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

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ANDROID-ДОДАТКУ ДЛЯ КОНТРОЛЮ СПОЖИВАННЯ СПОРТИВНОГО ХАРЧУВАННЯ З ВИКОРИСТАННЯМ KOTLIN ТА JETPACK COMPOSE

DEVELOPMENT OF AN ANDROID APPLICATION FOR TRACKING SPORTS NUTRITION CONSUMPTION USING KOTLIN AND JETPACK COMPOSEE

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

Виконав: здобувач вищої освіти
групи ІПЗс-21
Сищук Владислав Геннадійович

Керівник:
Потейчук Михайло Іванович

Кваліфікаційну роботу
допущено до захисту
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
10.06.2025 р.

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ

Завідувач кафедри

«20» 12 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Сищуку Владиславу Геннадійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка Android-додатку для контролю споживання спортивного харчування з використанням Kotlin та Jetpack Compose»*
Керівник роботи: *асистент Потейчук М. І.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р. № 474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжніта вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз Android-додатку для контролю споживання спортивного харчування і постановка завдань на кваліфікаційну роботу Специфікація вимог до розроблюваної системи Розробка android-додатку для контролю споживання спортивного харчування з використанням Kotlin та JetpackCompose.*
5. Перелік графічного матеріалу: *10 рисунків, 1 таблиця.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Потейчук М. І.*		
Специфікація вимог до розробленої системи	*Потейчук М. І..*		
Розробка об'єкта проектування	*Потейчук М. І.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	3,98 %		
Академічна доброчесність	*Потейчук М. І.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел з теми розробки соціальної мережі	до 04.02.2025 р.	виконано
2	Аналіз проблематики створення соціальної мережі для обміну рецептами	до 01.03.2025 р.	виконано
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	виконано
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 29.03.2025 р.	виконано
5	Практична реалізація об'єкта проектування та розробка бази даних	до 26.04.2025 р.	виконано
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	виконано
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	виконано

Здобувач вищої освіти _____ Владислав СИЩУК

Керівник кваліфікаційної роботи _____ Михайло ПОТЕЙЧУК

АНОТАЦІЯ

Сищук В. Г. Розробка Android-додатку для контролю споживання спортивного харчування з використанням Kotlin та Jetpack Compose. Рукопис. Кваліфікаційна робота бакалавра «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел.

Перший розділ присвячено аналізу сучасного стану проблеми та постановлення завдання на кваліфікаційну роботу.

В другому розділі здійснено вибір та обґрунтування засобів розробки.

Третій розділ присвячено практичній реалізації проєкта, його тестуванню, створення APK файлу.

Ключові слова: Android Studio, Kotlin, мобільний додаток, Android-додатків, Jetpack Compose, ViewModel, Compose.

ABSTRACT

Syshchuk V. Development of an Android Application for Tracking Sports Nutrition Consumption Using Kotlin and Jetpack Compose. Manuscript. Bachelor's qualification work "Software Engineering" specialty "Software Engineering". Lutsk National Technical University. Lutsk, 2025.

The qualification work consists of an introduction, three sections, conclusions, a list of sources used, an appendix.

The first section is devoted to the analysis of the problems of the current state and the formulation of the task for the qualification work.

The second section selects and justifies the development tools.

The third section is devoted to the practical implementation of the project, its testing, and the creation of an APK file.

Keywords: Android Studio, Kotlin, mobile application, Android applications, Jetpack Compose, ViewModel, Compose.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі здоровий спосіб життя та активні заняття спортом стали невід'ємною частиною життя багатьох людей. Зі зростанням інтересу до фітнесу та спортивного харчування зросла також потреба в зручних цифрових інструментах, що дозволяють ефективно контролювати споживання харчових добавок, вітамінів та спортивного харчування.

Метою роботи є розробка Android-додатку для контролю споживання спортивного харчування із використанням сучасних технологій Kotlin та Jetpack Compose, що дозволить користувачам вести облік прийому добавок, контролювати дозування та дотримуватися режиму харчування.

Об'єкт роботи – процес розробки мобільного програмного забезпечення для платформи Android.

Предметом роботи – засоби та технології створення інтерфейсу користувача й логіки роботи Android-додатку на базі мови Kotlin та фреймворку Jetpack Compose для контролю споживання спортивного харчування.

Для досягнення мети кваліфікаційної роботи бакалавра, поставлені наступні завдання:

– провести аналіз предметної області та визначити основні вимоги до функціональності мобільного додатку для контролю споживання спортивного харчування;

– ознайомитися з сучасними підходами, технологіями та інструментами розробки Android-додатків, зокрема Kotlin та Jetpack Compose;

– проаналізувати та визначити вимоги до розроблюваного програмного забезпечення та спроектувати архітектуру додатку, включаючи структуру даних, логіку роботи, навігацію та інтерфейс користувача;

– розробити Android-додаток з використанням Kotlin та Jetpack Compose;

– провести тестування додатку на коректність роботи та відповідність вимогам.

РОЗДІЛ 1
АНАЛІЗ ANDROID-ДОДАТКІВ ДЛЯ КОНТРОЛЮ СПОЖИВАННЯ СПОРТИВНОГО ХАРЧУВАННЯ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У наш час здоровий спосіб життя, регулярна фізична активність та збалансоване харчування є невід'ємними частинами життя багатьох людей. Водночас спортивне харчування набирає популярності, його використовують професійні та аматорські спортсмени для покращення своїх результатів, підтримки фізичної форми та швидшого відновлення. Існує потреба в зручному інструменті для моніторингу споживання спортивного харчування, оскільки неправильне використання може не тільки знизити ефективність тренувань, але й зашкодити здоров'ю. На ринку мобільних додатків вже існують аналогічні програми, які частково або повністю вирішують цю проблему. Однак більшість із них мають як переваги, так і суттєві недоліки, що створює передумови для розробки нової, більш зручної та функціональної системи.

Історія розвитку програм моніторингу харчування тісно пов'язана з поширенням смартфонів, доступом до інтернету та глобальними змінами в харчовій культурі. На ранніх етапах розвитку ці програми пропонували переважно такі функції, як простий підрахунок калорій, ручне введення даних про харчування та відображення результатів у спрощеному тексті. Цим додаткам бракувало гнучкості, вони не враховували специфіку спортивного харчування та не дозволяли планувати раціон на основі виду діяльності чи особистих спортивних цілей користувача. З часом додатки почали інтегрувати функції спортивного харчування, можливість введення інформації про харчові продукти у вигляді простих форм, а потім інтегрували зчитувачі штрих-кодів для швидкого введення даних. З'явилися перші системи рекомендацій для оптимізації харчування на основі рівня активності користувача, виду спорту та особистих характеристик.

Тому існує очевидна потреба в спеціалізованому програмному забезпеченні для платформи Android, призначеному для контролю спортивного харчування та інтеграції кількох важливих функцій. Це програмне забезпечення повинно мати простий та інтуїтивно зрозумілий інтерфейс для швидкого введення інформації про харчування, дозволяти розробку персоналізованих планів харчування на основі спортивних цілей та бути придатним для використання без доступу до Інтернету. Також важливо використовувати відкриті та оновлювані бази даних продуктів, що забезпечують чітку перевірку введених даних для отримання достовірної картини харчових звичок.

Необхідним є забезпечення гнучкого функціоналу для взаємодії із зовнішніми пристроями – фітнес-трекерами, датчиками активності, смарт-годинниками – для отримання максимально повної інформації про активність користувача, а також можливість формування комплексних рекомендацій для оптимізації харчування залежно від отриманих даних. Інтеграція із такими пристроями відкриває можливість побудови універсальної платформи для підтримки спортсменів, сприяючи формуванню усвідомленого й гнучкого процесу управління раціонами харчування залежно від інтенсивності й спрямованості тренувальної діяльності [1].

Для глибшого обґрунтування доцільності створення нового Android-додатку варто розглянути основні функціональні вимоги до такої системи та порівняти їх із можливостями наявних аналогів. Це дозволить чітко окреслити прогалини, які може заповнити новий програмний продукт:

– персоналізований моніторинг спортивного харчування, більшість існуючих програм не дозволяють налаштовувати програму спортивних добавок на основі тренувального циклу користувача (наприклад, фаза набору маси, сушки або відновлення). Нова програма дозволяє створювати індивідуальні плани споживання з можливістю вказати тип добавки, дозування, частоту використання та тривалість дії. Це дозволяє точно контролювати харчування та допомагає запобігти надмірному або неналежному споживанню речовин;

– інтеграція з фізичною активністю, аналоги часто ізольовано відстежують лише харчування або лише тренування. Передбачається реалізація зв’язку між вживанням спортивного харчування та типом фізичної активності. Наприклад, додаток може враховувати витрачені калорії під час тренування для адаптації плану споживання добавок;

– автоматичні нагадування, вживання спортивного харчування вимагає дотримання чіткого графіку. Наявні додатки або не мають системи нагадувань, або реалізують її надто загально. Запропонований додаток матиме гнучку систему сповіщень: користувач зможе налаштувати нагадування щодо кожного продукту окремо, враховуючи дні тижня, час доби, після або перед тренуванням тощо;

– база спортивного харчування з українським контекстом, однією з найбільших прогалин є відсутність баз даних із популярними на українському ринку добавками. Новий додаток матиме вбудовану базу з найпоширенішими брендами, а також можливість додавати власні продукти, імпортувати етикетки чи фотографії банок, зберігати їх та використовувати повторно. Це значно спростить введення інформації та зробить її точнішою;

– можливість ведення статистики та аналітики, більшість поточних систем не забезпечують аналіз ефективності вживання спортивного харчування. У запропонованому додатку передбачається система графіків і звітів щодо використаних продуктів, реакцій організму, змін ваги, результатів тренувань;

– сучасний та зручний інтерфейс, завдяки використанню Jetpack Compose, інтерфейс нового додатку буде інтуїтивно зрозумілим, швидким у роботі та адаптивним до будь-яких розмірів екранів. Це особливо важливо для щоденного використання, коли користувач потребує швидкого доступу до функцій без зайвих натискань;

– безпека даних та конфіденційність, оскільки додаток зберігатиме особисту інформацію про здоров’я та раціон, важливою є реалізація механізмів

захисту даних. Планується локальне зберігання з можливістю шифрування, а також резервне копіювання на Google Drive (за бажанням користувача).

