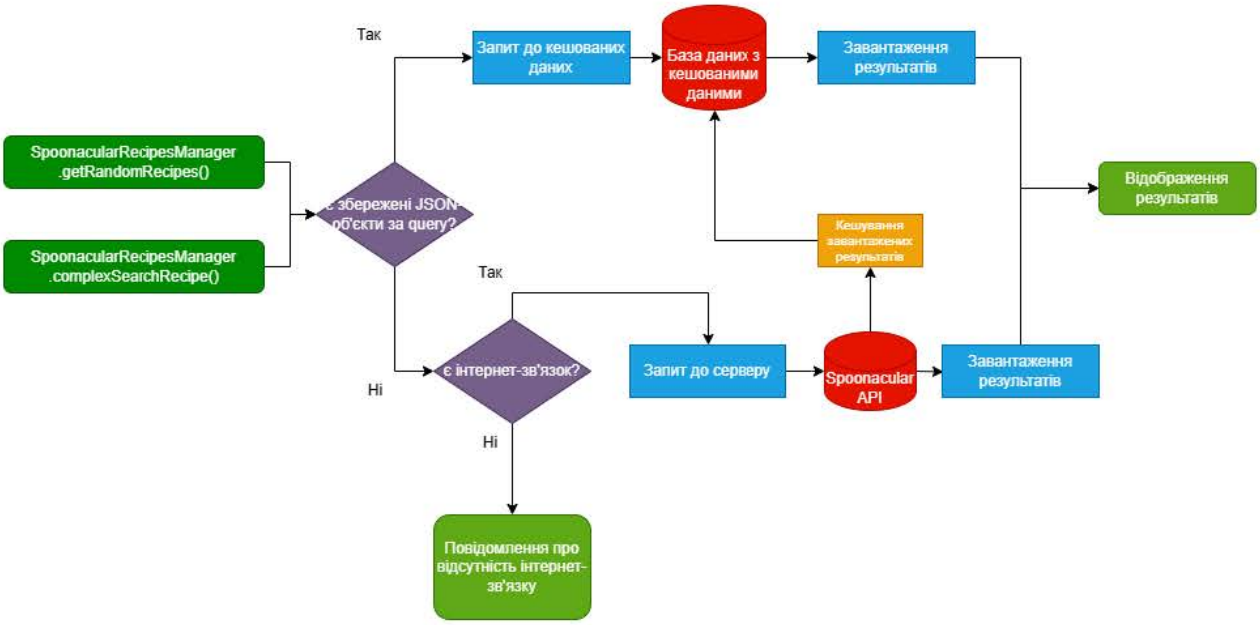

Рисунок 3.3 – Діаграма потоків процесу кешування даних

3.4 Інструкція для користувача

Для початку роботи з застосунком, користувач повинен створити профіль. Для цього, йому потрібно ввести свої характеристики до системи через паралельне ознайомлення з додатком у вступному модулі. Застосунок запрошує такі характеристики: стать, вік, поточна вага, цільова вага, зріст, фітнес-ціль та фізична активність. Після введення усіх даних, застосунок

відображає заповнений профіль користувача, і якщо дані правильно заповнено та користувач задоволений - він погодиться на збереження профілю. Після чого, користувач може повноцінно користуватися застосунком.

У користувача є три основні сторінки з якими він має працює: Book, Ration та Recipes.

На сторінці Book - користувач може продивлятися, як проходить його прогрес харчування: КБЖВ на добу, кількість калорій, які залишилися, спалені калорії та інше. Виходячи з результатів сторінки Book, користувач повинен планувати свій раціон на день, виходячи з своїх поточних результатів.

