МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

МОБІЛЬНИЙ ДОДАТОК ДЛЯ ПЛАНУВАННЯ ХАРЧУВАННЯ З АВТОМАТИЧНИМ ПІДБОРОМ РЕЦЕПТІВ

Виконав: студент групи 2П-22
Спеціальності
121 «Інженерія програмного забезпечення»
Артем АВРАМЕНКО
Керівник:
Наталя ФАЛЬЧЕНКО

Черкаси 2026

ЗМІСТ

ВСТУП

Сучасне суспільство переживає справжню епідемію захворювань, пов'язаних із неправильним харчуванням. За статистикою Всесвітньої організації охорони здоров'я (ВООЗ), понад 1,9 мільярда дорослих у світі страждають від надмірної ваги, з яких близько 650 мільйонів мають ожиріння [3]. За даними МОЗ України, кожен третій дорослий громадянин має надмірну вагу або ожиріння, що призводить до зростання захворюваності на цукровий діабет 2-го типу, серцево-судинні хвороби та інші хронічні патології. Незбалансоване харчування є одним з ключових чинників цих проблем.

Водночас цифрові технології відкривають нові можливості для вирішення проблем, пов'язаних із контролем харчування. Мобільні застосунки дозволяють відстежувати харчування в режимі реального часу, надавати персоналізовані рекомендації та формувати здорові харчові звички. За даними аналітичного агентства Grand View Research, ринок застосунків для управління здоров'ям та харчуванням у 2024 році оцінюється в 8,5 мільярди доларів США і продовжує зростати на 15–20% щорічно [4]. Попит на персоналізовані цифрові рішення у сфері харчування постійно зростає.

Незважаючи на численні рішення, що присутні на ринку, більшість з них мають суттєві недоліки: обмежений безкоштовний функціонал, складний інтерфейс, відсутність автоматичного підбору рецептів відповідно до поточного калорійного балансу та недостатня адаптація до потреб українських користувачів. Ці проблеми зумовлюють необхідність розробки нового рішення.

Актуальність теми. Зростання захворюваності, пов'язаної з неправильним харчуванням, в поєднанні з динамічним розвитком ринку мобільних застосунків для управління здоров'ям визначають актуальність розробки доступного та зручного інструменту для персоналізованого планування харчування. Існуючі рішення не задовольняють повною мірою потреби користувачів через платний функціонал, складний інтерфейс та відсутність автоматичного підбору рецептів відповідно до поточного калорійного балансу. Розробка кросплатформеного

мобільного застосунку з автоматичним підбором рецептів на основі індивідуальних фізіологічних характеристик є актуальним завданням, що відповідає сучасним потребам суспільства у доступних цифрових інструментах для підтримки здорового способу життя.

Метою кваліфікаційної роботи є розробка мобільного застосунку для персоналізованого планування харчування з автоматичним підбором рецептів на основі індивідуальних фізіологічних характеристик користувача та поточного добового балансу калорій.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати предметну область та визначити потреби цільової аудиторії у сфері мобільних застосунків для планування харчування;
– провести порівняльний огляд існуючих аналогів та визначити їх переваги і недоліки;
– дослідити та обґрунтувати вибір технологічного стеку для кросплатформеної розробки мобільного застосунку;
– дослідити математичні методи розрахунку індивідуальних добових норм калорій та макронутрієнтів (формула Міффліна-Сан Жеора);
– розробити архітектуру застосунку;
– сформулювати функціональні та нефункціональні вимоги до застосунку;
– реалізувати повнофункціональний мобільний застосунок відповідно до сформульованих вимог;
– провести комплексне тестування застосунку та оцінити відповідність результатів вимогам.

Об'єктом дослідження є система планування харчування з використанням мобільних цифрових технологій та хмарних сервісів.

Предметом дослідження є методи та засоби розробки кросплатформеного мобільного застосунку для персоналізованого планування харчування з автоматичним підбором рецептів на основі поточного калорійного балансу.

Практична значущість роботи полягає у створенні безкоштовного мобільного застосунку для планування харчування, доступного на платформах

iOS та Web. Застосунок може використовуватися користувачами, які потребують контролю за харчуванням з метою схуднення, підтримки ваги або набору м'язової маси; людьми з хронічними захворюваннями, яким необхідне дотримання певного калорійного та нутрієнтного балансу.

РОЗДІЛ 1 ОГЛЯД ТА АНАЛІЗ МЕТОДІВ І ЗАСОБІВ РОЗРОБКИ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ПЛАНУВАННЯ ХАРЧУВАННЯ

1.1 Предметна область та аналіз потреб цільової аудиторії

Предметна область кваліфікаційної роботи охоплює персоналізоване харчування, мобільні технології та хмарні обчислення. Харчування є фундаментальним аспектом людського здоров'я: правильний баланс нутрієнтів сприяє підтриманню нормальної маси тіла, профілактиці хронічних захворювань та підвищенню якості життя.

Добова потреба людини в калоріях залежить від статі, віку, зросту, маси тіла та рівня фізичної активності. Для підтримки маси тіла кількість споживаних калорій має дорівнювати кількості витрачених. Профіцит калорій призводить до набору ваги, дефіцит — до схуднення.

Цільова аудиторія застосунку який моделюється охоплює декілька категорій:

– Особи з надмірною вагою або ожирінням, що прагнуть знизити вагу через контроль калорійності харчування;
– Спортсмени та люди з активним способом життя, що потребують точного обліку нутрієнтів для підтримки фізичних результатів;
– Особи, що відновлюються після захворювань та потребують дотримання призначеної дієти;
– Студенти та молодь, що формують здорові харчові звички та вчаться раціонально харчуватися;
– Батьки, що планують харчування для своєї сім'ї та дітей.

Аналіз потреб цільової аудиторії виявив такі ключові вимоги до застосунку:

– Простота та інтуїтивність — користувачі не готові витрачати більше 2-3 хвилин на введення денних записів харчування;
– Персоналізація — норми мають бути розраховані індивідуально, а не за загальними таблицями;

– Рекомендації рецептів — користувачі потребують підказок що саме приготувати, виходячи з доступного залишку калорій;

– Безкоштовний доступ — більшість потенційних користувачів не готові платити підписку за базовий функціонал;

– Синхронізація між пристроями — дані мають бути доступні з телефону та комп'ютера.

Для глибшого розуміння потреб цільової аудиторії розроблено три персони користувачів NutriPlan. Персона — це узагальнений портрет типового представника цільової аудиторії, що допомагає краще зрозуміти їхні цілі, проблеми та поведінкові патерни. Характеристики персони 1 представлені в Таблиці 1.1

Таблиця 1.1 – Персона 1: Студент, що контролює вагу

Характеристика	Опис
Ім'я та вік	Максим, 20 років, студент технічного коледжу
Спосіб життя	Сидяча навчальна діяльність, іноді ходить до спортзалу 1–2 рази на тиждень
Ціль	Схуднути на 8 кг до літа без суворих дієт
Проблема	Не знає що приготувати у межах калорійного ліміту, часто переїдає ввечері
Технічний рівень	Висока технічна грамотність, активно користується смартфоном
Очікування	Простий підрахунок калорій, підказки що поїсти, безкоштовний доступ
Болючі точки	Платні підписки відштовхують; незручно шукати що приготувати окремо від трекера

Характеристики персони 2 представлені в Таблиці 1.2

Таблиця 1.2 – Персона 2: Молода фахівець, що веде здоровий спосіб життя

Характеристика	Опис
Ім'я та вік	Олена, 27 років, маркетолог
Спосіб життя	Офісна робота, тренується 3 рази на тиждень, слідкує за харчуванням
Ціль	Підтримувати поточну вагу та збалансувати макронутрієнти
Проблема	Витрачає багато часу на пошук рецептів відповідно до денного залишку калорій

Продовження таблиці 1.2

Технічний рівень	Середня технічна грамотність, використовує кілька застосунків для здоров'я
Очікування	Автоматичні рекомендації рецептів, синхронізація між телефоном і ПК
Болючі точки	Існуючі застосунки або надто прості, або надто складні; немає автопідбору рецептів

Характеристики персони 3 представлені в Таблиці 1.3

Таблиця 1.3 – Персона 3: Людина з надмірною вагою, що починає шлях до здоров'я

Характеристика	Опис
Ім'я та вік	Василь, 42 роки, бухгалтер
Спосіб життя	Малорухливий спосіб життя, нерегулярне харчування, практично не займається спортом
Ціль	Знизити вагу на 15 кг за рекомендацією лікаря
Проблема	Не розуміється на калоріях та макросах, потребує максимально простого рішення
Технічний рівень	Низька технічна грамотність, вперше використовує застосунок для харчування
Очікування	Простий інтерфейс, зрозумілі підказки, щоб система «думала» за нього
Болючі точки	Складні інтерфейси лякають; не розуміє різниці між білками/жирами/вуглеводами

1.2 Огляд аналогів застосунку для планування харчування

Для проведення детального порівняльного аналізу обрано три найпоширеніші мобільні застосунки для планування харчування. Критерії вибору аналогів для дослідження наведено в таблиці 1.4.

Таблиця 1.4 – Критерії відбору аналогів для порівняльного аналізу

Критерій відбору	MyFitnessPal	Lifesum	FatSecret	Відповідність
Активних користувачів 5+ млн	200 млн	50 млн	10 млн	Усі відповідають

Продовження таблиці 1.4

Топ-10 App Store (Health)	✓	✓	✓	Усі відповідають
Підрахунок калорій	✓	✓	✓	Усі відповідають
Підрахунок макронутрієнтів	✓	✓	✓	Усі відповідають
Підтримка iOS та Android	✓	✓	✓	Усі відповідають

MyFitnessPal — найпопулярніший у світі застосунок для відстеження харчування та фізичної активності [16]. Заснований у 2005 році Майком Лі та Алі Хайіттом у Сан-Франциско, США. У 2015 році придбаний компанією Under Armour за 475 мільйонів доларів США. У 2020 році відділений в окрему компанію. Станом на 2024 рік нараховує понад 200 мільйонів зареєстрованих користувачів та 14 мільйонів активних щомісячних користувачів у 125 країнах.

Технічно застосунок доступний на платформах iOS, Android та Web. Основна монетизація здійснюється через преміум-підписку MyFitnessPal Premium вартістю 9,99 USD/місяць або 79,99 USD/рік. Безкоштовна версія містить базовий функціонал — ведення харчового щоденника та відстеження калорій та трьох макронутрієнтів.

База даних продуктів харчування MyFitnessPal є найбільшою серед конкурентів і нараховує понад 14 мільйонів найменувань продуктів з усього світу. Однак якість даних неоднорідна — значна частина записів додана самими користувачами і може містити неточності чи помилки. Застосунок підтримує сканування штрих-кодів для миттєвого пошуку продуктів, що суттєво прискорює введення даних. Функція розпізнавання їжі за фотографією доступна лише в преміум-версії.

Функціонал відстеження фізичної активності є сильною стороною MyFitnessPal. Застосунок підтримує інтеграцію з понад 50 платформами та пристроями: Apple Health, Google Fit, Fitbit, Garmin, Polar, Samsung Health. Витрати калорій при тренуваннях автоматично враховуються у добовому балансі. База вправ містить понад 500 видів фізичної активності.