Серед популярних додатків для відстеження харчування та споживання добавок спортсменами є MyFitnessPal, YAZIO, Lifesum та Cronometer. Вони пропонують широкий асортимент продуктів, дозволяють вести щоденник харчування та розраховувати калорії та макронутрієнти. Однак вони не завжди зосереджені на спортивному харчуванні, а часто зосереджуються на загальному харчуванні та контролі ваги. Наприклад, MyFitnessPal не пропонує детальної класифікації спортивних добавок, таких як гейнери, протеїни, BCAA або креатин.

Користувач змушений самостійно додавати свої продукти, що ускладнює облік. Крім того, деякі функції є платними, що обмежує доступність додатка для широкого кола користувачів.

Додаток YAZIO (рис. 1.1) також пропонує інтерфейс планування харчування, але він зосереджений на дієтах та схудненні. Він дозволяє вести щоденник харчування, підраховувати калорії, білки, жири та вуглеводи, вручну вводити страви або сканувати штрих-коди продуктів для швидкого введення даних.

Додаток містить велику базу даних продуктів, що спрощує введення даних. Користувач візуалізує свій прогрес у вигляді графіків, оцінює якість свого раціону та відстежує свою вагу, відсоток жиру в організмі, рівень цукру в крові та навіть артеріальний тиск на основі введених даних.

Платна версія пропонує більш глибокий аналіз даних, широкий вибір дієт, відстеження довгострокових результатів та не містить реклами. Простий у використанні та інтуїтивно зрозумілий, додаток ідеально підходить для тих, хто хоче контролювати свій раціон, виробити свідомі харчові звички та чітко розуміти зв’язок між харчуванням, фізичною активністю та досягненням своїх цілей.

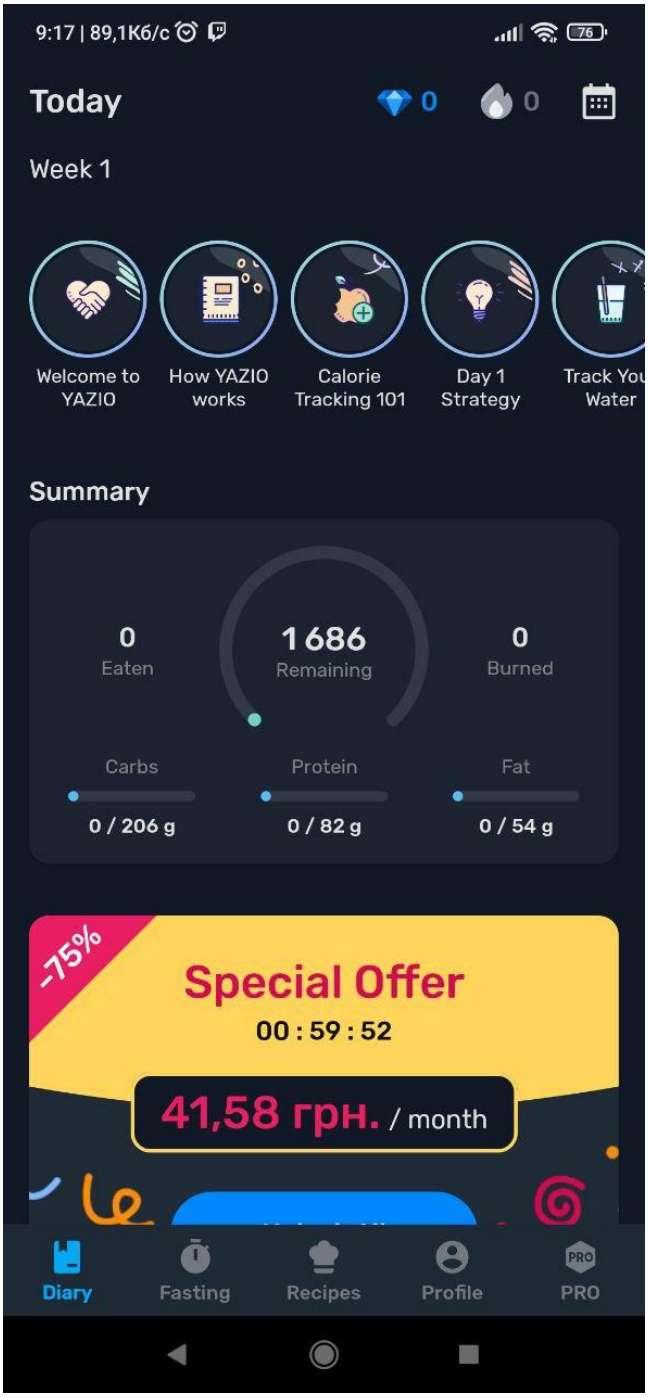

Рисунок 1.1 – Головний екран YAZIO [2]

Окремо варто відзначити такі спеціалізовані рішення, як Eating Buddy, додаток заохочує користувачів відстежувати свій голод і ситість, записувати те, що вони їдять, і розуміти причини свого харчового вибору, головний екран якого зображений на рисунку 1.2 або GymKeeper це умовний застосунок для фітнесу, спрямований на контроль, планування й оптимізацію тренувань у спортзалі, головний екран якого також зображений на рисунку 1.3.

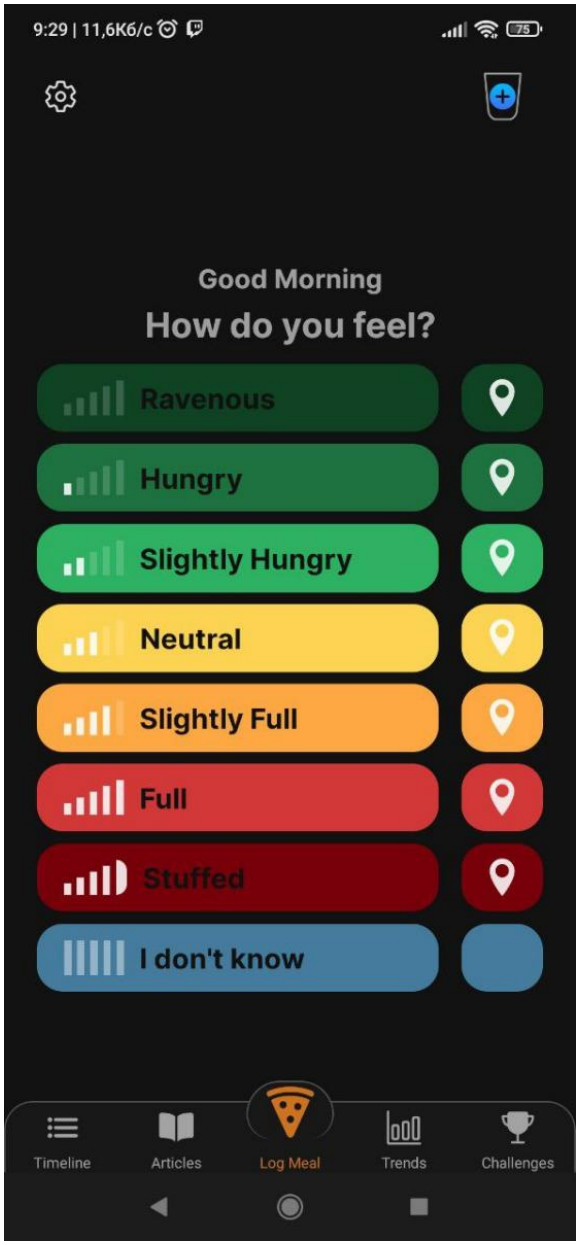

Рисунок 1.2 – Eating Buddy [3]

Перший дозволяє створювати списки прийому спортивних добавок та встановлювати нагадування, однак має обмежений функціонал щодо інтеграції з фізичною активністю та не враховує індивідуальні параметри користувача [4].

GymKeeper більше орієнтований на тренування, а не на харчування, хоча й дозволяє дещо вести облік добавок. Таким чином, існуючі рішення або надто загальні, або занадто вузькоспеціалізовані, що не дозволяє ефективно поєднати всі необхідні функції для цільової аудиторії – людей, які активно займаються спортом і вживають спортивне харчування.