Для додавання нових продуктів до свого раціону, користувачу потрібно перейти на сторінку Ration, адже на цій сторінці користувач може: продивлятися вживані калорії на кожен прийом їжі, обирати улюблені продукти, якщо є та додавати нові продукти через кнопку “Add”. Якщо користувач натиснув кнопку - він переходить до сторінки з пошуком, за допомогою якої можна знайти потрібний продукти харчування. Знайшовши потрібні продукти, користувач може одразу швидко додати його до кошику натиснувши відповідну кнопку поряд з знайденим продуктом або натиснути На сам продукт і перейти до його сторінки огляду, же вже можна продивитися всю інформацію в подробицях на налаштувати порцію. Після додавання їжі до раціону, користувач має перейти до кошику, в якому зберігається всі обрані продукти. На цій сторінці, користувач може продивлятися сумарну кількість калорій, порцій а також побачити прогрес, скільки буде додано до раціону, і як зміниться кількість калорій. У разі, якщо користувача все задовольняє - він має натиснути кнопку “Confirm” Для підтвердження списку продуктів, після чого, система їх визначає як вживані і долає до раціону. Користувач одразу бачить свій прогрес, так як після підтвердження - застосунок переходить на сторінку Book.

Окрім додавання продуктів, застосунок дозволяє передивлятися список рецептів та додавати їх також до раціону за допомогою сторінки Recipes. На цій сторінці, користувач може бачити список рецептів, що сподобалися. Для перегляду списку рецептів, користувач повинен натиснути кнопку “Overview”, після чого він перейде до сторінки пошуку рецептів. На цій сторінці, спочатку відображаються випадковий список рецептів, але користувач може натиснути на поле пошуку, ввести запит і натиснути кнопку “Search”: система виведе новий список з рецептами, які підходять під запит.

Далі, для вибору рецепту, користувачеві потрібно натиснути на картку з рецептом, після чого він перейде до сторінки з інформацією про рецепт. На цій сторінці відображаються усі дані про рецепт, його опис, тривалість готування, список інгредієнтів і інструментів та етапи .приготування.

Для додавання рецепту до раціону, користувачеві треба обрати кількість порцій і натиснути кнопку “Add”, після чого, рецепт буде додано до раціону. Крім того, користувач також може додати рецепт до улюбленого для швидкого доступу.

Окрім основних сторінок, користувачу також доступні допоміжні сторінки застосунку. Для їх доступу, користувачу треба лише відкрити меню навігації, і обрати сторінку:

– Settings – містить налаштування, за допомогою яких користувач може налаштовувати одиниці вимірювання, інтеграцію з Google Fit та інше;

– Profile – містить дані профілю, який користувач створив напочатку;

– Water – містить інформацію про водний баланс користувача;

– Analytics – містить аналітику прогресу;

– About – містить дані розрахунків та загальний опис як застосунок оброблює дані користувача.

Для перегляду інформації про водний баланс, користувач повинен натиснути на шкалу води на сторінці Tracker, після чого, система перейде до допоміжної сторінки Water або власноруч обрати у допоміжному меню відповідний пункт.

Для перевірки своєї успішності у досягнені поставленої цілі – користувач повинен перейди до допоміжної сторінки Analytics, де можна продивитися інформацію щодо наявності відхилень, прогнозування приблизної дати, виконання поставленої цілі.

Для перевірки свого профілю, користувач повинен зайти до допоміжної сторінки Profile, де будуть відображатися усі параметри поточного профілю.

Також користувач може перейди до сторінки “About” у разі, якщо треба доступ до пошти підтримки.

3.5 Висновки до третього розділу

У підрозділі 3.1 було розглянуто етапи ініціалізації проєкту, який реалізовує системи відповідно до функціональних вимог, а також структуру проєкту, яка відповідає принципам App Architecture, де проєкт розбивається на логічні шари. Також було визначено набір віртуальних та фізичних пристроїв, на яких є потреба протестувати застосунок на наявність проблем з продуктивністю та виявити сильні та слабкі сторони застосунку.

У підрозділі 3.2 було реалізовано основні функції застосунку, що забезпечують повноцінне ведення харчового щоденника. Було впроваджено системи пошуку, перегляду та додавання продуктів, а також обліку прийомів їжі. Реалізовано можливість формування раціону з обраних страв, додавання рецептів та автоматичний підрахунок спожитих калорій і нутрієнтів. Усі дані зберігаються локально та відображаються в інтерфейсі користувача через фрагменти, інтегровані в основні активності. Також у рамках цього підрозділу було реалізовано низку аналітичних модулів, покликаних підвищити ефективність користування застосунком. Зокрема, модуль порівняння фактичного споживання нутрієнтів, модуль аналізу динаміки змін ваги користувача та прогнозування дати досягнення встановленої цілі, водночас формуючи рекомендації. Також окрім звичайних аналітичних модулів, було реалізовано модулі виявлення тривалих порушень в

споживанні калорій, який надсилає відповідні попередження, а також адаптер, який запитує поточну вагу користувача. У разі змін ваги – модуль виконує перерахування добового плану під зміни ваги та надає користувачеві новий план харчування. Таким чином, аналітична складова забезпечує не лише контроль, а й підтримку користувача та адаптацію під поточні результати для ефективного досягнення фітнес-цілі.