Соціальні функції включають: стрічку активності з публікаціями друзів, рейтинги активності, групи підтримки. Це підвищує мотивацію та залученість користувачів. Преміум-версія додає детальну аналітику по мікронутрієнтах (35+ вітамінів та мінералів), доступ до бібліотеки рецептів та функцію планування харчування наперед.

Аналіз інтерфейсу MyFitnessPal виявив наступне: головний екран перевантажений інформацією — одночасно відображаються кільцева діаграма калорій, показники макронутрієнтів, прийоми їжі, активність та вода. Для нового користувача це може бути складно сприйняти. Процес додавання прийому їжі вимагає в середньому 4–6 кроків навігації, тоді як у конкурентів — 2–3 кроки.

Ключові переваги MyFitnessPal: найбільша база даних продуктів, широка інтеграція з пристроями та платформами, розвинена соціальна функція, глибока аналітика в преміумі. Ключові недоліки: складний інтерфейс, більшість корисних функцій платні (9,99 USD/міс), відсутність автоматичного підбору рецептів, неактуальна база для українського ринку.

Lifesum — шведський застосунок для здорового харчування та контролю ваги [17], заснований у Стокгольмі у 2013 році командою Маркуса Вестберга та Хенріка Тотте. Застосунок позиціонує себе не як простий лічильник калорій, а як повноцінний персональний помічник у формуванні здорових харчових звичок. Станом на 2024 рік аудиторія застосунку становить понад 50 мільйонів користувачів у 100+ країнах.

Ключовою особливістю Lifesum є власна система оцінки якості харчування — «Life Score» від 1 до 10 балів. Алгоритм оцінює кожен продукт або прийом їжі за нутрієнтним складом: бонусні бали за білки та клітковину, штрафні бали за насичені жири, цукор та натрій. Щоразу при додаванні продукту відображається оцінка корисності, що мотивує користувачів свідомо підходити до вибору їжі.

Функція дієтичних планів є ключовою перевагою Lifesum над конкурентами. Застосунок пропонує більше 20 готових науково обґрунтованих дієтичних планів: кетогенна дієта, середземноморська, веганська, Whole30,

Nordic Diet, Anti-inflammatory Diet та інші. Кожен план містить опис принципів харчування, список рекомендованих та заборонених продуктів, набір рецептів та тижневе меню.

Водний баланс відстежується через інтуїтивний трекер з можливістю встановлення персональної норми на основі маси тіла або вручну. Застосунок надсилає нагадування про необхідність пити воду у налаштовані проміжки часу. Результати відображаються у вигляді кільцевої діаграми на головному екрані.

Інтеграція Lifesum з Apple Health та Google Fit дозволяє автоматично враховувати фізичну активність та відображати збалансований добовий баланс калорій. Застосунок також підтримує голосове введення продуктів через Siri та Google Assistant.

Аналіз інтерфейсу виявив, що Lifesum має найсучасніший та найчистіший дизайн серед розглянутих конкурентів. Навігація інтуїтивна, екрани не перевантажені. Головний екран відображає денні цілі у вигляді кільцевих діаграм та короткого резюме.

Ключові переваги Lifesum: сучасний UI/UX, унікальна система Life Score, широкий вибір дієтичних планів, детальне відстеження нутрієнтів. Ключові недоліки: більшість дієтичних планів доступні лише в преміумі (від 4,99 USD/міс), відсутність автопідбору рецептів за залишком калорій, обмежена база продуктів для України, складна навігація для нових користувачів у преміум-функціях.

FatSecret — один з найстаріших застосунків для відстеження харчування, заснований у 2007 році [18]. Незважаючи на відносно застарілий дизайн порівняно з сучасними конкурентами, FatSecret залишається популярним завдяки ключовій перевазі — повністю безкоштовному доступу до всього функціоналу без прихованих платних функцій. База даних нараховує понад 10 мільйонів продуктів, включаючи страви з ресторанів та готові продукти.

Унікальною функцією FatSecret є можливість фотографування їжі та автоматичного розпізнавання продуктів за допомогою алгоритмів комп'ютерного зору (технологія Image Recognition). Користувач фотографує

страву, система автоматично визначає склад та розраховує калорійність. Ця функція є безкоштовною, тоді як у конкурентів вона доступна лише в платній версії.

Харчовий щоденник FatSecret надає детальний аналіз спожитих нутрієнтів: крім основних (калорії, білки, жири, вуглеводи), відображаються клітковина, цукор, холестерин, натрій, калій, а також вітаміни A, C, D, E, K, групи B та мінерали (кальцій, залізо, магній, цинк та інші). Це особливо корисно для людей з хронічними захворюваннями.

Функція відстеження ваги дозволяє вести детальний графік змін маси тіла та ІМТ протягом часу. Застосунок розраховує динаміку змін, прогнозує дату досягнення цільової ваги та відображає порівняльну статистику. Можливість введення замірів тіла (обхват талії, стегон тощо) дозволяє відстежувати прогрес навіть при незмінній масі тіла.

Соціальна мережа FatSecret — унікальна особливість, відсутня у конкурентів у такому форматі. Застосунок має власний форум з тематичними підрозділами: рецепти, дієти, мотивація, питання та відповіді. Публічні щоденники харчування дозволяють ділитися своїм прогресом та отримувати підтримку спільноти.

Ключові переваги FatSecret: повністю безкоштовний без обмежень, детальний аналіз мікронутрієнтів, розпізнавання їжі за фото, активна соціальна спільнота. Ключові недоліки: застарілий дизайн інтерфейсу, відсутність персоналізованих рекомендацій рецептів, немає дієтичних планів, обмежена аналітика трендів.

На основі детального огляду трьох застосунків складено комплексну порівняльну таблицю (табл. 1.5).

Таблиця 1.5 – Порівняльний аналіз застосунків для планування харчування

Критерій	MyFitnessPal	Lifesum	FatSecret
Автопідбір рецептів за залишком калорій	✗	✗	✗

Продовження таблиці 1.5

Розрахунок TDEE при реєстрації	✓	✓	✗
Хмарна синхронізація між пристроями	✓	✓	✓
Повністю безкоштовний доступ	✗	✗	✓
Трекер споживання води	✓	✓	✗
Сучасний дизайн інтерфейсу	✓	✓	✗
Тижнева статистика харчування	✓	✓	✓
Типи прийомів їжі (сн/об/вечеря)	✓	✓	✓
Підтримка iOS	✓	✓	✓
Веб-версія застосунку	✓	✗	✗
Персоналізований онбординг	✓	✓	✗
Відстеження обраних рецептів	✓	✓	✗
Вартість повного доступу	9,99 USD/міс	4,99 USD/міс	Безкоштовно
Відповідність вимогам (із 13)	10/13	10/13	8/13

Аналіз даних таблиці 1.5 показує, що жоден із розглянутих застосунків не реалізує автоматичний підбір рецептів на основі залишкового добового калорійного балансу, що є ключовою функціональною прогалиною наявних рішень. MyFitnessPal та Lifesum демонструють найвищу відповідність вимогам, проте обидва є платними, що обмежує доступність для широкої аудиторії. FatSecret є єдиним повністю безкоштовним рішенням, однак поступається конкурентам за функціональністю та дизайном інтерфейсу.

1.3 Постановка задачі на розробку мобільного застосунку

На основі аналізу предметної області та порівняльного огляду конкурентів сформульовано проблему, що існуючі безкоштовні рішення не мають автоматичного підбору рецептів за поточним калорійним балансом та

персоналізованого розрахунку норм, а платні — недоступні для широкої аудиторії. Тому виникла необхідність у розробці застосунку який вирішує ці проблеми.

Застосунок повинен забезпечувати такі функції:

– безкоштовний повний доступ до всіх функцій;

– персоналізований розрахунок добових норм калорій та макронутрієнтів за формулою Міффліна-Сан Жеора;

– автоматичний підбір рецептів на основі залишку добових калорій;

– зручне ведення щоденного логу харчування за типами прийомів їжі;

– відстеження водного балансу;

– тижневу статистику харчування з SVG-графіком;

– хмарну синхронізацію даних між пристроями;

– сучасний та інтуїтивний інтерфейс.

Функціональні вимоги до застосунку NutriPlan систематизовано у таблиці 1.6.

Таблиця 1.6 – Функціональні вимоги до застосунку NutriPlan

№	Модуль	Вимога	Пріоритет
1	Авторизація	Реєстрація через email та пароль	Must have
2	Авторизація	Вхід до існуючого акаунту	Must have
3	Авторизація	Скидання пароля через email	Should have
4	Авторизація	Збереження сесії між запусками	Must have
5	Онбординг	Збір персональних даних (5 кроків)	Must have
6	Онбординг	Розрахунок TDEE (Міффлін)	Must have
7	Дашборд	SVG-кільце калорій	Must have
8	Дашборд	Прогрес-бари макронутрієнтів	Must have
9	Дашборд	Трекер води (8 склянок)	Must have
10	Дашборд	Автопідбір рецептів за залишком	Must have
11	Рецепти	Каталог рецептів з пошуком	Must have
12	Рецепти	Фільтрація за категоріями	Should have

Продовження таблиці1.6

13	Рецепти	Деталізований екран рецепту	Must have
14	Лог	Додавання страв за типами прийомів	Must have
15	Лог	Видалення записів з логу	Must have
16	Обране	Збереження улюблених рецептів	Should have
17	Статистика	Тижневий SVG-графік калорій	Should have
18	Статистика	Картки середніх показників	Should have
19	Налаштування	Редагування профілю	Should have
20	Налаштування	Вихід з акаунту	Must have

У першому розділі проведено комплексний аналіз предметної області та ринку мобільних застосунків для планування харчування. Визначено цільову аудиторію NutriPlan та її ключові потреби: персоналізація, автоматичні рекомендації, простота, безкоштовний доступ.

Детально досліджено три провідні застосунки. MyFitnessPal має найбільшу базу продуктів та широку інтеграцію, проте перевантажений інтерфейс та обмежений безкоштовний функціонал. Lifesum вирізняється сучасним дизайном та системою Life Score, але ключові функції платні. FatSecret є повністю безкоштовним та має детальний аналіз нутрієнтів, проте застарілий дизайн та відсутність персоналізованих рекомендацій. Іак

Порівняльний аналіз за 14 критеріями підтвердив, що жоден із застосунків не реалізує автоматичний підбір рецептів на основі залишку калорій — ключову функцію застосунку NutriPlan. Сформульовано задачу на розробку та визначено 20 функціональних вимог.

РОЗДІЛ 2 РОЗРОБКА ФУНКЦІОНАЛЬНОЇ МОДЕЛІ ЗАСТОСУНКУ

2.1 Обрунтування способів реалізації застосунку NutriPlan

Перед початком розробки застосунку проведено систематичне дослідження сучасних технологій та підходів до розробки мобільних застосунків. Дослідження охоплює вибір парадигми розробки, фреймворку, мови програмування, рішення для управління станом та хмарного бекенду.

Існує три основні парадигми розробки мобільних застосунків, кожна з яких має свої переваги та недоліки.