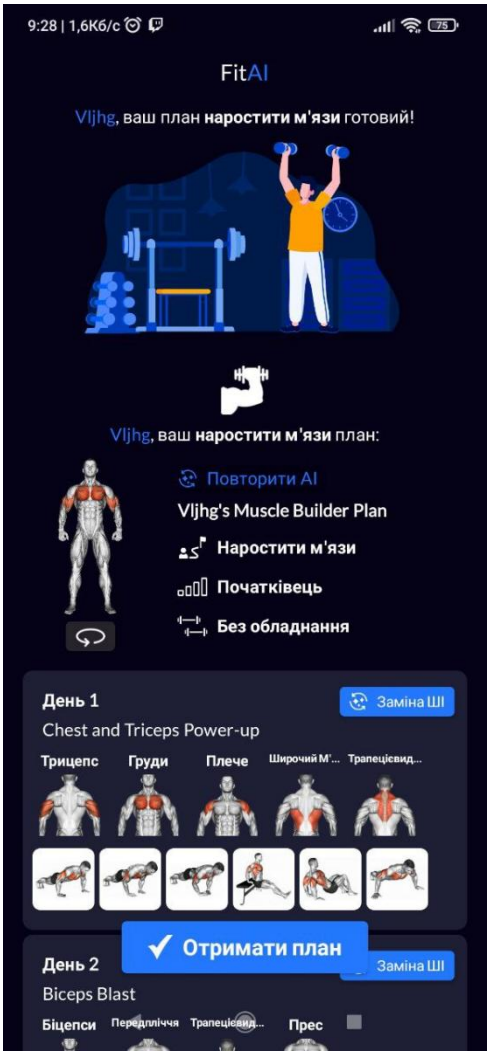

Рисунок 1.3 – GymKeeper [5]

Ще однією проблемою є відсутність локалізованих продуктів, що підтримують українську мову та українські реалії спортивного харчування. Більшість додатків орієнтовані на американський або західноєвропейський ринок, що впливає на асортимент дієтичних добавок, доступних у базі даних, та ускладнює управління даними для українських користувачів. Крім того, значна частина існуючих рішень базується на застарілих підходах до розробки інтерфейсу, що робить їх незручними на сучасних Android-пристроях [6].

Враховуючи ці аспекти, можна зробити висновок про доцільність створення нової мобільної системи, яка враховує недоліки існуючих аналогічних рішень та адаптується до потреб цільової аудиторії. Головною метою розробленого Android-додатку є забезпечення зручного та інтуїтивно

зрозумілого інструменту для щоденного моніторингу споживання спортивного харчування, з можливістю налаштування, створення графіків прийому, нагадувань та інтеграції його з фізичною активністю.

Використання сучасних технологій Kotlin та Jetpack Compose дозволяє реалізувати високопродуктивний додаток, адаптований до різних розмірів екрану та з приємним користувацьким інтерфейсом. Використання Jetpack Compose дозволяє уникнути надмірної складності в розробці інтерфейсу, тим самим спрощуючи процес створення адаптивних екранів та динамічного контенту, що є важливим для мобільних додатків у секторах охорони здоров'я та спорту. Kotlin, сучасна мова програмування для Android, дозволяє створювати менш об'ємний та безпечніший код, ніж Java, що покращує підтримку та масштабованість застосунку [7].

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою кваліфікаційного проєкту є розробка сучасного Android-додатку для відстеження споживання спортивного харчування з використанням мови програмування Kotlin та фреймворку користувацького інтерфейсу Jetpack Compose.

Серед цих інструментів чільне місце займають мобільні додатки, орієнтовані на спортсменів, любителів фітнесу та людей, які регулярно вживають спортивні добавки. Однак більшість існуючих рішень є універсальними та не враховують особливості споживання спортивних добавок, таких як білки, амінокислоти, креатин тощо.

Для досягнення мети кваліфікаційної роботи бакалавра, поставлені наступні завдання:

– провести аналіз предметної області та визначити основні вимоги до функціональності мобільного додатку для контролю споживання спортивного харчування;

– ознайомитися з сучасними підходами, технологіями та інструментами розробки Android-додатків, зокрема Kotlin та Jetpack Compose;

– проаналізувати та визначити вимоги до розроблюваного програмного забезпечення та спроектувати архітектуру додатку, включаючи структуру даних, логіку роботи, навігацію та інтерфейс користувача;

– розробити Android-додатк з використанням Kotlin та Jetpack Compose;

– провести тестування додатку на коректність роботи та відповідність вимогам.

Висновки до розділу 1

Аналіз поточної проблеми виявив значний інтерес користувачів до мобільних інструментів, що сприяють підтримці здорового способу життя, зокрема для контролю харчування та спортивного харчування. Однак на ринку мобільних додатків серйозно бракує спеціалізованих рішень, присвячених обліку та контролю спортивного харчування, які не перевантажені другорядними функціями та відповідають сучасним вимогам щодо дизайну, зручності використання та технологічної ефективності.

Також було виявлено, що сучасні підходи до розробки для Android, включаючи мову програмування Kotlin та фреймворк інтерфейсу користувача Jetpack Compose, дозволяють ефективно впроваджувати мобільні продукти із сучасною архітектурою, адаптивним інтерфейсом та високою продуктивністю.

Таким чином, на основі проведеного аналізу було розроблено комплексне технічне завдання, що враховує як сучасні тенденції в мобільному програмуванні, так і реальні потреби постійних користувачів спортивного харчування. Розробка такого додатку вирішує актуальну практичну проблему та пропонує потенціал для подальшого розвитку, розширення функціональності та інтеграції з іншими фітнес- або медичними сервісами.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Для проекту з розробки програми відстеження споживання спортивного харчування для Android за допомогою Kotlin та Jetpack Compose важливим завданням було побудувати чітку модель програмної системи, здатну відображати ключові процеси взаємодії користувача з системою, а також структуру даних, функціональність та зв'язки між компонентами. Для моделювання програмної системи було обрано об'єктно-орієнтовану модель, оскільки вона природним чином відображає основні сутності (користувач, харчовий продукт, журнал споживання) як класи, а їх взаємодії як методи, асоціації та залежності. Таке рішення сприяє прозорому процесу проектування, легшому впровадженню та підтримці, а також пропонує можливість масштабування програми.

Для виявлення та аналізу потреб ми використовували методи інтерв'ювання з потенційними користувачами (спортсменами, тренерами), аналіз існуючих програм в App Store/Google Play та побудову діаграм варіантів використання для формалізації функціональних вимог. Ці методи забезпечують вичерпний вибір функцій та взаємодій користувача з програмою, а також сприяють чіткому вибору основних функціональних модулів.

Для синтезу програмних моделей було використано методи формального UML-моделювання: діаграми класів, що описують ключові сутності («Користувач», «Продукт», «Щоденник харчування»), діаграми послідовностей, що відображають сценарії взаємодії (введення нового продукту в раціон, отримання рекомендацій щодо споживання), та діаграми діяльності, що візуалізують процес введення даних у застосунок. Таке формалізоване моделювання забезпечує чіткий вибір функціональних, логічних та інформаційних аспектів системи [8].

Специфікації застосунку описують функціональні можливості (реєстрація користувачів, введення інформації про харчування, генерація статистичних звітів), нефункціональні вимоги (висока продуктивність, простота використання, сумісність з версіями Android від 8.0) та вимоги до інтерфейсу (інтуїтивно зрозумілий дизайн, швидке введення даних). Такий формальний опис вимог забезпечує добре розуміння між учасниками проекту та служить керівництвом для впровадження, на рисунку 2.1 показана діаграма.

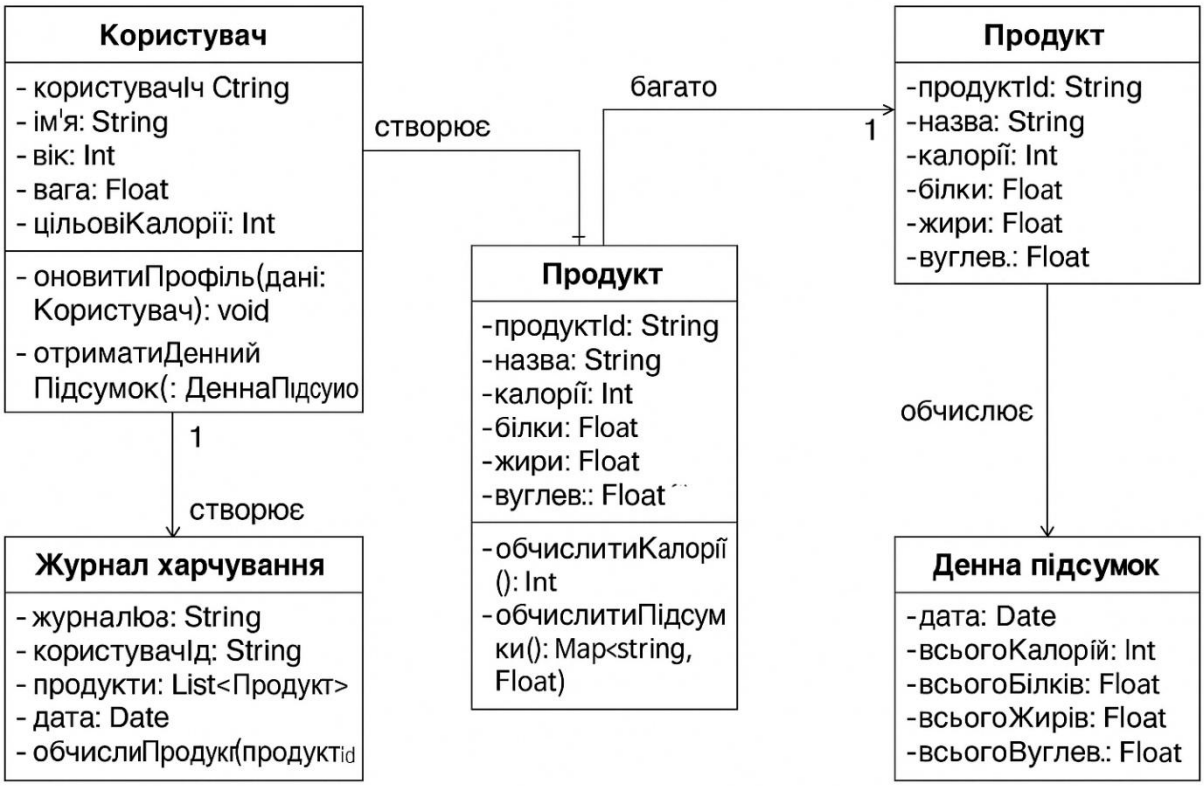

Рисунок 2.1 – UML-діаграма класів

Вона відображає чотири ключові класи для Android-додатку для контролю споживання спортивного харчування:

– користувач, зберігає основні дані про користувача – ID, ім’я, вік, вагу, цільову калорійність. Має методи для оновлення профілю та отримання денного підсумку. Зв’язаний відношенням 1-багато із класом «Журнал харчування» [9];

– журнал харчування, містить записи про вжиті продукти, дату, а також посилання на конкретні продукти;

– продукт, зберігає інформацію про продукт (ID, назву, калорії, вміст білків, жирів, вуглеводів). Зв’язок багато-багато із класом «Журнал харчування», тому один журнал містить багато продуктів, а продукт може бути у багатьох журналах;

– денний підсумок, зберігає дату, загальну кількість калорій, білків, жирів, вуглеводів для конкретного дня.