Окрім функціональної реалізації, у цьому ж підрозділі описано тестування застосунку: перевірялися всі ключові сценарії, включно з роботою аналітичних модулів, сповіщень, а також стабільністю систем збереження й відновлення даних. Тестування охоплювало ручні й автоматизовані методи, що дозволило гарантувати надійну й точну роботу навіть за інтенсивного використання.

У підрозділі 3.3 було розглянуто реалізацію API-сервісу Spoonacular у застосунку, що забезпечує можливість розширеного пошуку продуктів, інгредієнтів, продуктів та рецептів з отриманням повної харчової інформації. Було реалізовано відповідні інтерфейси для взаємодії з API через бібліотеку Retrofit, що дозволяє надсилати запити за ключовими словами, дієтичними фільтрами або ID елементів. З метою оптимізації роботи застосунку впроваджено систему локального кешування результатів запитів, яка підвищує швидкодію, зменшує кількість звернень до серверу і дозволяє використовувати отримані дані в офлайн-режимі.

У підрозділі 3.4 було представлено інструкцію для користувача, яка містить пояснення основних дій у застосунку: створення профілю, відстеження щоденного раціону, перегляд статистики. Інструкція охоплює інтерфейс головної сторінки, навігацію між вкладками, взаємодію з фрагментами раціону, додавання рецептів і продуктів, а також використання графічних звітів. Окрему увагу приділено поясненню сповіщень, налаштуванням цілей і користуванню аналітичними модулями, що робить застосунок доступним і зрозумілим навіть для нових користувачів.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра, було реалізовано застосунок «Quick Calorie Book», який призначений для можливості вести харчовий щоденник, контролю споживання продуктів харчування і водного балансу та відстеження прогресу у досягненні поставленої фітнес-цілі.

Протягом розробки мобільного застосунку «Quick Calorie Book» було виконано комплекс теоретичних і практичних завдань, спрямованих на реалізацію ефективного інструменту для ведення харчового щоденника та аналізу впливу раціону на фітнес-результати.

На першому етапі було здійснено аналіз літературних джерел, наукових статей та актуальних досліджень предметної області на тему харчового трекінгу. Це дало змогу сформувати теоретичне підґрунтя для розробки застосунку, враховуючи сучасні тенденції та підходи до контролю харчування й здоров'я.

Далі був проведений аналіз існуючих Android-застосунків, які виконують подібні функції. Аналіз їхньої функціональності, інтерфейсів, переваг і обмежень дозволив виділити ключові аспекти, які потребують покращення, та визначити конкурентні переваги майбутнього застосунку.

На основі аналізу ринку та потреб цільової аудиторії було сформульовано функціональні та нефункціональні вимоги до системи. Зокрема, враховано очікування користувачів щодо персоналізації, зручності введення даних, наочності статистики та мотиваційних елементів.

Після формулювання вимог було розроблено технічне завдання, яке включало чіткий перелік необхідних функцій, логіку взаємодії модулів, структуру даних та основні сценарії використання. Це завдання слугувало основою для архітектурного проєктування та реалізації програмного продукту.

Особливу увагу приділено обґрунтуванню вибору платформи Android, мови програмування Kotlin, а також середовища розробки Android Studio,

бібліотек Room, Gson, Retrofit та інтеграції зі стороннім API Spoonacular для роботи з харчовими продуктами й рецептами. Такий підхід забезпечив масштабованість та простоту подальшого розширення функціоналу.

У рамках архітектурного проєктування, було створено модульну структуру застосунку з розділенням на логічні компоненти – менеджери даних, ViewModel, локальну БД, репозиторії та. Також реалізовано механізми для збереження та обробки даних про раціон, прогрес і мету користувача.