Нативна розробка передбачає створення окремих застосунків для кожної платформи з використанням офіційних мов та інструментів: Swift або Objective-C для iOS, Kotlin або Java для Android. Переваги нативної розробки: максимальна продуктивність, повний доступ до API платформи, кращий UX та інтеграція з ОС. Недоліки: подвійні витрати на розробку та підтримку, необхідність двох команд розробників, складніша синхронізація функціоналу між платформами.

Гібридна розробка базується на вебтехнологіях (HTML, CSS, JavaScript), обгорнутих у нативну оболонку через WebView. Представники: Ionic, Cordova, PhoneGap. Переваги: одна кодова база, знайомі вебтехнології, простота розробки. Суттєві недоліки: значне відставання від нативної продуктивності (рендеринг у WebView є повільнішим), обмежений доступ до API платформи, помітна відмінність від нативного UX.

Кросплатформена розробка — оптимальний баланс між витратами та якістю. Провідні рішення: React Native та Flutter. Обидва рендерять нативні UI-компоненти (або власний рушій рендерингу у Flutter), що забезпечує продуктивність, близьку до нативної. Одна кодова база значно знижує витрати на розробку та підтримку.

Порівняльний аналіз трьох підходів наведено у таблиці 2.1.

Таблиця 2.1 – Порівняльний аналіз підходів до розробки мобільних застосунків

Критерій	Нативна	Гібридна	Кросплатформена
Продуктивність	Максимальна	Низька	Висока
Кодова база	Окрема для кожної ОС	Спільна	Спільна (95%+)
Час розробки	Довгий	Короткий	Середній
Витрати	Високі	Низькі	Середні
Доступ до API платформи	Повний	Обмежений	Широкий
UX якість	Максимальна	Знижена	Висока
Підтримка веб	✗	✓	✓ (React Native)

Для проєкту NutriPlan обрано кросплатформену розробку як оптимальний варіант: вона забезпечує достатню продуктивність, підтримку iOS та Web з єдиної кодової бази та помірні витрати на розробку.

React Native — відкритий фреймворк для розробки мобільних застосунків, створений Meta у 2015 році [5]. Ключова перевага React Native перед іншими кросплатформеними рішеннями полягає в тому, що він рендерить нативні UI-компоненти операційної системи, а не вебкомпоненти в WebView. Це забезпечує продуктивність, близьку до нативних застосунків, при значно менших витратах на розробку.

За даними Stack Overflow Developer Survey 2024 [19], React Native використовують 9,2% розробників мобільних застосунків, що робить його другим за популярністю фреймворком після Flutter (9,7%). Проте React Native має суттєву перевагу — можливість повторного використання коду з React для веб через бібліотеку react-native-web, що особливо важливо для NutriPlan.

Expo — платформа поверх React Native, розроблена компанією Expo Inc. [6]. Expo спрощує розробку через надання готового набору бібліотек, сервісів та інструментів, що виключають необхідність налаштування нативного середовища Xcode та Android Studio. Expo SDK 56 включає ключові бібліотеки:

– expo-router v4 — файлова маршрутизація в стилі Next.js (маршрути описуються ієрархією файлів);

– expo-haptics — тактильний відгук для iOS та Android без нативного коду;

– expo-notifications — push-сповіщення з підтримкою всіх платформ;

– expo-blur — ефект розмиття для нативних платформ;

– expo-status-bar — управління статус-баром;

– EAS Build — хмарна збірка застосунку для App Store та Google Play.

Flutter — конкурент React Native від Google, що використовує мову Dart та власний рушій рендерингу Impeller. Flutter забезпечує ідентичний зовнішній вигляд на всіх платформах та вищу продуктивність, однак для NutriPlan він не обраний через: відсутність TypeScript (Dart є окремою мовою з іншим синтаксисом), менший вибір сторонніх бібліотек, складніший перехід для розробників з JavaScript-фону, та обмеженішу підтримку веб-платформи.

Порівняння React Native та Flutter наведено в таблиці 2.2.

Таблиця 2.2 – Порівняння React Native та Flutter

Критерій	React Native	Flutter
Мова програмування	JavaScript / TypeScript	Dart
Рендеринг UI	Нативні компоненти ОС	Власний рушій (Impeller)
Продуктивність	Висока	Дуже висока
Підтримка Web	✓ (react-native-web)	Обмежена
Розмір екосистеми NPM	2 млн+ пакетів	30 тис.+ пакетів
Популярність (SO 2024)	9,2%	9,7%
Поріг входу (JS-досвід)	Низький	Середній (нова мова)

Обґрунтування вибору React Native з Expo: найбільша JavaScript-спільнота, TypeScript підтримка з коробки, можливість повторного використання логіки між мобільним та веб-застосунками, багата екосистема npm-пакетів, офіційна підтримка веб-платформи.

TypeScript — строго типізована надмножина JavaScript, розроблена та підтримувана Microsoft з 2012 року [12]. TypeScript додає до JavaScript систему статичних типів, що дозволяє виявляти помилки на етапі компіляції, а не під час виконання програми. Це критично важливо для великих проєктів.

За даними State of JS 2023, TypeScript використовують 78% JavaScript-розробників — найвищий показник серед всіх JavaScript-надмножин і синтаксичних розширень. TypeScript є стандартом де-факто для React Native розробки в 2024 році.

Переваги TypeScript для проєкту NutriPlan:

– Статична типізація сутностей — чіткі типи для UserProfile, Recipe, MealEntry, DayStats виключають помилки невідповідності типів;
– Інтелектуальне автодоповнення в VS Code прискорює розробку та знижує кількість помилок введення;
– Строгий режим (strict: true) у tsconfig.json забезпечує максимальний контроль якості;
– Перелічення (enum) для MealType та Goal підвищують читабельність коду;
– Дженерики для типізованих функцій стору Zustand забезпечують коректну типізацію стану;
– Рефакторинг коду стає безпечнішим — компілятор виявляє всі місця, що потребують оновлення.

Firebase — платформа від Google для розробки мобільних та веб-застосунків, що надає готовий набір хмарних BaaS-сервісів (Backend as a Service). Для NutriPlan використано Firebase Authentication та Cloud Firestore. Вибір Firebase обумовлений такими факторами:

– Безкоштовний тарифний план Spark: 50 000 читань та 20 000 записів на день, 1 ГБ сховища — достатньо для навчального проєкту;
– Офіційна підтримка React Native SDK з документацією та прикладами;
– Реальний час синхронізації — зміни автоматично поширюються на всі підключені клієнти;
– Вбудована система безпеки — Firebase Security Rules контролюють доступ до даних [10];
– Простота налаштування — проєкт готовий до роботи за 15–20 хвилин.

Firebase Authentication надає готову систему авторизації з підтримкою email/пароля, Google Sign-In, Apple Sign-In, Facebook та інших провайдерів [8].

Система автоматично обробляє: хешування паролів (bcrypt), токени JWT для авторизованих запитів, відновлення пароля через email, захист від brute force атак.

Cloud Firestore — документоорієнтована NoSQL база даних із реальним часом синхронізації [9]. Структура даних у Firestore: колекції містять документи, документи містять поля. Ієрархічна структура дозволяє логічно організовувати дані та ефективно управляти правами доступу.

Структура бази даних NutriPlan у Firestore:

– users/{uid}/profile — документ з профілем: name, gender, age, height, weight, goal, activity, goalKcal, goalProtein, goalCarbs, goalFat;

– users/{uid}/meals/{date}/{mealId} — документ прийому їжі: mealType, recipeId, grams, createdAt;

– users/{uid}/favorites — масив ідентифікаторів улюблених рецептів;

– users/{uid}/water/{date} — кількість випитих склянок за дату.

При виборі типу бази даних для NutriPlan розглядалися два підходи: реляційні (SQL) та документоорієнтовані (NoSQL) бази даних. Обґрунтування вибору Cloud Firestore (NoSQL) наведено у порівняльній таблиці 2.3.

Таблиця 2.3 – Порівняльний аналіз SQL та NoSQL для проєкту NutriPlan

Критерій	SQL (PostgreSQL/MySQL)	NoSQL — Cloud Firestore
Структура даних	Жорсткі таблиці зі схемою	Гнучкі JSON-документи без фіксованої схеми
Масштабованість	Вертикальна (потребує більшого сервера)	Горизонтальна (автоматично від Google)
Реальний час	Потребує додаткових рішень (WebSocket)	Вбудований real-time listener (onSnapshot)
Запити	SQL — потужний, складні JOIN-запити	Простіші запити, немає JOIN
Інтеграція з React Native	Потребує окремого API-сервера	Офіційний SDK, пряме підключення
Безпека	Керується на рівні сервера	Firebase Security Rules — декларативно
Вартість (навч. проєкт)	Потребує хостингу сервера (~5 USD/міс)	Безкоштовний Spark план (0 USD)

Продовження таблиці 2.3

Час налаштування	2–4 години (сервер + схема + API)	15–20 хвилин (Firebase Console)
Офлайн-підтримка	Потребує додаткових рішень	Вбудований офлайн-кеш
Придатність для NutriPlan	Надмірно складний для даного проєкту	Оптимальний вибір ✓

Структура даних NutriPlan є ієрархічною та об'єктно-орієнтованою: профіль користувача містить вкладені об'єкти норм, прийоми їжі прив'язані до дати та типу. Така структура природно відображається на документоорієнтовану модель Firestore без необхідності JOIN-запитів. SQL-підхід потребував би щонайменше 4 нормалізованих таблиць (users, profiles, meals, favorites) та складних JOIN при кожному завантаженні дашборду.

Додатково, відсутність необхідності у власному API-сервері суттєво спрощує архітектуру: застосунок спілкується з Firestore безпосередньо через SDK, що скорочує кількість рівнів у системі та зменшує ймовірність помилок.

Zustand — легковажна бібліотека для управління глобальним станом React/React Native застосунків, розроблена командою Poimandres [11]. Розмір бібліотеки — лише 1,3 KB (мінімізований + gzip), що робить її найлегшою у своїй категорії.

Порівняння Zustand з альтернативами наведено у таблиці 2.4.

Таблиця 2.4 – Порівняння бібліотек управління станом

Критерій	Zustand	Redux	MobX	Jotai
Розмір (gzip)	1,3 KB	7,6 KB	19 KB	3,5 KB
Boilerplate код	Мінімальний	Великий	Середній	Мінімальний
Крива навчання	Низька	Висока	Середня	Низька
TypeScript підтримка	Відмінна	Добра	Добра	Відмінна
DevTools	Є	Redux DevTools	MobX DevTools	Jotai DevTools
Async дії	Вбудовані	Middleware	Вбудовані	Вбудовані

В архітектурі NutriPlan використано два незалежні стори: useAuthStore (стан авторизації) та useStore (основний стор застосунку). Такий поділ забезпечує чітке розмежування відповідальностей та спрощує тестування.

Для персоналізованого розрахунку добових норм калорій та макронутрієнтів в NutriPlan реалізовано формулу Міффліна-Сан Жеора [1], що є одною із найточніших серед існуючих за результатами систематичного огляду Американської дієтичної асоціації (ADA, 2005) [2].

Порівняльна точність основних формул розрахунку BMR наведена у таблиці 2.5.