Специфікація вимог для застосунку містить, функціональні вимоги і нефункціональні вимоги.

Функціональні вимоги описують конкретні дії, які додаток повинен виконувати. Насамперед система має забезпечити користувачеві можливість додавати продукти спортивного харчування до особистого раціону. Для цього необхідно передбачити зручний інтерфейс введення інформації про назву продукту, його харчову цінність (білки, жири, вуглеводи, калорійність) та кількість, що споживається [10]. Важливо, щоб додаток автоматично обчислював добове надходження калорій та макроелементів на основі внесених даних, а також порівнював ці значення з індивідуальними цільовими показниками користувача.

Функціональні вимоги Android-додатку:

– реєстрація користувача, авторизація;

– збереження профілю користувача (вік, вага, цілі);

– збереження інформації про введені продукти харчування;

– зведення даних для отримання статистичних результатів (калораж, білки, жири, вуглеводи);

– надсилання нагадувань про введення даних для формування повноцінного профілю харчування.

Нефункціональні вимоги характеризують якісні характеристики додатку та визначають, наскільки ефективно він виконує функціональні задачі. Однією з основних вимог є зручність користування. Інтерфейс має бути інтуїтивно

зрозумілим, мінімалістичним, із логічно структурованими екранами та зрозумілими візуальними елементами. Застосування Jetpack Compose у поєднанні з дизайном на основі Material Design дозволяє досягнути сучасного вигляду та високої ергономічності.

Нефункціональні вимоги Android-додатку:

– сумісність із версіями Android від 8.0;

– простий, інтуїтивно-зрозумілий інтерфейс, реалізований із використанням Jetpack Compose;

– робота в умовах обмежених ресурсів (оперативної пам’яті, швидкості процесора);

– надійне збереження даних у SQLite/Room для можливості роботи в умовах відсутності Інтернету;

– гарантований захист введених даних шляхом використання внутрішньої бази даних, а в майбутньому – підтримка шифрування даних.

Отриманий результат моделювання дозволив чітко вказати ключові елементи програмної архітектури, встановити взаємозв’язки між функціональними модулями, виділити основні процеси взаємодії користувача із системою. Таке рішення аргументовано сприяє швидшому процесу реалізації, простоті подальшої підтримки й розвитку програмного забезпечення, а також забезпечує прозору документовану структуру для усіх учасників проєкту. Застосування UML-діаграм гарантує чітке бачення логіки й структури програми, сприяє уникненню потенційних проблем у процесі розробки й сприяє отриманню програмного продукту, максимально орієнтованого на потреби кінцевого користувача.

Модель даних відповідає за збереження інформації про продукти харчування, графік, історію прийому. Представлення реалізоване з використанням Jetpack Compose – декларативного UI-інструментарію, що дозволяє створювати гнучкі та адаптивні інтерфейси. Компонент ViewModel забезпечує зв’язок між цими шарами, опрацьовує дії користувача та оновлює інтерфейс.

Для зберігання даних на пристрої використано Room – бібліотеку для роботи з локальною базою даних SQLite. Це дозволяє забезпечити збереження введеної інформації та її доступність у будь-який момент [11].

Окрему увагу приділено проєктуванню інтерфейсу: реалізовано головний екран з календарем і статистикою, форму додавання продуктів, історію споживання та екран з профілем користувача. Усі екрани побудовано відповідно до принципів Material Design, що забезпечує сучасний вигляд і зручність у використанні.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Розробка мобільного додатку для контролю споживання спортивного харчування є надзвичайно актуальним завданням для спортсменів, бодібілдерів та всіх, хто веде активний спосіб життя. Такий додаток дасть змогу користувачам ефективно відстежувати свій раціон, контролювати споживання життєво важливих нутрієнтів, таких як білки, жири, вуглеводи, а також вітамінів та мінералів, ставити перед собою цілі та впевнено їх досягати. Застосування передових технологій, а саме Kotlin для реалізації основної логіки та Jetpack Compose для створення візуального інтерфейсу, гарантує розробку продуктивного, легко масштабованого та зручного у використанні додатку.

Для створення надійного та функціонального Android-додатку для контролю спортивного харчування з використанням Kotlin та Jetpack Compose, необхідно ретельно підібрати інструменти та бібліотеки. Kotlin буде основною мовою програмування, як рекомендована Google для Android-розробки, вона пропонує лаконічний синтаксис, високу безпеку від нульових посилань (null safety), розширення функцій та покращену взаємодію з Java-кодом. Це забезпечує високу продуктивність та мінімізує кількість помилок. Для побудови користувацького інтерфейсу обирається Jetpack Compose – сучасний декларативний UI-інструментарій від Google. Він дозволяє створювати

інтерфейси за допомогою Kotlin, значно спрощуючи та прискорюючи розробку UI, роблячи його більш інтуїтивним та реактивним [12].

Щодо архітектури додатку, оптимальним вибором є MVVM (Model-View-ViewModel). Ця архітектура забезпечує чітке розділення відповідальності між компонентами. Model представляє дані та бізнес-логіку, наприклад, моделі даних для продуктів, прийомів їжі та цілей. View, реалізований за допомогою Jetpack Compose, відповідає за відображення UI та отримання взаємодій користувача. ViewModel містить логіку для підготовки даних для View, обробляє події від View та взаємодіє з Model, наприклад, логіка для розрахунку калорій або збереження споживання. Для додаткової підтримки надійної архітектури використовуються Android Architecture Components. Це включає Lifecycle для керування життєвим циклом компонентів, LiveData або StateFlow для реактивного потоку даних між ViewModel та UI (причому StateFlow кращий для Jetpack Compose завдяки ефективнішій роботі з асинхронними даними), а також Room Persistence Library для локального зберігання даних. Room – це абстракція над SQLite, що спрощує роботу з базами даних та гарантує безпеку під час компіляції.

Для управління даними та їх зберігання, Room Database буде основною базою даних для локального зберігання інформації про продукти, прийоми їжі, користувацькі налаштування, цілі та історію споживання, забезпечуючи швидкий доступ до даних офлайн. Залежно від потреб, може бути розглянута інтеграція з віддаленим API для отримання розширеної інформації про продукти, синхронізації даних або використання сторонніх баз даних харчування, для чого підійдуть Ktor Client або Retrofit. Kotlinx.Serialization або Gson використовуватимуться для серіалізації та десеріалізації об’єктів Kotlin у формат JSON при роботі з API.

Серед додаткових бібліотек та інструментів слід виділити Hilt або Koin для управління залежностями (Dependency Injection), що спрощує тестування та покращує модульність коду. Kotlin Coroutines будуть використовуватися для асинхронних операцій, таких як робота з базою даних та мережеві запити,

забезпечуючи неблокуючий режим та роблячи асинхронний код більш читабельним. Accompanist може бути корисним для розширення функціоналу Jetpack Compose, наприклад, для анімації або обробки дозволів. Для візуалізації даних споживання та прогресу користувача у вигляді графіків будуть інтегровані бібліотеки для побудови діаграм, такі як MPAndroidChart, або ж реалізовані власні рішення на основі Compose Canvas. І, звичайно, для забезпечення сучасного та послідовного зовнішнього вигляду додатку будуть використовуватися компоненти Google Material Design в Jetpack Compose [13].

Ефективна розробка Android-додатку з Kotlin та Jetpack Compose вимагає застосування перевірених методів та підходів. Основний метод – це модульний підхід та компонентна розробка**.** Додаток буде розбитий на логічні модулі, такі як app, data, domain та окремі функціональні модулі. Це покращить організацію коду, прискорить компіляцію, полегшить тестування та подальше розширення функціоналу. Інтерфейс буде будуватися з невеликих, багаторазово використовуваних та незалежних Composable-функцій, оскільки кожен Composable відповідатиме за невелику частину UI, дозволяючи легко їх комбінувати та підтримувати.

Обробка даних та логіка буде побудована на шаблоні Репозиторій (Repository) для абстракції джерел даних, таких як локальна база даних чи віддалений API. Репозиторій відповідатиме за отримання, зберігання та кешування даних, надаючи єдиний інтерфейс для ViewModel. Для конкретних бізнес-операцій створюватимуться окремі класи – Use Cases (Interactor), наприклад, AddFoodEntryUseCase або CalculateDailyMacrosUseCase. Це інкапсулює бізнес-логіку та робить ViewModel більш легкою та сфокусованою. Всі операції, що можуть блокувати основний потік, виконуватимуться в окремих Kotlin Coroutines для забезпечення плавної роботи додатку без зависань [14].