На основі сформованих вимог було спроектовано інтерфейс користувача, який відповідає сучасним вимогам UI/UX-дизайну. Особливу увагу приділено зручності взаємодії, наочності візуалізації прогресу та доступності функцій як для початківців, так і для досвідчених користувачів.

Під час розробки реалізовано основні функціональні модулі: створення профілю, індивідуальний розрахунок КБЖВ, щоденне відстеження харчування, прогнозування результатів, аналітику, нагадування, пошук продуктів і керування улюбленими стравами.

Також впроваджено модуль аналітики, який дає змогу користувачу бачити, як динаміка його харчування впливає на досягнення поставленої мети. Це дозволяє робити висновки щодо ефективності поточного плану та своєчасно коригувати раціон.

На завершальному етапі було проведено тестування програмного забезпечення – перевірено стабільність, швидкодію, коректність обробки даних та зручність взаємодії з користувачем. За результатами тестування сформовано рекомендації щодо майбутнього розвитку й вдосконалення функціоналу.

В якості переваг, розроблене програмне має:

- фокус на досягнення фітнес-цілей, а не лише підрахунок калорій;
- автоматичне коригування добового плану на основі змін ваги;
- аналіз прогресу і прогнозування досягнення мети на його основі;
- зручна і проста візуалізація щоденного прогресу;

– налаштована система сповіщень про прийоми їжі або у разі відхилень від плану;

– можливість обліку спалених калорій або води без сторонніх фітнес-трекерів;

– фільтрація рецептів за дієтичними обмеженнями;

– можливість повноцінної роботи без доступу до інтернету за рахунок кешування даних.

Але окрім переваг, які надає виконаний застосунок, він має ряд недоліків, які можуть деяким користувачам завдавати незручності у використанні додатком:

– обмежена база продуктів у порівнянні з глобальними сервісами;

– відсутність соціальної складової, наприклад: груп, досягнень, персональних завдань, спільнот та іншого;

– недостатньо глибока персоналізація для складних дієт;

– потреба у подальшому вдосконаленні аналітичного модуля;

– відсутність локалізації пропрієтарної бази даних продуктів харчування.

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Укрінформ. URL: https://www.ukrinform.ua/rubric-health/3791712-v-ukraini-blizko-60-ludej-maut-zajvu-masu-tila.html (дата звернення 28.04.2025).

2. Novo nordisk. URL: https://www.novonordisk.ua/content/dam/nncorp/ua/uk/media/obesity-press-release-Nov2021-ukraine.pdf (дата звернення 28.04.2025).

3. Forbes. URL: https://forbes.ua/lifestyle/ukraina-mozhe-vtratiti-ponad-5-vvp-cherez-ozhirinnya-yak-khvoroba ruynue-nashu-ekonomiku-ta-shcho-z-tsim-robiti-09102023-16325 (дата звернення 28.04.2025).

4. Khurtenko, O., Dmitrenko, S., Sorokina, N., & Lyshyshyn, G. (2021). Using mobile applications as one of the means of a healthy lifestyle. с. 138. URL: https://spppc.com.ua/index.php/journal/article/download/374/363/ (дата звернення 28.04.2025).

5. PubMed Central. URL: https://pubmed.ncbi.nlm.nih.gov/31353783/ (дата звернення 28.04.2025).

6. PubMed Central: https://pmc.ncbi.nlm.nih.gov/articles/PMC10034244/ (дата звернення 28.04.2025).

7. CIKAVOSTI. URL: https://cikavosti.com/mobilni-dodatki-dopomagayut-shudnuti-krashhe-nizh-diyeti (дата звернення 28.04.2025).

8. PubMed Central: https://pubmed.ncbi.nlm.nih.gov/38745944/ (дата звернення 28.04.2025).

9. BMC Obesity. URL: https://bmcobes.biomedcentral.com/articles/10.1186/s40608-014-0022-4 (Дата звернення 28.04.2025).

10. Мобільний застосунок EatFit. URL: https://play.google.com/store/apps/details?id=com.mymealplanner.grechakura (дата звернення 29.04.2025).

11. Застосунок YAZIO. URL: https://play.google.com/store/apps/details?id=com.yazio.android (дата звернення 29.04.2025).

12. Застосунок Lifesum. URL: https://play.google.com/store/apps/details?id=com.sillens.shapeupclub (дата звернення 29.04.2025).