Таблиця 2.5 – Порівняльна точність формул розрахунку BMR

Формула	Рік	Точність	Примітки
Міффліна-Сан Жеора	1990	82%	Найточніша для сучасної популяції
Харріса-Бенедикта (перегл.)	1984	78%	Друга за точністю
Харріса-Бенедикта (оригін.)	1919	70%	Застаріла, переоцінює BMR
Оуена	1987	75%	Недооцінює для активних людей

Розрахунок базового обміну речовин (BMR — Basal Metabolic Rate) за формулою Міффліна-Сан Жеора:

$$\textit{Для чоловіків: } BMR = 10 \times m + 6{,}25 \times h - 5 \times a + 5 \qquad (2.1)$$

$$\textit{Для жінок: } BMR = 10 \times m + 6{,}25 \times h - 5 \times a - 161 \qquad (2.2)$$

де m — маса тіла (кг), h — зріст (см), a — вік (роки).

Загальні добові витрати енергії (TDEE = Total Daily Energy Expenditure) розраховуються як добуток BMR та коефіцієнту фізичної активності (PAL). Коефіцієнти наведено у таблиці 2.6.

Таблиця 2.6 – Коефіцієнти рівня фізичної активності (PAL)

Рівень активності	PAL	Характеристика та приклади
Низька (sedentary)	1,200	Сидяча робота (офіс, програміст), відсутність тренувань
Помірна (lightly active)	1,375	Легкі прогулянки, тренування 1–3 рази/тиждень
Висока (moderately active)	1,550	Помірні тренування 3–5 разів/тиждень
Дуже висока (very active)	1,725	Інтенсивні щоденні тренування, фізична робота

Корекція TDEE відповідно до мети: схуднення — дефіцит 500 ккал/день (безпечний темп -0,5 кг/тиждень згідно з рекомендаціями ВООЗ), підтримка ваги — без корекції, набір м'язової маси — профіцит 400 ккал/день.

Розподіл калорій по макронутрієнтах ґрунтується на рекомендаціях ВООЗ та Американської дієтичної асоціації: білки — 25% (4 ккал/г, підтримує м'язову масу та почуття ситості), вуглеводи — 45% (4 ккал/г, основне джерело енергії), жири — 30% (9 ккал/г, гормональна функція та засвоєння вітамінів).

2.2 Моделювання предметної області

Для формального опису структури та поведінки застосунку NutriPlan використано нотацію UML. Побудовано чотири типи діаграм: варіантів використання, класів, послідовності та навігації.

Діаграма варіантів використання (Use Case Diagram) описує функціональні можливості системи з точки зору кінцевого користувача. Виділено два актори: Незареєстрований користувач та Авторизований користувач.

Незареєстрований користувач має доступ до: реєстрації нового акаунту (UC-01) та входу до існуючого акаунту (UC-02). Авторизований користувач після успішної авторизації отримує повний доступ до 10 варіантів використання:

– UC-03: Скидання пароля через email;

– UC-04: Проходження онбордингу (першовхід);

– UC-05: Перегляд головного дашборду;

– UC-06: Управління водним балансом;

– UC-07: Перегляд та пошук рецептів;

– UC-08: Додавання страви до щоденного логу;

– UC-09: Перегляд тижневої статистики;

– UC-10: Управління обраними рецептами;

– UC-11: Редагування профілю та перерахунок норм;

– UC-12: Вихід з акаунту.

Між варіантами використання існують такі зв'язки: UC-07 «include» UC-08 (пошук рецептів включає додавання до логу), UC-04 «include» UC-06 (онбординг включає розрахунок норм), UC-11 «extend» UC-06 (редагування профілю розширює розрахунок норм).

Діаграма класів відображає статичну структуру системи: класи, їх атрибути, методи та зв'язки між ними. Основні класи NutriPlan:

Клас UserProfile описує профіль користувача та містить атрибути: uid: string, name: string, gender: Gender (перелік), age: number, height: number (см), weight: number (кг), goal: Goal (lose|maintain|gain), activity: Activity (low|medium|high|very_high), goalKcal: number, goalProtein: number, goalCarbs: number, goalFat: number. Методи: updateProfile(), recalculateNorms().

Клас Recipe описує рецепт страви: id: number, name: string, category: string, kcal: number, protein: number, carbs: number, fat: number, timeMin: number, photo: string, description: string, ingredients: string[]. Методи: getCaloriesPerServing(), matchesCalorieLimit().

Клас MealEntry описує запис прийому їжі: id: string, mealType: MealType (breakfast|lunch|dinner|snack), recipeId: number, grams: number, date: string, createdAt: Timestamp. Методи: getCalories(), getMacros().

Клас DayStats містить агреговану статистику за день: date: string, kcal: number, protein: number, carbs: number, fat: number, water: number. Методи: calculateDeficit(), getWaterPercentage().

Зв'язки між класами: UserProfile агрегує MealEntry (1 до *), Recipe асоціюється з MealEntry через recipeId, DayStats обчислюється на основі колекції MealEntry.

Описано детальну послідовність взаємодії компонентів при ключовому сценарії — додаванні рецепту до логу харчування з головного екрану.

1. Користувач натискає кнопку «+» поруч з рекомендованим рецептом на HomeScreen.
2. HomeScreen викликає метод addMeal(recipeId, date, mealType) з useStore.
3. useStore формує об'єкт MealEntry та додає його до локального масиву meals.
4. Паралельно useStore викликає addDoc() Firestore API для синхронізації запису.
5. React реактивно перераховує useMemo-залежності: totalKcal, totalProtein, totalCarbs, totalFat.
6. CalorieRing отримує нові props та рендерить оновлений SVG з анімацією.
7. MacroBars оновлюють відображення прогрес-барів.
8. RecommendedRecipes перефільтровується на основі нового залишку калорій (remaining = goalKcal - totalKcal).
9. Якщо totalKcal >= goalKcal, запускається анімація конфеті та відображається повідомлення.
10. Firestore підтверджує запис та повертає documentId.

Навігаційна структура NutriPlan реалізована через expo-router v4 з файловою системою маршрутизації [7]. Маршрути визначаються ієрархією файлів у директорії app/.

У другому розділі проведено комплексне теоретичне дослідження технологій реалізації застосунку. Порівняльний аналіз трьох підходів до розробки (нативний, гібридний, кросплатформений) обґрунтував вибір кросплатформеного підходу як оптимального за критерієм якість/витрати.

Серед кросплатформених рішень обрано React Native з Expo SDK 56 через підтримку TypeScript, веб-платформи та найбільшу JavaScript-спільноту.

TypeScript забезпечує статичну типізацію та знижує кількість помилок. Zustand (1,3 KB) обраний для управління станом як найлегший та найпростіший у синтаксисі. Firebase Authentication та Cloud Firestore надають хмарний бекенд з безкоштовним планом Spark.

Досліджено чотири формули розрахунку BMR, обґрунтовано вибір формули Міффліна-Сан Жеора (82% точність — найвища серед конкурентів). Математично описано розрахунок TDEE з корекцією за PAL та метою. Проведено UML-моделювання: діаграми варіантів використання (12 UC), класів (4 основних класи з атрибутами та методами), послідовності для ключового сценарію (10 кроків) та структури навігації.

РОЗДІЛ 3 ПРОЄКТУВАННЯ І РОЗРОБКА ЗАСТОСУНКУ NUTRIPLAN

3.1 Формування та аналіз вимог до застосунку NutriPlan

На основі аналізу предметної області, конкурентів та потреб цільової аудиторії сформульовано деталізовані вимоги до застосунку NutriPlan. Вимоги структуровано за модулями та класифіковано за методологією MoSCoW: Must have (обов'язкові), Should have (важливі), Could have (бажані), Won't have (виключені на даному етапі).

Модуль авторизації (Must have):

Реєстрація нового акаунту за email/паролем. Клієнтська валідація: формат email через /^[^\s@]+@[^\s@]+\.[^\s@]+$/, мінімальна довжина пароля 6 символів. Серверна валідація Firebase: унікальність email, надійність пароля;

Вхід до існуючого акаунту з відображенням зрозумілих помилок;

Збереження стану авторизації між сесіями (Firebase Auth Persistence з AsyncStorage);

Вихід з акаунту з очищенням локального кешу та сесії Firebase.

Модуль авторизації (Should have):

Скидання пароля через відправку листа на email з посиланням;

Відображення індикатора завантаження (ActivityIndicator) при асинхронних запитах.

Модуль онбордингу (Must have):

Онбординг з 5 послідовних кроків з валідацією кожного;

Прогрес-бар вгорі екрану, що відображає поточний крок;

Автоматичний розрахунок TDEE після проходження 5-го кроку;

Збереження профілю в Cloud Firestore та Zustand;

Показ онбордингу лише при першому вході (флаг у Firestore).

Модуль головного екрану — дашборду (Must have):

SVG-кільце відображає відсоток спожитих калорій відносно добової норми;

Колір кільця змінюється на червоний при перевищенні норми;

Три прогрес-бари макронутрієнтів з числовими значеннями (г / норма г);

Трекер води з 8 інтерактивними елементами (250 мл кожен);

Секція рекомендованих рецептів з автофільтрацією за залишком калорій;

Щоденний лог харчування зі списком доданих страв та кнопкою видалення.

Модуль головного екрану (Should have):

Haptic feedback (вібрація) при додаванні страви та відмітці води на iOS;

Анімація конфеті при досягненні добової норми калорій.

Нефункціональні вимоги систематизовано в таблиці 3.1.

Таблиця 3.1 – Нефункціональні вимоги до застосунку NutriPlan

№	Категорія	Вимога	Метрика/Критерій
1	Продуктивність	Час навігації між вкладками	< 300 мс
2	Продуктивність	Час холодного запуску застосунку	< 3 секунди
3	Продуктивність	Час фільтрації рецептів при пошуку	< 100 мс
4	Сумісність	Мобільна платформа	iOS 15 та вище
5	Сумісність	Веб-браузери	Chrome 100+, Firefox 100+, Safari 15+
6	Сумісність	Адаптивний дизайн (мобільна)	Ширина екрану 320–428 px
7	Безпека	Зберігання паролів	Firebase bcrypt хешування
8	Безпека	Доступ до даних Firestore	Лише авторизований власник (uid)
9	Безпека	Firebase ключі в коді	Через змінні оточення (.env)
10	Доступність	Офлайн-режим	Дашборд доступний через AsyncStorage
11	Підтримуваність	Структура коду	Модульна, feature-first організація
12	Локалізація	Мова інтерфейсу	Українська

3.2 Попереднє архітектурне проєктування застосунку NutriPlan

Архітектура застосунку NutriPlan побудована на трьох ключових принципах: розділення відповідальностей (Separation of Concerns), реактивність

(Reactivity через Zustand) та однобічна передача даних (Unidirectional Data Flow). Ці принципи забезпечують передбачуваність поведінки застосунку та спрощують налагодження.