Для обґрунтування зробленого вибору доцільно порівняти основні популярні середовища й фреймворки для розробки застосунків (як для

платформи Android, так і мультиплатформених), виділивши ключові переваги й недоліки що зображено в таблиці 2.1.

Таблиця 2.1 – Порівняння середовищ розробки.

Характеристика / IDE	Android Studio (Kotlin/Java)	Flutter (Dart)	React Native (JavaScript/TypeScript)	Xamarin (.NET/C#)
Платформи	Android	Android, iOS, Web	Android, iOS	Android, iOS, Windows
Мова програмування	Kotlin, Java	Dart	JavaScript, TypeScript	C#
Інструменти UI	Jetpack Compose, XML	Flutter Widgets	React-компоненти	Xamarin.Forms, MAUI
Офіційна підтримка Android	Повна підтримка	Часткова	Часткова	Часткова
Продуктивність	Висока (рідний код)	Висока (рендеринг)	Середня (через JavaScript Bridge)	Середня
Крива навчання	Середня	Середня/Висока	Низька/Середня	Середня/Висока
Швидкість розробки	Середня	Висока	Висока	Середня
Гнучкість дизайну	Висока (Compose)	Висока	Висока	Середня
Інтеграція з Android API	Повна	Через плагіни	Через native-модулі	Через Binding
Підтримка гарячого перезапуску (Hot Reload)	Обмежена (Compose Preview)	Повна	Повна	Часткова
Ком’юніті та ресурси	Дуже велике	Зростає швидко	Дуже велике	Менше

Для формування структури програми використано метод об’єктно-орієнтованого проєктування, спрямований на виділення ключових сутностей у вигляді класів, чітке окреслення їх взаємодії для отримання прозорої й логічної структури взаємозв’язків. Таке рішення сприяє формуванню модульності програмного продукту, відкриваючи можливість виділення функціональності в

самостійні модулі для спрощення процесу підтримки й вдосконалення застосунку [15].

Для введення даних від користувача застосовано методи валідації в реальному часі, спрямовані на контроль правильності введення інформації про харчування. Реалізовано перевірку введених даних прямо в момент взаємодії із формами введення для уникнення помилок, неточності інформації про харчування, забезпечення максимально прозорої взаємодії із користувачами. Таке рішення сприяє формуванню достовірної інформації для подальшого проведення аналізу результатів харчування.

Для організації взаємодії із внутрішньою базою даних застосовано метод використання ORM для оптимізації доступу до даних [16]. Застосунок спрямований на роботу із SQLite як основним засобом збереження даних, тому метод взаємодії із базами даних сприяє формуванню прозорої структури доступу для отримання інформації про введені користувачами продукти харчування, результатів їхньої активності, статистичних даних для подальшого аналізу.

Для забезпечення максимально швидкого отримання результатів реалізовано методи кешування отриманих даних, спрямовані на оптимізацію доступу в умовах низької швидкості Інтернет-з’єднання. Таке рішення відкриває можливість користування застосунком навіть в умовах відсутності Інтернету, сприяючи формуванню сприятливого клієнтського досвіду для спортсменів.

Для ефективної реалізації застосунку для контролю спортивного харчування важливим етапом є формування оптимальних алгоритмів, спрямованих на отримання достовірної інформації про введені дані, формування результатів для користувача, взаємодії із зовнішніми пристроями для отримання даних про активність, а також оптимізації процесу обчислень для формування персональних рекомендацій. Такі алгоритми сприяють прозорому й швидкому отриманню результатів, відкриваючи можливість для

забезпечення основних функціональних вимог, спрямованих на оптимальне функціонування застосунку [17].

Для введення даних реалізовано алгоритм валідації введених відомостей про харчування, спрямований на контроль правильності введення інформації (кількість продукту, вага порції, тип продукту). Алгоритм валідації відкриває можливість усунення помилок введення даних у реальному часі, сприяючи отриманню достовірної інформації для подальшого аналізу [18].

Висновки до розділу 2

На етапі аналізу було вивчено предметну область спортивного харчування, особливості споживання добавок, а також поведінкові моделі користувачів, які ведуть облік харчування. Це дозволило сформулювати реальні сценарії використання та зрозуміти ключові функціональні потреби користувачів. Було визначено цільову аудиторію додатку, основні завдання, які він має вирішувати, а також типові щоденні дії, які мають бути підтримані у функціоналі.

У результаті визначення вимог до програмного забезпечення було сформовано перелік функціональних та нефункціональних вимог. До функціональних увійшли облік продуктів, графік споживання, ведення історії, відображення статистики, а також push-сповіщення. До нефункціональних вимог – продуктивність, адаптивність, підтримка офлайн-режиму, безпека та зручність інтерфейсу. Ці вимоги стали основою для проєктування майбутнього додатку.

Здійснено вибір методів, засобів і технологій для реалізації програмного продукту. Як основну мову програмування обрано Kotlin – сучасну, безпечну та рекомендовану Google для Android-розробки. Для побудови інтерфейсу використано Jetpack Compose, що забезпечує декларативний стиль розробки UI, спрощує підтримку коду та пришвидшує розробку.

РОЗДІЛ 3
РОЗРОБКА ANDROID-ДОДАТКУ ДЛЯ КОНТРОЛЮ СПОЖИВАННЯ СПОРТИВНОГО ХАРЧУВАННЯ

3.1 Практична реалізація об’єкта проєктування

Реалізація мобільного додатку здійснюється поетапно та відповідно до спроєктування архітектури, вимог до функціоналу й середовища виконання. Основна мета полягає в створенні ефективного та зручного інструменту для контролю споживання спортивного харчування, що працює стабільно на Android-пристроях.

На цьому етапі здійснено повний цикл створення програмного продукту – від побудови архітектури та модулів взаємодії до написання функціонального коду, проєктування інтерфейсу, реалізації внутрішньої логіки та проведення тестування.

Для реалізації обрано мову програмування Kotlin, яка є офіційно рекомендованою Google для Android-розробки. У якості фреймворку для розробки інтерфейсу було використано Jetpack Compose – сучасну декларативну UI-бібліотеку, яка значно спрощує створення адаптивного та гнучкого інтерфейсу користувача. Такий підхід дозволить зменшити кількість шаблонного коду та забезпечити високий рівень узгодженості між логікою та візуальною частиною програми. Kotlin дозволяє писати чистіший, лаконічніший код із меншою кількістю помилок завдяки нуль-безпечності, розширенням функцій та підтримці корутин для асинхронного програмування. Для побудови інтерфейсу користувача було обрано Jetpack Compose – декларативну UI-бібліотеку нового покоління від Google. Цей інструмент дозволяє створювати інтерфейси швидше та ефективніше порівняно з XML-макетами. Завдяки Compose стало можливим створити гнучкий, адаптивний та візуально привабливий інтерфейс, який легко масштабується та модифікується.

Також було використано Room – об’єктно-реляційну бібліотеку для роботи з локальною базою даних, яка забезпечує збереження історії прийому

спортивного харчування та інших даних користувача. Архітектурно додаток побудовано на основі патерну MVVM (Model-View-ViewModel), що дозволяє розділити логіку програми, зберігання стану та візуалізацію.

На рисунку 3.1 зображено як має виглядати структура файлів для вдалого функціонування програми, за допомогою цього можна запобігти плутанини в коді.

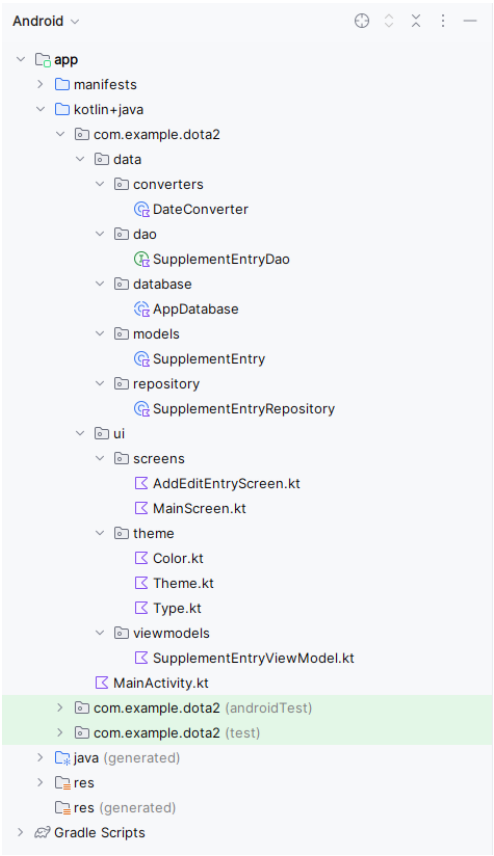

Рисунок 3.1 – Вигляд структури файлів

Структура файлів Android-додатку в Android Studio є важливою складовою розробки, оскільки визначає організацію всіх ресурсів, вихідного коду та конфігураційних файлів проєкту. Основним є каталог app, у якому міститься папка src, що в свою чергу поділяється androidTest та main. У папці main є папка kotlin яка має всі ключові файли для коректної роботи програми, в даній папці знаходяться наступні файли:

– com.example.dota2, в даній папці розміщено папка з елементами інтерфейсу, та файлом який відповідає за навігацію, за допомогою MainActivity відбувається навігація по додатку, в папці ui розміщено екран ViewModel який управляє станом UI та взаємодіє з репозиторієм, також розміщено файл який відповідає за головний екран додатку на якому розміщено список споживань доданих раніше, за допомогою кнопки «Додати споживання» дає можливість додавання бажаний продукт в список для відслідковування;

– data, в цій папці розміщено Модель Даних (Data Model - Room Entity) він представляє одиницю спортивного харчування та її споживання, Repository який є абстракцією доступу до даних, Room Database, є класом для ініціалізації бази даних.