13. Застосунок Tracker – calorie counter. URL: https://play.google.com/store/apps/details?id=com.nutritionix.nixtrack (дата звернення 30.04.2025).

14. Застосунок Nutrilio. URL: https://play.google.com/store/apps/details?id=net.nutrilio (дата звернення 31.04.2025).

15. Android Developers. Guide to app Architecture. URL: https://developer.android.com/topic/architecture#recommended-app-arch (дата звернення 31.04.2025).

16. Jin B., Sahni S., Shevat A. Designing Web APIs: Building APIs That Developers Love. O'Reilly Media. (2018). 217 p.

17. Android Studio. URL: https://developer.android.com/studio (дата звернення 01.05.2025).

18. Android Developers. Windows: Verify system requirements. URL: https://developer.android.com/codelabs/basic-android-kotlin-compose-install-android-studio#1 (дата звернення 01.05.2025).

19. Kotlin Programming Language. URL: https://kotlinlang.org/ (дата звернення 01.05.2025).

20. Null safety | Kotlin Documentation. URL: https://kotlinlang.org/docs/null-safety.html (дата звернення 03.05.2025).

21. Coroutines | Kotlin Documentation. URL: https://kotlinlang.org/docs/coroutines-overview.html (дата звернення 04.05.2025).

22. Android Developers. Save data in a local database using room. URL: https://developer.android.com/training/data-storage/room (дата звернення 05.05.2025).

23. Gson. URL: https://github.com/google/gson (дата звернення 07.05.2025).

24. Square. Retrofit. URL: https://square.github.io/retrofit/ (дата звернення 07.05.2025).

25. Square. OkHttp. URL: https://square.github.io/okhttp/ (дата звернення 07.05.2025).

26. Material Design 2 Homepage. URL: https://m2.material.io/ (дата звернення 08.05.2025).

27. Material Design 3 Home. URL: https://m3.material.io/ (дата звернення 08.05.2025).

28. Material Theme Builder. URL: https://material-foundation.github.io/material-theme-builder/ (дата звернення 18.05.2025).

29. Conventional Commits. URL: https://www.conventionalcommits.org/en/v1.0.0/ (дата звернення 16.05.2025).

30. Spoonacular Docs. URL: https://spoonacular.com/food-api/docs (дата звернення 26.05.2025).

ДОДАТКИ

Таблиця А.1

Функціональні вимоги до персоналізації та планування харчування

Назва вимоги	Опис вимоги
Створення персонального профілю користувача	система повинна запрошувати дані користувача та формулювати профіль базуючись на цих даних
Розрахунок плану на основі профілю користувача	система повинна з великою точністю розраховувати план для користувача
Відображення необхідної кількості нутрієнтів на добу	система повинна вчасно і точно показувати необхідну кількість нутрієнтів і води на добу
Відображення добового прогресу у графічних елемента.	система повинна коректно відображати прогрес вживаних нутрієнтів і води на добу відносно плану
Автоматичне збереження щоденної статистики з моменту першого використання застосунку	система повинна автоматично зберігати та правильно відображати статистичні дані щодо харчування з моменту першого використання
Автоматична корекція плану на основі змін ваги	система повинна оновлювати план відповідно до змін у вазі користувача
Порівняння спожитих нутрієнтів з добовим планом	система повинна аналізувати фактичне споживання нутрієнтів і виявляти відхилення
Контроль критичних відхилень	система повинна повідомляти про критичні відхилення, які тривають кілька днів поспіль

Таблиця А.2

Функціональні вимоги до статистики застосунку, аналітики та нагадування

Назва вимоги	Опис вимоги
Автоматичне збереження щоденної статистики	система повинна зберігати всі дані з моменту першого запуску
Відображення статистики за певний день, обраний користувачем	користувач має можливість переглянути статистику використання застосунку за будь-який обраний день
Прогнозування досягнення мети	система повинна оцінювати прогрес та розраховувати приблизний прогноз щодо виконання фітнес-цілі користувача
Введення спалених калорій	користувач може вводити кількість спалених калорій для врахування цих даних системою
Вчасне нагадування про прийом їжі та води	система повинна надсилати push-нагадування в час, коли розпочинається прийом їжі або води