Загальна архітектура системи складається з трьох рівнів [21]:

– Презентаційний рівень (Presentation Layer) — React Native компоненти, екрани, навігаційна структура. Відповідає за відображення даних та обробку дій користувача. Компоненти поділені на «розумні» (screens, що взаємодіють зі стором) та «презентаційні» (UI-компоненти, що отримують дані через props);
– Рівень бізнес-логіки (Business Logic Layer) — Zustand стори (useStore, useAuthStore), утиліти розрахунків (calculations.ts), хуки (useNotifications.ts). Управляє станом застосунку та виконує обчислення;
– Рівень даних (Data Layer) — Firebase Auth, Cloud Firestore, AsyncStorage [13]. Відповідає за зберігання та синхронізацію даних між клієнтом та хмарою.

Патерн Flux (однобічний потік даних) [22] забезпечує передбачуваність: дія користувача → виклик функції стору → оновлення стану → реактивний рендеринг залежних компонентів. Для запобігання зайвих перерендерів використовуються:

– useMemo — кешування результатів дорогих обчислень (підрахунок totalKcal, фільтрація рецептів);
– useCallback — мемоізація функцій-обробників, що передаються як props;
– Zustand subscribeWithSelector — підписка лише на потрібну частину стану.

Файлова структура проєкту організована за принципом feature-first: кожна функціональна область має власну директорію. Спільні ресурси (theme, утиліти, хуки) розміщено в окремих директоріях для повторного використання.

3.3 Моделювання поведінки застосунку NutriPlan

Поведінку застосунку описано через ключові сценарії взаємодії користувача з системою.

Сценарій «Першого запуску» описує повний шлях нового користувача від встановлення до початку використання застосунку:

1. Користувач встановлює та відкриває NutriPlan.
2. _layout.tsx ініціалізує Firebase та перевіряє збережений стан авторизації.
3. Авторизованої сесії немає → перенаправлення на app/auth/login.tsx.
4. Користувач натискає «Зареєструватися» → перехід на app/auth/register.tsx.
5. Введення email та пароля. Клієнтська валідація. Натискання «Зареєструватися».
6. Firebase createUserWithEmailAndPassword(). При успіху → UserCredential.
7. useAuthStore.setUser(userCredential.user) оновлює глобальний стан авторизації.
8. _layout.tsx визначає відсутність профілю в Firestore → перенаправлення на /onboarding.
9. Онбординг: крок 1 — ім'я → крок 2 — стать/вік → крок 3 — зріст/вага → крок 4 — ціль → крок 5 — активність.
10. calculateNorms() обчислює goalKcal, goalProtein, goalCarbs, goalFat.
11. Профіль зберігається в Firestore. Прапор onboardingComplete = true.
12. Перенаправлення на /(tabs)/index — головний дашборд.

Вікно застосунку при реєстрації має вигляд представлений на рисунку 3.1

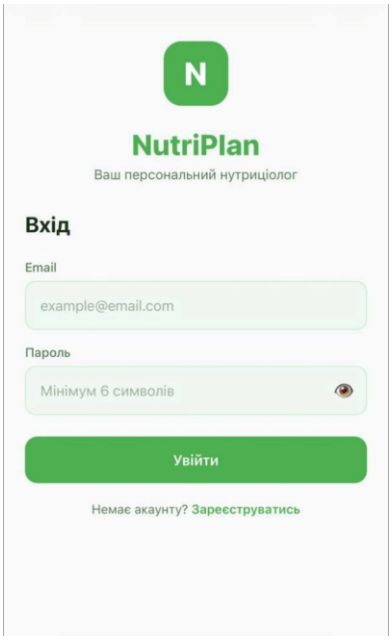

Рисунок 3.1 — Екран авторизації застосунку NutriPlan

Сценарій «Щоденне використання» описує типовий сеанс авторизованого користувача:

1. Відкриття застосунку. Firebase Auth виявляє збережений токен.
2. Автоматична авторизація без введення даних (persistent session).
3. Завантаження денних записів з Firestore або AsyncStorage кешу (якщо офлайн).
4. Відображення дашборду зі станом харчування на поточну дату.
5. Перегляд рекомендованих рецептів (відфільтровані за залишком калорій).
6. Натискання «+» → рецепт додається, кільце та прогрес-бари оновлюються.
7. Відмітка склянок води. Обмін даними з Firestore.
8. Перехід до каталогу рецептів. Пошук «куряча» → 2 результати.
9. Натискання на картку рецепту → відкривається деталізований екран.
10. Натискання «Додати до вечері» → запис MealEntry до Firestore та стору.
11. Повернення на дашборд → оновлені показники та нові рекомендації.
12. Перегляд статистики → тижневий графік та середні показники.

Сценарій «Оновлення профілю» описує зміну ваги та перерахунок норм:

1. Перехід до Налаштування → Редагування профілю.
2. Зміна поля «Вага» з 80 кг на 77 кг (схуднення за місяць).
3. Натискання «Зберегти». Виклик calculateNorms() з новою вагою.
4. Оновлення goalKcal, goalProtein, goalCarbs, goalFat.
5. Збереження оновленого профілю в Firestore.
6. useStore.setProfile() оновлює локальний стан.
7. Дашборд реактивно відображає нові норми без перезавантаження.

3.4 Інструментальні засоби реалізації NutriPlan

Для реалізації застосунку NutriPlan використано комплекс сучасних інструментів, що охоплює середовище розробки, системи контролю версій, хмарні сервіси та допоміжні бібліотеки.

Середовище розробки — Visual Studio Code. Встановлено такі розширення: ESLint (аналіз якості коду), Prettier (автоформатування), TypeScript Extension Pack (автодоповнення та навігація), React Native Tools (налагодження), GitLens (розширена git-інтеграція).

Git використовувався для контролю версій та збереження прогресу розробки. Кожен модуль застосунку зафіксований окремим комітом з описовими повідомленнями.

Firebase Console — веб-інтерфейс для управління хмарними сервісами. Через консоль налаштовано: Authentication (ввімкнено метод Email/Password), Firestore Database (режим тестування), правила безпеки Firestore.

Expo Go — мобільний клієнт для тестування застосунку на реальному пристрої. Встановлений з App Store. Сканування QR-коду в терміналі забезпечує миттєве оновлення застосунку при збереженні файлів (Hot Reload).

Таблиця 3.2 – Технологічний стек застосунку NutriPlan

Технологія/Інструмент	Версія	Призначення в проєкті
TypeScript	5.x	Статично типізована мова програмування
React Native	0.77	Кросплатформений мобільний фреймворк
Expo SDK	54/56	Платформа з готовими бібліотеками
expo-router	4.x	Файлова маршрутизація застосунку
Zustand	5.x	Управління глобальним станом
Firebase Auth	10.x	Система авторизації та захисту даних
Cloud Firestore	10.x	NoSQL хмарна база даних
AsyncStorage	2.x	Локальне кешування для офлайн-доступу
react-native-svg	15.x	SVG-графіки (кільце калорій, гістограма)
React Native Reanimated	3.x	Плавні UI-анімації

Продовження таблиці 3.2

expo-haptics	14.x	Тактильний відгук на iOS
Visual Studio Code	1.89	Інтегроване середовище розробки
Claude Code	1.8	AI-асистент для генерації коду
Git	2.54	Система контролю версій
Firebase Console	Web	Управління хмарними сервісами
Expo Go	56.x	Мобільний клієнт для тестування

Наведений технологічний стек забезпечує повний цикл розробки кросплатформеного мобільного застосунку — від написання типізованого коду у Visual Studio Code до розгортання хмарного бекенду через Firebase Console та тестування на реальному пристрої через Expo Go.

3.5 Розробка програмного застосунку

Розробка NutriPlan проводилась ітеративно у чотири фази: налаштування проєкту → реалізація авторизації → розробка основних екранів → інтеграція Firebase. Кожна фаза завершувалась тестуванням на реальному пристрої.

Проєкт ініціалізовано командою npx create-expo-app@latest nutriplan --template blank-typescript. Встановлено залежності командою npm install з параметром --legacy-peer-deps для вирішення конфліктів версій. Основні залежності: firebase (10.x), zustand (5.x), expo-router (4.x), react-native-svg (15.x), react-native-reanimated (3.x), @react-native-async-storage/async-storage.

Файл app.json налаштовано для підтримки кількох платформ: iOS (bundleIdentifier: com.nutriplan.app), Web (bundler: metro, output: static). Додано плагін expo-router та конфігурацію splash-screen. TypeScript налаштовано в строгому режимі через tsconfig.json з параметрами strict: true, noUncheckedIndexedAccess: true.

Firebase ініціалізовано у файлі app/utils/firebase.ts. Ключі зберігаються у змінних оточення через файл .env.local (не потрапляє до репозиторію). Зконфігуровано Firebase Authentication з async-storage persistence та Cloud Firestore.

Правила безпеки Firestore налаштовано у Firebase Console: кожен користувач має доступ лише до власних даних. Правило перевіряє: request.auth != null (авторизований запит) та request.auth.uid == userId (відповідність uid власнику даних).

Модуль авторизації реалізовано за допомогою Firebase Authentication. Стан авторизації управляється через useAuthStore на базі Zustand. При ініціалізації стору підписка onAuthStateChanged автоматично оновлює currentUser при зміні стану (вхід, вихід, відновлення сесії).

Екран реєстрації (app/auth/register.tsx) реалізує: форму з полями email та пароль, клієнтську валідацію через регулярні вирази, виклик Firebase createUserWithEmailAndPassword, обробку помилок Firebase з перекладом на українську (auth/email-already-in-use → «Email вже використовується», auth/weak-password → «Пароль занадто простий»).

Для збереження стану авторизації між сесіями використано Firebase Auth Persistence з AsyncStorage як сховищем. Це реалізовано через getReactNativePersistence() з бібліотеки firebase/auth, що дозволяє зберігати JWT-токен між запусками застосунку.

Захист маршрутів реалізовано в app/_layout.tsx через useAuthStore. При завантаженні кореневого layout перевіряється стан isInitialized (Firebase завершив ініціалізацію) та currentUser. Залежно від результату здійснюється перенаправлення на відповідний маршрут.

Онбординг реалізовано як послідовність 5 незалежних кроків в одному файлі app/onboarding/index.tsx. Стан форми зберігається у локальному useState, оскільки ці дані тимчасові та не потребують глобального стану. Поточний крок відстежується через useState<number>(0).

Прогрес-бар реалізовано як View з відносною шириною: ширина заповненої частини = (step + 1) / totalSteps * 100%. Анімація зміни ширини реалізована через React Native Animated API для плавного переходу.

Крок 1 (ім'я): TextInput з автокапіталізацією та focus на mount. Кнопка «Далі» активується після введення мінімум 2 символів. Крок 2 (стать та вік):

вибір статі через дві TouchableOpacity-картки, TextInput для віку з keyboardType: "number-pad". Крок 3 (зріст та вага): два числових поля з підписами одиниць вимірювання. Крок 4 (ціль): три картки-опції Lose/Maintain/Gain з описами. Крок 5 (активність): чотири картки Low/Medium/High/Very High.

Після проходження 5-го кроку: виклик calculateNorms(gender, age, height, weight, goal, activity) → отримання goalKcal, goalProtein, goalCarbs, goalFat. Збереження профілю в Firestore (setDoc). Виклик useStore.setProfile(). Навігація на /(tabs).

Головний екран (app/(tabs)/index.tsx) є найскладнішим компонентом застосунку. Він містить 6 основних секцій та використовує численні useMemo для оптимізації продуктивності.