Її дотримання її дозволяє підтримувати порядок, масштабованість і зрозумілу архітектуру додатку, що є особливо важливим.

Для початку був створений головний екран в якому знаходиться екран споживань який зображений на рисунку 3.2, його основна мета – надати миттєвий огляд поточної статистики споживання, прогресу в досягненні цілей та швидкий доступ до ключових функцій програми, таких як додавання нового споживання, або його видалення за необхідністю. Він допомагає переглянути добавлені користувачем продукти в список для контролю харчування. В лістингу 3.1 наведено що він представлений у вигляді списка з самими добавленими продуктами які показують:

– назву продукту;

– вказані калорії;

– кількість в грамах.

Лістинг 3.1 – Верхня частина інтерфейсу користувача з елементами

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun MainScreen(
   navController: NavController,
   viewModel: SupplementEntryViewModel
) {
```

```
    val entries by viewModel.allEntries.collectAsState(initial = emptyList())
    Scaffold(
        topBar = {
            TopAppBar(title = { Text("Мої Добавки") })
        },
        floatingActionButton = {
            FloatingActionButton(onClick = { navController.navigate("add_edit_entry") }) {
                Icon(Icons.Filled.Add, "Додати запис")
            }
        }
    ) { paddingValues ->
        if (entries.isEmpty()) {
            Box(
                modifier = Modifier
                    .fillMaxSize()
                    .padding(paddingValues),
                contentAlignment = androidx.compose.ui.Alignment.Center
            ) {
                Text("Поки що немає записів. Натисніть '+' щоб додати.")
            }
        } else {
            LazyColumn(
                modifier = Modifier
                    .fillMaxSize()
                    .padding(paddingValues)
                    .padding(horizontal = 16.dp, vertical = 8.dp)
            ) {
                items(entries) { entry ->
                    SupplementEntryCard(entry = entry) {
                        viewModel.deleteEntry(entry)
                    }
                    Spacer(modifier = Modifier.height(8.dp))
                }
            }
        }
    }
}
```

кінець лістингу 3.1

На рисунку 3.2 зображений головний екран, на ньому можна побачити список з продуктами який добавив користувач для відслідковування його харчування. Нижче зображена кнопка функціонал якої це добавити потрібний продукт у список для його зберігання. Також є можливість видаляти непотрібний користовачу вже доданий раніше продукт.

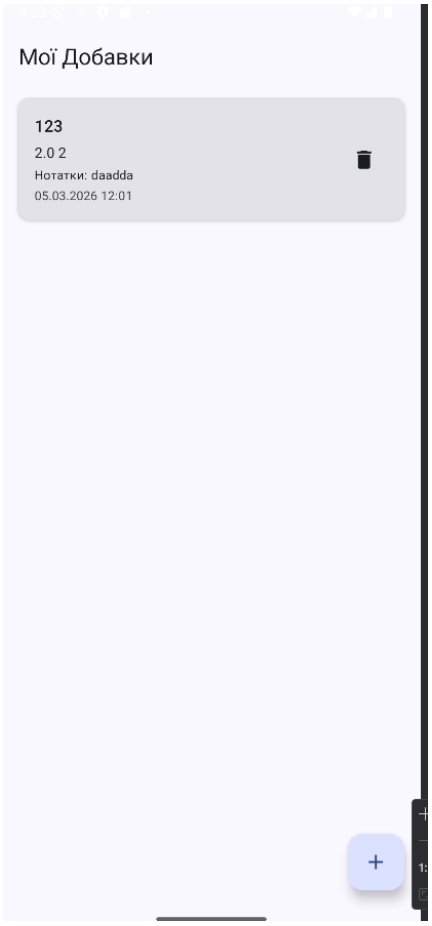

Рисунок 3.2 – Головний екран

Для того щоб була можливість додавати певні продукти користовачу в лістингу 3.2 наведено створений спеціальний екран з інтерфейсом додавання продукту який потрібне користувачеві. В даном екрані також є можливість вписати параметри продукту такі як:

– назву бажаного продукту який користувач хоче добавити;

– вказані калорії, для контролю деної норми для його маси тіла;

– кількість в грамах продукту який був вживаний.

Лістинг 3.2 – Інтерфейс додавання продукту

```
@Composable
fun AddSupplementScreen(
    onNavigateBack: () -> Unit,
    viewModel: SportSupplementViewModel
) {
    var name by remember { mutableStateOf("") }
    var quantity by remember { mutableStateOf("") }
```

```
var unit by remember { mutableStateOf("г") } // Початкове значення
Scaffold(
  topBar = {
    TopAppBar(
      title = { Text("Додати Споживання") },
      navigationIcon = {
        IconButton(onClick = onNavigateBack) {
          Icon(Icons.Filled.ArrowBack, contentDescription = "Назад")
        }
      }
        )
```

кінець лістингу 3.2

На рисунку 3.3 зображено візуальний вигляд екрану додавання продукту. Якщо користувач ввів неправильні дані, програма це помітить і видасть помилку, можна вибрати потрібну дату та час.

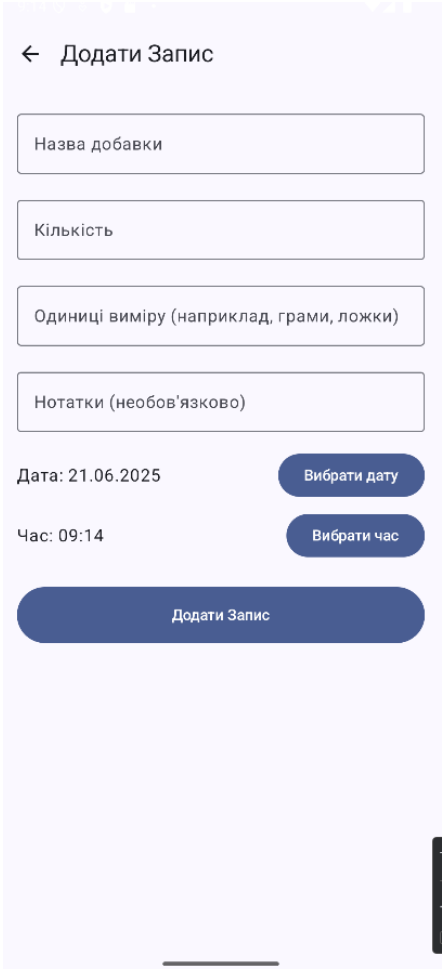

Рисунок 3.3 – Додавання продукту до списку

В лістингу 3.3 це продемонстровано окремому вікні щоб процес введення інформації користувачем був максимально простим, чітким та зрозумілим. Це дозволить усунути непотрібне візуальне навантаження, дозволить уникнути помилок при введенні даних користувачем.

Лістинг 3.3 – Інтерфейс додавання продукту з вказуванням його параметрів

```
OutlinedTextField(
            value = name,
            onValueChange = { name = it },
            label = { Text("Назва добавки") },
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(16.dp))
        OutlinedTextField(
            value = quantity,
            onValueChange = { quantity = it },
            label = { Text("Кількість") },
            keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Number),
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(16.dp))
        OutlinedTextField(
            value = unit,
            onValueChange = { unit = it },
            label = { Text("Одиниці виміру (наприклад, грами, ложки)") },
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(16.dp))
        OutlinedTextField(
            value = notes,
            onValueChange = { notes = it },
            label = { Text("Нотатки (необов’язково)") },
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(16.dp))
Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceBetween,
            verticalAlignment = androidx.compose.ui.Alignment.CenterVertically
        ) {
            Text(text = "Дата: ${android.text.format.DateFormat.format("dd.MM.yyyy",
selectedDate)}")
            Button(onClick = { datePickerDialog.show() }) {
                Text("Вибрати дату")
            }
        }
```

```
        Spacer(modifier = Modifier.height(8.dp))
        Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceBetween,
            verticalAlignment = androidx.compose.ui.Alignment.CenterVertically
        ) {
            Text(text = "Час: ${android.text.format.DateFormat.format("HH:mm", selectedDate)}")
            Button(onClick = { timePickerDialog.show() }) {
                Text("Вибрати час")
            }
        }
```

кінець лістингу 3.3

В нижній частині даного сегмента додавання продукту знаходится кнопка яка призначенна для завершення додавання продукту, і у випадку якщо дані були введені не коректно от буди видавати помилку, в інших випадках буде повернення до головного екрану де можна переглянути доданий продукт, це наведено у лістингу 3.4.

Лістинг 3.4 – Функціонал кнопки для додавання продукту

```
Spacer(modifier = Modifier.height(24.dp))
        Button(
            onClick = {
                if (name.isNotBlank() && quantity.isNotBlank() && unit.isNotBlank()) {
                    val parsedQuantity = quantity.toDoubleOrNull()
                    if (parsedQuantity != null) {
                        if (entryId == null) {
                            viewModel.addEntry(name, parsedQuantity, unit, selectedDate, notes.ifBlank { null })
                        } else {
                            viewModel.updateEntry(SupplementEntry(entryId, name, parsedQuantity, unit, selectedDate, notes.ifBlank { null }))
                        }
                        navController.popBackStack()
                    }
                }
            },
            modifier = Modifier.fillMaxWidth(),
            contentPadding = PaddingValues(16.dp)
        ) {
            Text(if (entryId == null) "Додати Запис" else "Зберегти Зміни")
        }
```

кінець лістингу 3.4

Розроблена модель даних для андроїд додатку в лістингу 3.5, її суть полягає в тому, щоб створити логічну та ефективну структуру для всієї інформації, з якою працює додаток. Без чітко визначеної моделі даних неможливо коректно зберігати, отримувати, оновлювати та видаляти інформацію. Якщо потрібно додати нову функціональність, наприклад, можливість додавати зображення до споживання, модель даних чітко показує, як інтегрувати цю нову властивість, не порушуючи існуючу структуру. Вона забезпечує цілісність даних, запобігаючи збереженню некоректної або неповної інформації, що є критично важливим для додатків, де точність фінансових даних має першочергове значення, це наведено у лістингу 3.5.

Лістинг 3.5 – Модель Даних