Таблиця А.3

Функціональні вимоги до організації керування продуктами та рецептами

Назва вимоги	Опис вимоги
Пошук, огляд та вибір доступної їжі	система повинна надати зручний інтерфейс для пошуку, огляду та вибору продуктів
Пошук, огляд та вибір рецептів страв	система повинна дозволяти знайти рецепти за запитом, відображати всю доступну інформацію про рецепт та мати можливість його додавання до раціону
Пошук рецептів за дієтичними фільтрами	користувач повинен мати змогу фільтрувати рецепти за типом дієти.
Додавання, видалення та перегляд продуктів у списку улюбленого	користувач може позначати їжу, як улюблену, що надає швидкий доступ та зручність

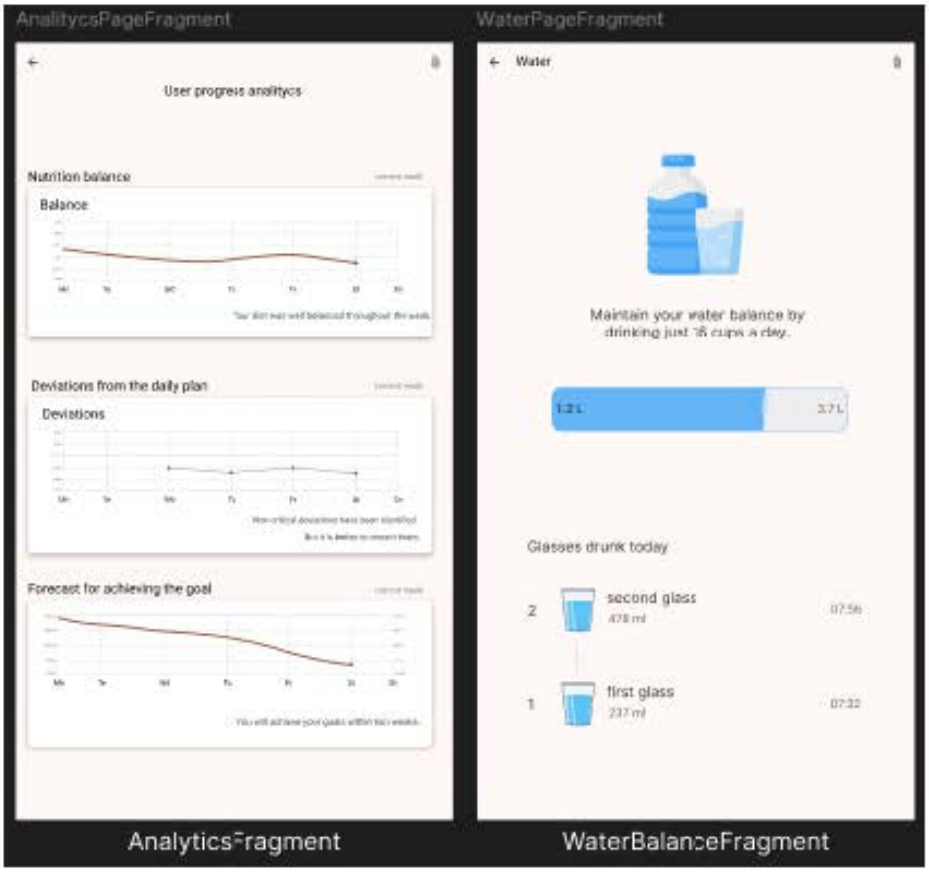

Рисунок А.1 – Інтерфейс сторінок Analytics та Water

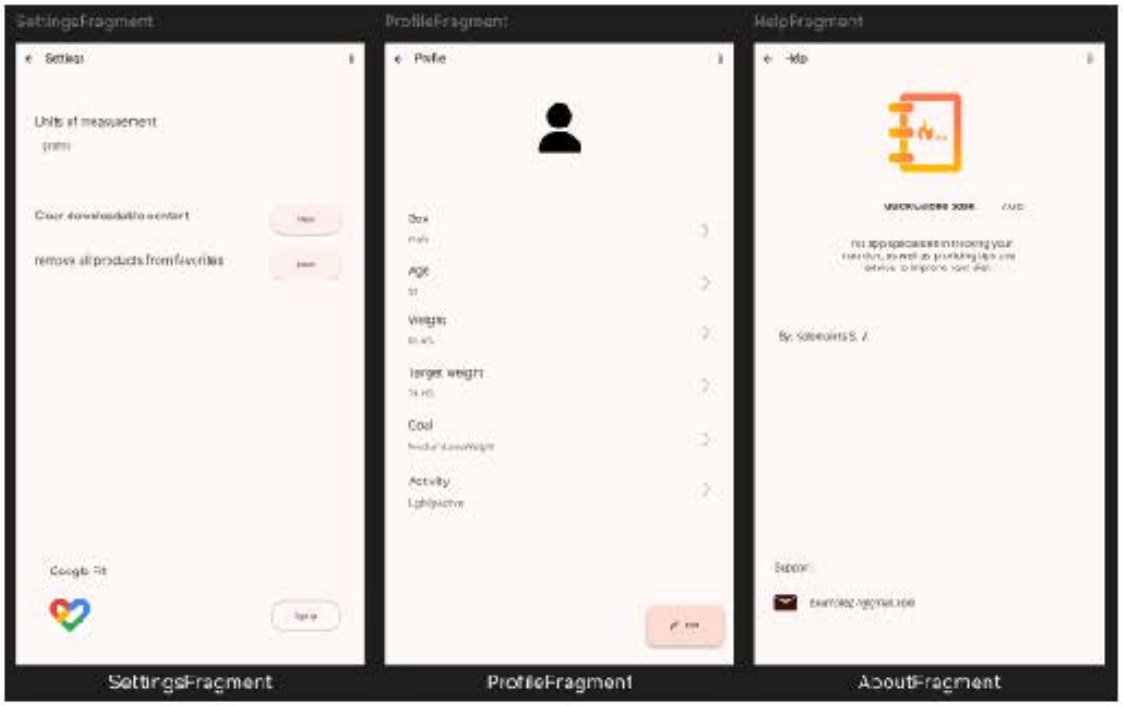

Рисунок А.2 – Інтерфейс сторінок Settings, Profile та About

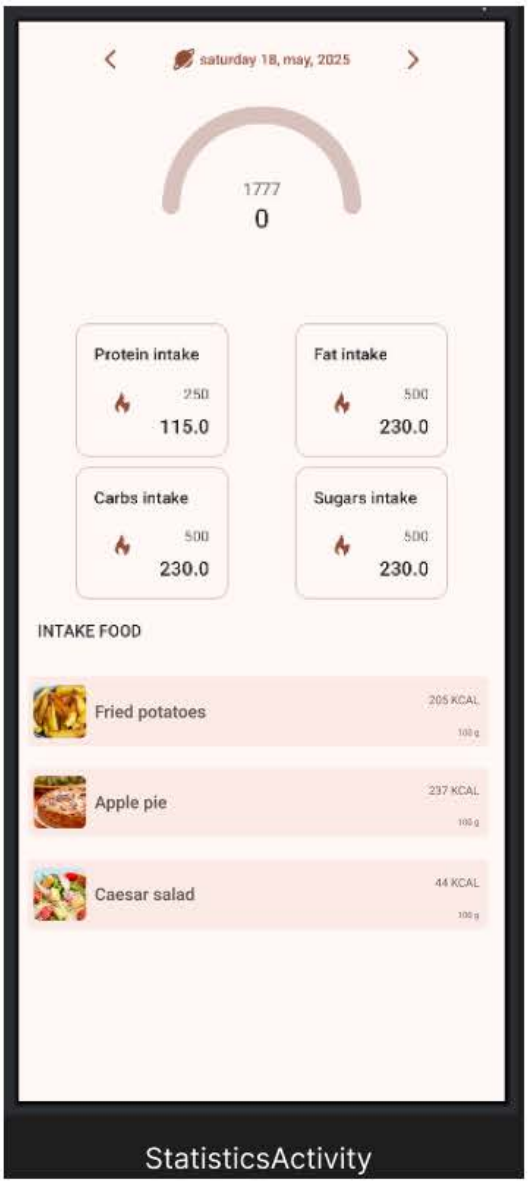

Рисунок А.3 – Інтерфейс сторінки Statistics