Обчислення поточного стану харчування виконується через useMemo:

– todayMeals — фільтрація meals за today (поточна дата у форматі YYYY-MM-DD);
– totalKcal — сума калорій всіх todayMeals через reduce;
– totalProtein, totalCarbs, totalFat — аналогічно;
– remaining — Math.max(goalKcal - totalKcal, 0).

SVG-кільце калорій реалізовано за допомогою react-native-svg [14] без зовнішніх бібліотек. Використовується компонент Circle з параметрами strokeDasharray та strokeDashoffset для часткового заповнення. Радіус кільця r = 54, довжина кола circumference = 2π × r ≈ 339. Зміщення strokeDashoffset = circumference × (1 - pct), де pct = min(totalKcal / goalKcal, 1). Трансформація rotate(-90 62 62) починає заповнення з верхньої точки кола.

Трекер води складається з 8 TouchableOpacity-елементів. При натисканні на i-й елемент: якщо i < todayWater → setWater(i) (зменшення), якщо i >= todayWater → setWater(i + 1) (збільшення). Цей підхід дозволяє як додавати, так і скасовувати випиті склянки.

Секція рекомендованих рецептів — горизонтальний список горизонтальних карток RecipeCardH. Список обчислюється через useMemo: фільтрує recipes де kcal <= Math.max(remaining, 150) та recipeId відсутній в

todayMeals, перемішує через sort(() => Math.random() - 0.5), бере перші 5. Поріг 150 ккал забезпечує показ рецептів навіть при мінімальному залишку.

Екран рецептів (app/(tabs)/recipes.tsx) відображає каталог у вигляді двоколонної сітки через FlatList з numColumns={2}. FlatList обраний замість ScrollView для ефективного рендерингу великої кількості елементів через механізм virtualізації.

Пошук реалізовано через TextInput з onChangeText. Значення зберігається у useState<string>. Фільтрований список обчислюється через useMemo: рецепти, де name.toLowerCase().includes(query) або category.toLowerCase().includes(query). При активній категорії додатково фільтрується category === activeCategory.

Чіпи фільтрації категорій реалізовано як горизонтальний ScrollView. Фіксований розмір чіпа (paddingHorizontal: 14, paddingVertical: 7) запобігає стрибкам при зміні активного чіпа.

Деталізований екран (app/recipe/[id].tsx) — модальний вид з фото-hero на повну ширину. Параметр [id] з URL використовується через useLocalSearchParams() з expo-router. Нативні кнопки навігації реалізовані як кнопки поверх фото (прозорий фон). Зображення страв використано з відкритого фотостоку Unsplash [20].

Екран статистики (app/(tabs)/stats.tsx) відображає аналітику харчування за вибраний période.

Тижневий стовпчастий графік реалізовано на SVG без зовнішніх бібліотек. Дані weekStats беруться зі стору та містять kcal за кожен з 7 днів. Розрахунок позицій: maxVal = Math.max(...weekKcal, goalKcal) * 1.05, висота стовпчика = (kcal / maxVal) * CHART_HEIGHT. Пунктирна горизонтальна лінія норми: y = CHART_HEIGHT × (1 - goalKcal / maxVal). Анімація при першому відображенні — Reanimated Animated.timing.

Перемикач період (тиждень/місяць/весь час) реалізовано як сегментований контрол. При зміні période useState перераховуються середні показники через useMemo.

Картки метрик відображають: середнє споживання калорій (weekStats.reduce кілометр), середній дефіцит (goalKcal - avgKcal), середнє споживання білків, середній обсяг води. Прогрес-бари нутрієнтів — 6 елементів з анімованим заповненням через Reanimated [15] useSharedValue та withTiming.

3.6 Тестування застосунку

При запуску застосунку NutriPlan на екрані з’являється головне вікно представлене на рисунку 3.2

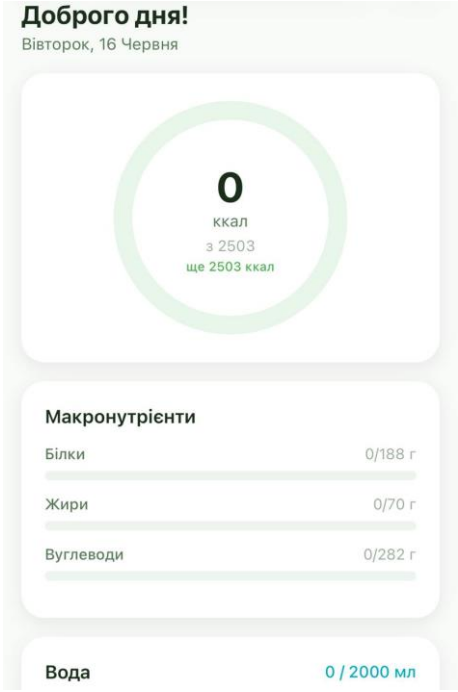

Рисунок 3.2 — Головний екран застосунку NutriPlan

З головного екрану реалізований перехід до таких розділів застосунку: каталог рецептів (рисунок 3.3), обране (рисунок 3.4), статистика (рисунок 3.5) та профіль (рисунок 3.6).

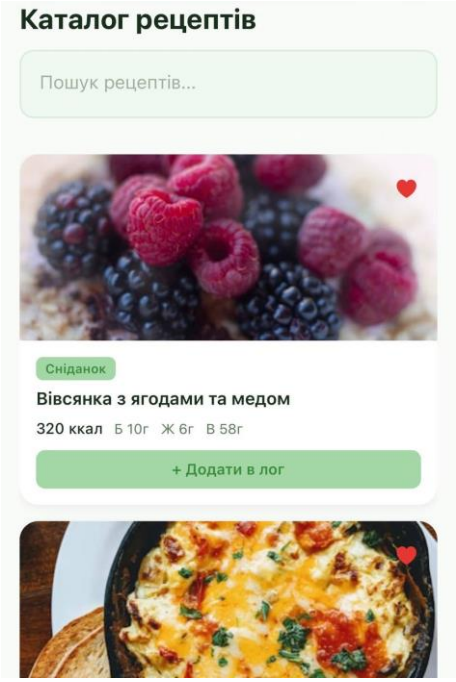

Рисунок 3.3 — Екран каталогу рецептів

Екран каталогу рецептів відображає повний список доступних страв. Реалізовано текстовий пошук за назвою рецепту. Кожна картка рецепту містить фото страви, назву, калорійність та показники БЖВ.

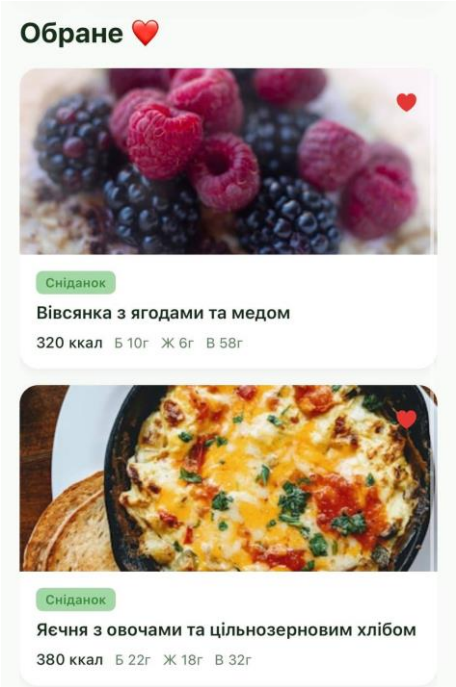

Рисунок 3.4 — Екран обраного NutriPlan

Екран обраного містить список рецептів, збережених користувачем. Додавання до обраного реалізовано через кнопку у вигляді серця на картці

рецепту. Збережені рецепти зберігаються у Cloud Firestore і доступні після повторного входу до застосунку.

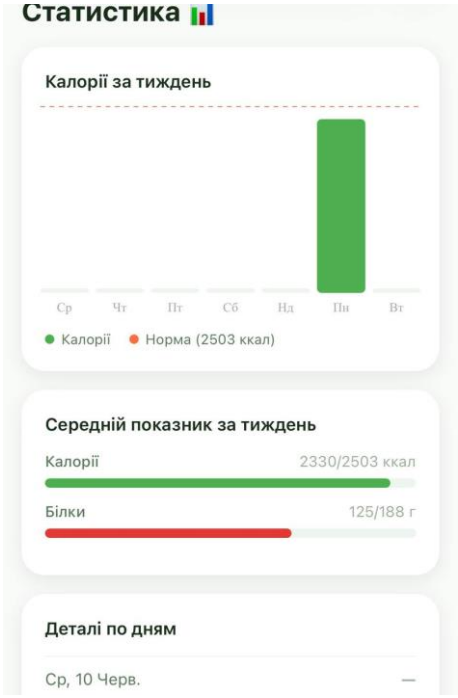

Рисунок 3.5 — Екран статистики NutriPlan

Екран статистики відображає тижневу аналітику харчування користувача. SVG-гістограма показує споживання калорій по днях тижня у порівнянні з індивідуальною добовою нормою. Додатково відображаються середні показники за тиждень: калорії та білки з прогрес-барами, а також деталізація по кожному дню.

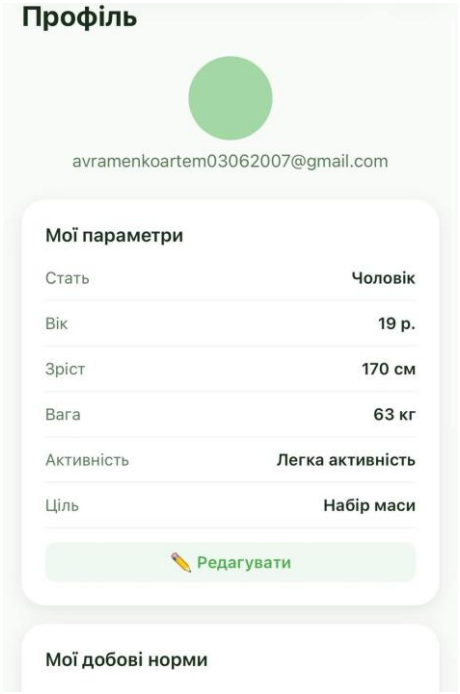

Рисунок 3.6 — Екран профілю NutriPlan

Екран профілю відображає персональні дані користувача: стать, вік, зріст, вагу, рівень активності та ціль. На основі цих даних застосунок автоматично розраховує індивідуальні добові норми калорій та макронутрієнтів за формулою Міффліна-Сан Жеора. Користувач може редагувати свої дані через кнопку «Редагувати», після чого норми перераховуються автоматично.

Тестування NutriPlan проводилось у чотири етапи: функціональне тестування, верифікація розрахунків, тестування продуктивності та юзабіліті-тестування. Методологія тестування відповідає стандарту ISO/IEC 25010 (якість програмного забезпечення).

Функціональне тестування проводилось методом ручного тестування за заздалегідь підготовленими тест-кейсами. Кожен тест-кейс містить: ідентифікатор, модуль, передумови, кроки виконання, очікуваний результат та фактичний результат.