```
@Entity(tableName = "sport_supplements")
data class SportSupplement(
   @PrimaryKey(autoGenerate = true)
   val id: Long = 0L,
   val name: String, // Назва добавки (напр., "Протеїн", "Креатин")
   val quantity: Double, // Кількість (напр., 30.0 грамів, 5.0 таблеток)
   val unit: String, // Одиниці виміру (напр., "г", "мірна ложка", "таблетка")
   val timestamp: LocalDateTime = LocalDateTime.now() // Час споживання
)
package com.your_app_name.data
import androidx.room.TypeConverter
import java.time.LocalDateTime
import java.time.format.DateTimeFormatter
class Converters {
   private val formatter = DateTimeFormatter.ISO_LOCAL_DATE_TIME
   @TypeConverter
   fun fromTimestamp(value: String?): LocalDateTime? {
      return value?.let {
         return formatter.parse(it, LocalDateTime::from)
      }
   }
   @TypeConverter
   fun dateToTimestamp(date: LocalDateTime?): String? {
      return date?.format(formatter)
   }
}
```

кінець лістингу 3.5

Для додатка моніторингу споживання DAO буде відігравати центральну роль у всіх операціях, пов’язаних з даними. Коли користувач на екрані додавання споживання вводить нові дані, ці дані будуть передані бізнес-логіці, яка, своєю чергою, викличе відповідний метод у DAO. Так само, коли головний екран потрібно оновити, щоб відобразити агреговані дані або список останніх споживань, він звернеться до бізнес-логіки, яка через DAO отримає необхідну інформацію з бази даних. DAO забезпечує, що всі ці операції є послідовними, безпечними та оптимізованими, відокремлюючи складність роботи з базою даних від інших частин програми. Таким чином, DAO є не просто паттерном проектування, а незамінним компонентом для побудови надійних, легко підтримуваних та масштабованих додатків, це наведено у лістингу 3.6.

Лістинг 3.6 – DAO (Data Access Object)

```
@Dao
interface SupplementEntryDao {
   @Insert
   suspend fun insert(entry: SupplementEntry)
   @Update
   suspend fun update(entry: SupplementEntry)
   @Delete
   suspend fun delete(entry: SupplementEntry)
   @Query("SELECT * FROM supplement_entries ORDER BY date DESC")
   fun getAllEntries(): Flow<List<SupplementEntry>>
   @Query("SELECT * FROM supplement_entries WHERE id = :id")
   suspend fun getEntryById(id: Int): SupplementEntry?
}
```

кінець лістингу 3.6

RoomDatabase це просто клас даних, який анотується @Entity і мапується на таблицю в базі даних. Кожне поле цього класу стає стовпцем таблиці, а властивість @PrimaryKey визначає первинний ключ. DAOs, як ми вже обговорювали, є інтерфейсами з анотованими методами, які визначають операції з даними. Сама Room Database – це абстрактний клас, що розширює RoomDatabase, анотується @Database, вказує, які сутності входять до цієї бази

даних, та надає абстрактні методи для отримання екземплярів DAOs. Ключова перевага Room полягає у перевірці SQL-запитів під час компіляції.

Кожне споживання, яке користувач вводить на відповідному екрані, буде представлено як сутність, яка потім буде збережена в базі даних за допомогою ConsumptionDao. Room забезпечить, що ці дані надійно зберігаються навіть після закриття додатка. Коли головний екран потребує відображення агрегованих даних або списку останніх операцій, Room дозволяє ефективно отримати цю інформацію з бази даних за допомогою простих, але потужних SQL-запитів, визначених у DAO. Інтеграція Room з іншими компонентами Android Jetpack, такими як LiveData або Kotlin Flow, також дозволяє автоматично оновлювати інтерфейс користувача щоразу, коли дані в базі даних змінюються, забезпечуючи реактивний та динамічний досвід, це наведено у лістингу 3.7.

Лістинг 3.7 – Room Database

```
@Database(entities = [SupplementEntry::class], version = 1, exportSchema = false)
@TypeConverters(DateConverter::class)
abstract class AppDatabase : RoomDatabase() {
  abstract fun supplementEntryDao(): SupplementEntryDao
  companion object {
    @Volatile
    private var INSTANCE: AppDatabase? = null
    fun getDatabase(context: Context): AppDatabase {
      return INSTANCE ?: synchronized(this) {
        val instance = Room.databaseBuilder(
          context.applicationContext,
          AppDatabase::class.java,
          "supplement_tracker_db"
        ).build()
        INSTANCE = instance
        instance
      }
    }
  }
}
```

кінець лістингу 3.7

Для додатку контролю споживання, де важливо швидко додавати дані та переглядати статистику була розроблена навігація, навігація має бути інтуїтивно зрозумілою та ефективною. Navigation Compose дозволяє нам визначати маршрути для кожного екрану та вказувати, як між ними переходити. Навігація у мобільному додатку визначає, як користувач переміщується між різними екранами та функціями, це наведено у лістингу 3.8.

Лістинг 3.8 – Navigation Compose