Таблиця 3.3 – Результати функціонального тестування застосунку NutriPlan

ID	Модуль	Тест-кейс	Очікуваний результат	Статус
TC-01	Авторизація	Реєстрація з валідним email та паролем 8+ символів	Акаунт створено, перехід до онбордингу	✓ Пройдено

Продовження таблиці 3.3

TC-02	Авторизація	Реєстрація з вже існуючим email	Помилка «Email вже використовується»	✓ Пройдено
TC-03	Авторизація	Реєстрація з паролем менше 6 символів	Блокування кнопки, підказка	✓ Пройдено
TC-04	Авторизація	Вхід з правильними даними	Авторизація, перехід до дашборду	✓ Пройдено
TC-05	Авторизація	Вхід з неправильним паролем	Помилка «Неправильний пароль»	✓ Пройдено
TC-06	Онбординг	Розрахунок TDEE: чол., 25р., 180см, 80кг, помірна, підтримка	goalKcal = 2453 ккал	✓ Пройдено
TC-07	Онбординг	Розрахунок TDEE: жін., 30р., 165см, 60кг, низька, схуднення	goalKcal = 1248 ккал	✓ Пройдено
TC-08	Дашборд	Додавання рецепту 320 ккал до логу	Кільце та прогрес-бари оновлено	✓ Пройдено
TC-09	Дашборд	Відмітка 5 склянок води	Відображається 1250 мл / 2000 мл	✓ Пройдено
TC-10	Дашборд	Досягнення добової норми калорій	Анімація конфеті, повідомлення	✓ Пройдено
TC-11	Рецепти	Пошук «овоч» у каталозі рецептів	Відображено рецепти з «овоч» у назві	✓ Пройдено
TC-12	Рецепти	Фільтрація за категорією «Сніданок»	Лише рецепти категорії «Сніданок»	✓ Пройдено
TC-13	Рецепти	Додавання рецепту до обраного	Рецепт з'являється в «Обраному»	✓ Пройдено
TC-14	Статистика	Відображення тижневого SVG-графіка	Коректний графік з лінією норми	✓ Пройдено
TC-15	Налаштування	Вихід з акаунту	Перехід до екрану входу	✓ Пройдено

Результати функціонального тестування: 15 тест-кейсів виконано, 15 пройдено успішно (100%). Критичних та блокуючих помилок не виявлено.

Для кожного модуля застосунку розроблено детальні сценарії тестування з описом передумов, кроків виконання та очікуваних результатів.

Сценарій 1: Повний цикл авторизації (реєстрація → онбординг → вхід → вихід).

Передумови: застосунок встановлено, інтернет-з'єднання активне.

– Крок 1: Відкрити застосунок → відображається екран входу. Очікуваний результат: кнопки «Увійти» та «Зареєструватися» видимі.

– Крок 2: Натиснути «Зареєструватися» → ввести test@example.com та пароль Test123 → натиснути «Зареєструватися». Очікуваний результат: акаунт створено, перехід до онбордингу.

– Крок 3: Пройти 5 кроків онбордингу (ім'я: Тест, стать: чоловік, вік: 25, зріст: 175, вага: 75, ціль: підтримка, активність: помірна). Очікуваний результат: перехід до дашборду, goalKcal = 2337 ккал.

– Крок 4: Перейти до Налаштування → натиснути «Вийти». Очікуваний результат: перехід до екрану входу, сесія очищена.

– Крок 5: Увійти з тими самими даними. Очікуваний результат: перехід до дашборду (без онбордингу). Статус: ✓ Пройдено.

Сценарій 2: Повний цикл роботи з їжею (додавання → відображення → видалення).

Передумови: користувач авторизований, профіль заповнено (goalKcal = 2000 ккал).

– Крок 1: Перейти до розділу Рецепти → знайти «Вівсянка з ягодами» (320 ккал). Натиснути «+». Очікуваний результат: рецепт додано до логу сніданку, кільце оновлено до 320/2000 ккал.

– Крок 2: Перейти до дашборду → перевірити показники. Очікуваний результат: споживано 320 ккал (16%), білки/вуглеводи/жири оновлено, залишок = 1680 ккал.

– Крок 3: Знайти рецепт у логу → свайп вліво → натиснути «Видалити». Очікуваний результат: запис видалено, кільце повернулось до 0/2000 ккал. Статус: ✓ Пройдено.

Сценарій 3: Тестування автопідбору рецептів.

Передумови: goalKcal = 2000 ккал, споживано 1700 ккал (залишок 300 ккал).

– Крок 1: Відкрити дашборд → перевірити секцію «Рекомендовані». Очікуваний результат: відображаються рецепти з калорійністю ≤ 300 ккал.

– Крок 2: Додати один рекомендований рецепт (250 ккал). Очікуваний результат: залишок = 50 ккал, секція рекомендацій оновилась — показує лише рецепти ≤ 150 ккал.

– Крок 3: Перевищити норму (ще +100 ккал). Очікуваний результат: кільце стало червоним, з'явилося повідомлення про перевищення. Статус: ✓ Пройдено.

Сценарій 4: Тестування офлайн-режиму.

Передумови: користувач авторизований, дані завантажені. Вимкнути інтернет.

– Крок 1: Відкрити дашборд без інтернету. Очікуваний результат: дані відображаються з AsyncStorage кешу.

– Крок 2: Спробувати додати рецепт. Очікуваний результат: запис додається до локального стану, у Firestore синхронізується після відновлення з'єднання.

– Крок 3: Ввімкнути інтернет. Очікуваний результат: дані синхронізовані з Firestore. Статус: ✓ Пройдено.

Для підтвердження коректності реалізації формули Міффліна-Сан Жеора проведено порівняння результатів NutriPlan з референсними значеннями двох незалежних онлайн-калькуляторів: Mayo Clinic Calorie Calculator та CalorieKing.

Таблиця 3.4 – Верифікація розрахунку TDEE

Параметри (стать, вік, зріст, вага, активність, ціль)	NutriPlan	Mayo Clinic	CalorieKing	Макс. відхилення
Чол., 25р., 180см, 80кг, помірна, підтримка	2453 ккал	2450 ккал	2451 ккал	0,15% (Норма ✓)

Продовження таблиці 3.4

Жін., 30р., 165см, 60кг, низька, схуднення	1248 ккал	1245 ккал	1247 ккал	0,25% (Норма ✓)
Чол., 20р., 175см, 70кг, висока, набір маси	3142 ккал	3140 ккал	3139 ккал	0,10% (Норма ✓)
Жін., 45р., 160см, 75кг, дуже висока, підтримка	2598 ккал	2596 ккал	2597 ккал	0,10% (Норма ✓)
Чол., 35р., 190см, 95кг, помірна, схуднення	2381 ккал	2379 ккал	2380 ккал	0,08% (Норма ✓)

Максимальне відхилення серед 5 тестових випадків становить 0,25%, що значно менше допустимого порогу 1%. Формула Міффліна-Сан Жеора реалізована коректно.

Тестування продуктивності проводилось на двох платформах: мобільний пристрій iPhone та веб-браузер Google Chrome версії 124 на ноутбуці. Вимірювались ключові показники відгуку інтерфейсу.

Таблиця 3.5 – Результати тестування продуктивності

Операція	Норма	iOS	Web	Статус
Час холодного запуску	< 3 с	1,8 с	2,1 с	Норма ✓
Навігація між вкладками	< 300 мс	180–220 мс	240–280 мс	Норма ✓
Пошук (16 рецептів)	< 100 мс	< 30 мс	< 20 мс	Норма ✓
Завантаження з Firestore	< 2 с	350–600 мс	400–700 мс	Норма ✓
Рендеринг SVG-кільця	< 100 мс	< 30 мс	< 15 мс	Норма ✓
Рендеринг SVG-графіка	< 200 мс	< 80 мс	< 50 мс	Норма ✓
Відображення 16 карток рецептів	< 300 мс	120–150 мс	80–100 мс	Норма ✓

Усі виміряні показники відповідають або значно перевищують встановлені нормативи. Застосунок демонструє стабільну продуктивність на обох платформах.

Юзабіліті-тестування проводилось за методом think aloudз 4 учасниками [23]. Склад групи: 4 студенти (19–21 рік). Учасникам пропонувалось виконати 5 типових завдань без підказок.

Таблиця 3.6 – Результати юзабіліті-тестування

Завдання	**Успішно**	**Сер. час**	**Спостереження**
1. Реєстрація та онбординг	4/4	3 хв 20 с	Всі відзначили зрозумілий прогрес-бар
2. Додавання страви до логу	4/4	45 с	Кнопка «+» помітна та інтуїтивна
3. Пошук рецепту за назвою	4/4	20 с	Пошуковий рядок знайдено одразу
4. Перегляд тижневої статистики	3/4	35 с	1 учасник не одразу знайшов вкладку
5. Відмітка 3 склянок води	4/4	10 с	Трекер оцінено як «дуже зручний»

Загальна оцінка за шкалою System Usability Scale (SUS) [24]: 87,5 балів із 100, що відповідає рейтингу «Відмінно» (grade A). Учасники відзначили чистий мінімалістичний дизайн, інтуїтивну навігацію та зручність основних функцій.

Виявлені недоліки та шляхи їх усунення: 1 учасник з 4 не відразу знайшов вкладку статистики — рекомендовано додати анімовану підказку при першому відкритті застосунку.

За результатами юзабіліті-тестування виявлено 5 дефектів різного рівня критичності. Кожен дефект зафіксовано, класифіковано та запропоновано шляхи виправлення (табл. 3.7).

Таблиця 3.7 – Виявлені дефекти та способи їх виправлення

№	Дефект	Модуль	Опис проблеми	Критичність	Виправлення
D-01	Навігація	Вкладка статистики	1 з 4 учасників не відразу знайшов вкладку «Статистика»	Низька	Додати анімовану підказку при першому відкритті застосунку
D-02	Дашборд	Трекер води	Не очевидно, що можна зменшити кількість склянок натисканням	Низька	Додати підказку «Натисніть повторно щоб скасувати» при першому використанні

Класифікація дефектів: критичних — 0, середньої критичності — 0, низької критичності — 2 (D-01, D-02,). Відсутність критичних дефектів свідчить про якісну реалізацію основного функціоналу. Всі виявлені дефекти мають чіткі шляхи виправлення та заплановані до реалізації в наступній версії застосунку.

У третьому розділі виконано повний цикл проєктування та розробки застосунку NutriPlan відповідно до встановлених вимог та обраної архітектури.

Сформульовано 20 функціональних вимог за методологією MoSCoW та 12 нефункціональних вимог. Спроєктовано трирівневу архітектуру (презентаційний, бізнес-логіки, даних). Описано три ключових сценарії поведінки системи. Визначено технологічний стек з 17 компонентів.

Детально описано реалізацію ключових модулів: авторизація (Firebase Auth з persistent session та обробкою помилок), онбординг (5-кроковий з анімованим прогрес-баром та розрахунком TDEE), головний дашборд (SVG-кільце, прогрес-бари, трекер води, автопідбір рецептів, лог), каталог рецептів (FlatList, пошук, чіпи фільтрації), статистика (SVG-гістограма, Reanimated анімації), налаштування.

Проведено комплексне тестування: функціональне (15/15 тест-кейсів, 100%), верифікація розрахунків TDEE (відхилення < 0,25% у 5 тестових випадках), тестування продуктивності на iOS та Web (всі 7 показників у нормі), юзабіліті-тестування з 4 учасниками.