```
@Composable
fun SportNutritionApp() {
    val navController = rememberNavController()
    val context = LocalContext.current
    val database = AppDatabase.getDatabase(context)
    val repository = SportSupplementRepository(database.sportSupplementDao())
    val viewModel: SportSupplementViewModel = viewModel(
        factory = SportSupplementViewModel.provideFactory(repository)
    )
    NavHost(navController = navController, startDestination = "home") {
        composable("home") {
            HomeScreen(
                onNavigateToAdd = { navController.navigate("add_supplement") },
                viewModel = viewModel
            )
        }
        composable("add_supplement") {
            AddSupplementScreen(
                onNavigateBack = { navController.popBackStack() },
                viewModel = viewModel
            )
        }
    }
}
```

кінець лістингу 3.8

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Було проведено тестування Android-додатку на швидкодію:

– швидкість його завантаження;

– швидкість додавання, видалення продуктів.

На рисунку 3.4 показано швидкість з якої проект запускається на емуляторі, це показує що даний додаток є оптимізований і на не особливо сучасних смартфонах вона буде запускатися досить швидко.

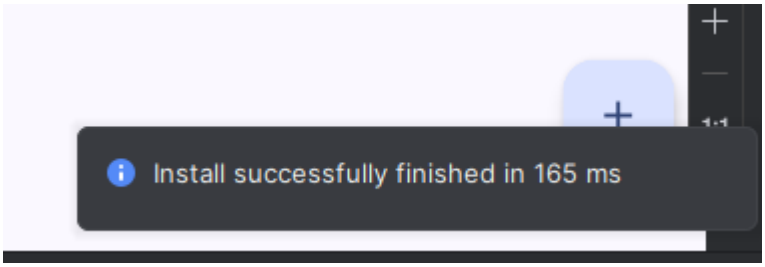

Рисунок 3.4 – Швидкість завантеження

Наступний тест показує що можна добавляти декілька продуктів, це зображено на рисунку 3.5, і якщо вони не будуть влазити в екран список буде дальше створюватися але його не буде видно, для цього зроблено що при свайпі по екрану верх список буде скролитися, також інтерфейс не перекриває кнопку для додавання нового продукту, це дає можливість зрозуміти що інтерфейс створений продумуючи різні можливі варіанти користування даним інтерфейсом програми.

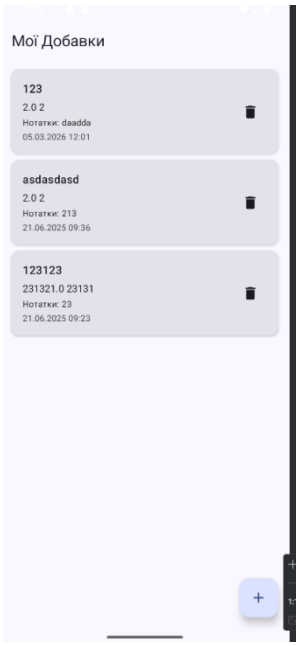

Рисунок 3.5 – Тестування кількості доданих продуктів

3.3 Розробка інсталяційного пакета

Створити «інсталяційний пакет» для Android-додатку означає зібрати його у форматі APK. Ці файли містять весь код, ресурси, асети та маніфест додатка, необхідні для встановлення на Android-пристрій.

Android Studio надає зручний інтерфейс для генерації інсталяційних пакетів. Залежно від ваших потреб, ви можете створити налагоджувальну (debug) версію для тестування або релізну версію для публікації.

Налагоджувальний APK (рис 3.6) ідеально підходить для швидкого тестування на пристрої або емуляторі, а також для обміну з іншими розробниками для внутрішнього тестування. Він не підписаний релізним ключем і не підходить для публікації в Google Play Store.

Щоб його створити потрібно зробити:

– відкрити проєкт в Android Studio;

– у верхньому меню перейдіть до Build -> Build Bundle(s) / APK(s) -> Build APK(s);

– Android Studio почне процес збірки. Це може зайняти деякий час;

– після завершення збірки можна побачити сповіщення в нижньому правому куті Android Studio: Build Analyzer successfully analyzed build. An APK is generated for the module... з посиланням locate або analyze;

– натиснувши на посилання locate, це відкриє файловий провідник (або Finder на macOS) у директорії, де збережений ваш APK файл;

– зазвичай, налагоджувальний APK знаходиться за шляхом: ваш_проєкт/app/build/outputs/apk/debug/app-debug.apk.

Можна скопіювати цей app-debug.apk на Android-пристрій і встановити його вручну (можливо, доведеться дозволити встановлення з невідомих джерел у налаштуваннях безпеки пристрою).

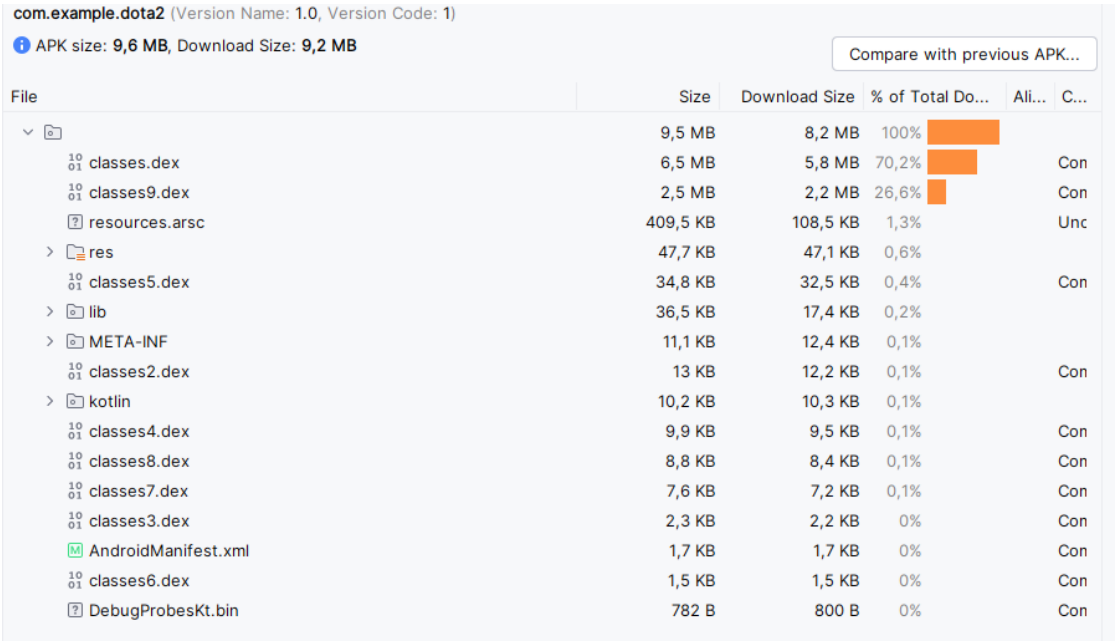

Рисунок 3.6 – Створений APK файл

Висновки до розділу 3

За результатами розробки та практичної реалізації мобільного додатку для контролю споживання спортивного харчування було успішно виконано усі етапи проєктування, реалізації та тестування програмного продукту відповідно до поставлених цілей і завдань.

На етапі практичної реалізації було створено повнофункціональний Android-додаток з використанням сучасних технологій Kotlin та Jetpack Compose. Застосовано модульну архітектуру, що дозволяє легко масштабувати функціонал та супроводжувати додаток у майбутньому. Було реалізовано основні функції: додавання та редагування продуктів спортивного харчування. Інтерфейс користувача побудовано відповідно до принципів зручності та адаптивності, що дозволяє комфортно використовувати додаток на пристроях із різними розмірами екранів.

Таким чином, реалізований мобільний додаток продемонстрував функціональну повноцінність, стабільну роботу, відповідність технічним вимогам та готовність до подальшого розгортання і використання.

ВИСНОВКИ

Розробка Android-додатку для контролю споживання спортивного харчування за допомогою Kotlin та Jetpack Compose успішно створений. Такий додаток дозволить користувачам ефективно відстежувати прийом добавок, планувати раціон та досягати поставлених спортивних цілей. Його використання сучасних технологій, таких як Kotlin для логіки та Jetpack Compose для інтерфейсу, забезпечує високу продуктивність, гнучкість та привабливий дизайн, що робить процес розробки ефективним, а сам додаток – зручним у використанні. Цей проєкт демонструє потенціал мобільних технологій у сфері здоров'я та фітнесу, надаючи користувачам потужний інструмент для самоконтролю та оптимізації їхнього харчування.

Аналіз предметної області показав, що користувачі спортивного харчування потребують інструменту для точного відстеження дозувань, часу прийому та типу добавок. Додаток дозволяє користувачам вводити інформацію про різні види спортивного харчування, такі як протеїн, креатин, вітаміни тощо, а також фіксувати кількість та час їх споживання.

Ознайомлення з сучасними підходами у розробці Android-додатків виявило доцільність використання Kotlin як основної мови програмування завдяки її лаконічності, безпечності та підтримці сучасних парадигм. Jetpack Compose визначено як ключовий інструмент для побудови інтерфейсу користувача, оскільки він пропонує декларативний підхід, що значно спрощує та прискорює розробку UI, робить його більш інтерактивним та сучасним. Інші інструменти з бібліотек Jetpack, такі як Room для роботи з базами даних, Navigation для управління переходами між екранами та ViewModel для управління станом, також розглядалися як необхідні для побудови надійної та масштабованої архітектури додатку.

На основі аналізу було визначено, що архітектура додатку є модульна та гнучка. Була використана рекомендована Google архітектура, що включає шари UI, ViewModel, Repository та Data. База даних була реалізована за допомогою

Room для зберігання інформації про спортивне харчування, включаючи назву, тип, дозування, час прийому та історію. Навігація в додатку була побудована за допомогою Jetpack Navigation, забезпечуючи інтуїтивно зрозумілі переходи між екранами.

Процес розробки Android-додатку включає створення окремих Compose функцій для кожного елемента інтерфейсу, таких як екрани введення даних, списки споживань, календар та налаштування. Логіка додатку була реалізована на Kotlin, використовуючи Coroutines для асинхронних операцій та Room для взаємодії з локальною базою даних. Особлива увага була приділена створенню інтерактивних елементів, таких як графіки споживання, адаптивний дизайн для різних розмірів екранів та плавні анімації.

Тестування додатку дало можливість зрозуміти що інтерфейс додатку інтуїтивно зрозумілий. Не виявлено багів які можуть бути як критичними так і незначними, що дає повне насолодження користуванням додатком. Також було проведено тест на швидкою додатку, які додаток пройшов і показав що він є опитимізований.

Розроблений застосунок буде корисним як для аматорів фітнесу, так і для професійних спортсменів, а також має потенціал до подальшого розвитку – зокрема, шляхом додавання хмарної синхронізації, інтеграції з носимими пристроями та розширення бази харчових продуктів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Колеснік П. Топ-10 кращих додатків на Android в 2024 році, які потрібно встановити. URL: https://www.rbc.ua/rus/styler/top-10-krashchih-program-android-2024-rotsi-1717854259.html (дата звернення: 11.03.2025).

2. Create your first Android app. Android Developers. URL: https://developer.android.com/codelabs/basic-android-kotlin-compose-first-app#0 (дата звернення: 11.03.2025).

3. Eating Buddy. URL: https://www.eatingbuddy.app (дата звернення: 13.03.2025).

4. Розробити додаток на андроїд: інструкція зі створення. FoxmindEd. URL: https://foxminded.ua/rozrobyty-dodatok-na-android/ (дата звернення: 13.03.2025).

5. GymKeeper – Workout tracker. URL: https://gymkeeper.app (дата звернення: 14.03.2025).

6. Топ 9 мов програмування для Android-додатку. URL: https://wezom.com.ua/ua/blog/prilozheniya-dlya-android (дата звернення: 16.03.2025).

7. Kotlin Programming Language. Kotlin. URL: https://kotlinlang.org (дата звернення: 18.03.2025).

8. Як виглядає специфікація вимог і як її тестувати. Онлайн-курси від компанії QATestLab. Головна сторінка. URL: https://training.qatestlab.com/blog/technical-articles/what-the-requirements-specification-looks-like-and-how-to-test-it/ (дата звернення: 29.03.2025).

9. PubMed. URL: https://pubmed.ncbi.nlm.nih.gov (дата звернення: 18.03.2025).

10. Формулювання вимог до інформаційної системи. Розробка технічного завдання. Лабораторна робота з Проектування інформаційних систем. Робота № 530540. Всеукраїнський студентський архів. URL: https://antibotan.com/file.html?work_id=530540 (дата звернення: 19.04.2025).

11. Get started with Jetpack Compose. Android Developers. URL: https://developer.android.com/develop/ui/compose/documentation (дата звернення: 22.04.2025).

12. Newest Questions. Stack Overflow. URL: https://stackoverflow.com/questions (дата звернення: 22.04.2025).

13. Coroutines guide. Kotlin Help. URL: https://kotlinlang.org/docs/coroutines-guide.html (дата звернення: 27.04.2025).

14. Jetpack Compose basics. Android Developers. URL: https://developer.android.com/codelabs/jetpack-compose-basics#0 (дата звернення: 02.05.2025).

15. Android Developers. URL: https://medium.com/androiddevelopers (дата звернення: 03.05.2025).

16. Android Weekly – Free weekly Android & Kotlin development newsletter. URL: https://www.androidweekly.net (дата звернення: 03.05.2025).

17. Kodeco. Learn iOS, Android & Flutter. URL: https://www.kodeco.com/android (дата звернення: 05.05.2025).

18. Create your first Android app. Android Developers. URL: https://developer.android.com/codelabs/basic-android-kotlin-compose-first-app#0 (дата звернення: 10.05.2025).