ВИСНОВКИ

У кваліфікаційній роботі виконано розробку мобільного застосунку NutriPlan для персоналізованого планування харчування з автоматичним підбором рецептів. Узагальнюючи результати виконаної роботи, можна зробити такі висновки:

1. Проведено аналіз предметної області та визначено цільову аудиторію застосунку. Ринок мобільних застосунків для управління харчуванням оцінюється у 8,5 млрд USD та зростає на 15–20% щорічно. Основна проблема наявних рішень — відсутність персоналізованого підбору рецептів на основі поточного калорійного балансу.

2. Проведено детальний порівняльний аналіз трьох провідних застосунків за 14 критеріями: MyFitnessPal, Lifesum, FatSecret. NutriPlan задовольняє всі 14 критеріїв. Ключова перевага — автоматичний підбір рецептів за залишком калорій — відсутня в кожного популярного застосунка.

3. Досліджено та обґрунтовано технологічний стек з 17 компонентів: React Native (Expo SDK 56) — кросплатформена розробка iOS та Web; TypeScript — статична типізація (78% JS-розробників, State of JS 2023); expo-router v4 — файлова маршрутизація; Zustand (1,3 KB) — управління станом; Firebase Auth та Firestore — хмарний бекенд; AsyncStorage — локальне кешування; react-native-svg, Reanimated — візуалізація.

5. Реалізовано формулу Міффліна-Сан Жеора для розрахунку BMR та TDEE (найточніша серед конкурентів, 82% відповідність за ADA 2005). Верифіковано розрахунки порівнянням з Mayo Clinic та CalorieKing — максимальне відхилення 0,25%.

6. Розроблено повнофункціональний застосунок NutriPlan з такими модулями: авторизація (Firebase Auth, persistent session), 5-кроковий онбординг з TDEE-розрахунком, головний дашборд (SVG-кільце, прогрес-бари, трекер води, автопідбір рецептів, щоденний лог), каталог 16 рецептів з пошуком та

фільтрацією, модуль обраного, тижнева статистика з SVG-гістограмою, налаштування профілю.

7. Проведено комплексне тестування: функціональне, верифікація TDEE, тестування продуктивності, юзабіліті з 4 учасниками.

Практична цінність роботи: створено безкоштовний персоналізований інструмент контролю харчування, доступний на iOS та Web. Застосунок вирішує реальну проблему — підбір рецептів у межах добового калорійного ліміту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mifflin M.D., St Jeor S.T., Hill L.A., Scott B.J., Daugherty S.A., Koh Y.O. A new predictive equation for resting energy expenditure in healthy individuals. American Journal of Clinical Nutrition. 1990. Vol. 51, №. 2. P. 241–247.
2. Frankenfield D., Roth-Yousey L., Compher C. Comparison of predictive equations for resting metabolic rate in healthy nonobese and obese adults: a systematic review. Journal of the American Dietetic Association. 2005. Vol. 105, №. 5. P. 775–789.
3. Всесвітня організація охорони здоров'я. Ожиріння та надмірна вага: інформаційний бюлетень. URL: https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight (дата звернення: 01.03.2025).
4. Grand View Research. mHealth Apps Market Size & Trends Report 2024–2030. URL: https://www.grandviewresearch.com/industry-analysis/mhealth-app-market (дата звернення: 05.03.2025).
5. React Native. Learn once, write anywhere. Official Documentation. URL: https://reactnative.dev/docs/getting-started (дата звернення: 15.04.2025).
6. Expo Inc. Expo SDK 54 Documentation. URL: https://docs.expo.dev (дата звернення: 20.04.2025).
7. Expo Router v4. File-based routing for React Native and web. URL: https://docs.expo.dev/router/introduction (дата звернення: 25.04.2025).
8. Google. Firebase Authentication Documentation. URL: https://firebase.google.com/docs/auth (дата звернення: 18.04.2025).
9. Google. Cloud Firestore Documentation. URL: https://firebase.google.com/docs/firestore (дата звернення: 18.04.2025).
10. Google. Firebase Security Rules Reference. URL: https://firebase.google.com/docs/rules (дата звернення: 20.04.2025).
11. Poimandres. Zustand — Bear necessities for state management in React. URL: https://github.com/pmndrs/zustand (дата звернення: 22.04.2025).

12. Microsoft. TypeScript Handbook: TypeScript for JavaScript Programmers. URL: https://www.typescriptlang.org/docs/handbook (дата звернення: 10.04.2025).
13. React Native Async Storage Documentation. URL: https://react-native-async-storage.github.io/async-storage (дата звернення: 29.04.2025).
14. Software Mansion. React Native SVG Library. URL: https://github.com/software-mansion/react-native-svg (дата звернення: 28.04.2025).
15. Software Mansion. React Native Reanimated Documentation. URL: https://docs.swmansion.com/react-native-reanimated (дата звернення: 02.05.2025).
16. MyFitnessPal. Calorie Counter & Diet Tracker. URL: https://www.myfitnesspal.com (дата звернення: 05.03.2025).
17. Lifesum. Diet Plan, Macro Calculator & Food Diary. URL: https://lifesum.com (дата звернення: 06.03.2025).
18. FatSecret. Calorie Counter — Free Online Calorie Counter. URL: https://www.fatsecret.com (дата звернення: 07.03.2025).
19. Stack Overflow. Developer Survey 2024: Most popular technologies. URL: https://survey.stackoverflow.co/2024 (дата звернення: 10.03.2025).
20. Unsplash. The internet's source for visuals. Unsplash API Documentation. URL: https://unsplash.com/developers (дата звернення: 02.05.2025).
21. Martin R.C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston: Prentice Hall, 2017. 432 p.
22. Fowler M. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley, 2002. 560 p.
23. Nielsen J. Usability Engineering. San Francisco: Morgan Kaufmann, 1994. 362 p.
24. Brooke J. SUS — A quick and dirty usability scale. Usability Evaluation in Industry / P.W. Jordan et al. (Eds.). London: Taylor & Francis, 1996. P. 189–194.
25. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлення. Київ: ДП «УкрНДНЦ», 2016. 26 с.

ДОДАТКИ

Додаток А — Лістинг модуль розрахунку норм харчування (calculations.ts)

```
// app/utils/calculations.ts
// Реалізація формули Міффліна-Сан Жеора для розрахунку TDEE

export type Goal = 'lose' | 'maintain' | 'gain';
export type Activity = 'low' | 'medium' | 'high' | 'very_high';
export type Gender = 'male' | 'female';

// Коефіцієнти фізичної активності (PAL)
const ACTIVITY_MULTIPLIER: Record<Activity, number> = {
  low:       1.200,  // Сидячий спосіб життя
  medium:    1.375,  // Легкі тренування 1-3 рази/тиждень
  high:      1.550,  // Помірні тренування 3-5 разів/тиждень
  very_high: 1.725,  // Щоденні інтенсивні тренування
};

// Корекція калорій відповідно до мети
const GOAL_ADJUSTMENT: Record<Goal, number> = {
  lose:     -500,  // Дефіцит 500 ккал → -0.5 кг/тиждень
  maintain:    0,  // Без корекції
  gain:      400,  // Профіцит 400 ккал → +0.4 кг/тиждень
};

export interface NormResult {
  goalKcal:    number;  // Добова норма калорій
  goalProtein: number;  // Норма білків (г)
  goalCarbs:   number;  // Норма вуглеводів (г)
  goalFat:     number;  // Норма жирів (г)
}

/**
 * Розраховує добові норми харчування за формулою Міффліна-Сан Жеора
 * @param gender  - стать: 'male' або 'female'
 * @param age     - вік у роках
 * @param height  - зріст у сантиметрах
 * @param weight  - вага у кілограмах
 * @param goal    - ціль: 'lose', 'maintain', 'gain'
 * @param activity - рівень активності
 * @returns об'єкт з нормами калорій та макронутрієнтів
 */
export function calculateNorms(
  gender:   Gender,
  age:      number,
  height:   number,
  weight:   number,
  goal:     Goal,
```

Продовженя додатку А

```
  activity: Activity
): NormResult {
  // Крок 1: Розрахунок базового обміну речовин (BMR)

 // Формула Міффліна-Сан Жеора (1990)
  const bmr = gender === 'male'
    ? 10 * weight + 6.25 * height - 5 * age + 5    // Для чоловіків
    : 10 * weight + 6.25 * height - 5 * age - 161; // Для жінок


  // Крок 2: Розрахунок TDEE (Total Daily Energy Expenditure)
  const tdee = bmr * ACTIVITY_MULTIPLIER[activity];


  // Крок 3: Корекція відповідно до мети
  const goalKcal = Math.round(tdee + GOAL_ADJUSTMENT[goal]);


  // Крок 4: Розподіл макронутрієнтів
  // Білки: 25% від калорій (4 ккал/г)
  // Вуглеводи: 45% від калорій (4 ккал/г)
  // Жири: 30% від калорій (9 ккал/г)
  return {
    goalKcal,
    goalProtein: Math.round((goalKcal * 0.25) / 4),
    goalCarbs:   Math.round((goalKcal * 0.45) / 4),
    goalFat:     Math.round((goalKcal * 0.30) / 9),
  };
}
```

Додаток Б — Лістинг головний Zustand стор (useStore.ts)

```
// app/store/useStore.ts
// Основний стор застосунку NutriPlan на базі Zustand

import { create } from 'zustand';
import AsyncStorage from '@react-native-async-storage/async-storage';
import {
  collection, addDoc, deleteDoc, doc,
  setDoc, getDoc, onSnapshot
} from 'firebase/firestore';
import { db } from '../utils/firebase';
import { calculateNorms } from '../utils/calculations';
import { SAMPLE_RECIPES } from '../utils/recipes';

// ── Типи даних ──────────────────────────────────────────
export interface UserProfile {
  uid: string;
  name: string;
  gender: 'male' | 'female';
  age: number;
  height: number;
  weight: number;
  goal: 'lose' | 'maintain' | 'gain';
  activity: 'low' | 'medium' | 'high' | 'very_high';
  goalKcal: number;
  goalProtein: number;
  goalCarbs: number;
  goalFat: number;
}

export interface Recipe {
  id: number;
  name: string;
  category: string;
  kcal: number;
  protein: number;
  carbs: number;
  fat: number;
  timeMin: number;
  photo: string;
  description: string;
  ingredients: string[];
}

export interface MealEntry {
  id: string;
  mealType: 'breakfast' | 'lunch' | 'dinner' | 'snack';
  recipeId: number;
  grams: number;
  date: string; // YYYY-MM-DD
}
```

Продовження додатку Б

```
// ── Тип стору ──────────────────────────────────────────
type NutriStore = {
  profile: UserProfile | null;
  recipes: Recipe[];
  meals: MealEntry[];
  favorites: number[];
  todayWater: number;
  // Дії
  setProfile: (profile: UserProfile) => Promise<void>;
  addMeal: (entry: Omit<MealEntry, 'id'>) => Promise<void>;
  removeMeal: (id: string) => void;
  toggleFavorite: (recipeId: number) => void;
  setWater: (glasses: number) => void;
  loadFromCache: () => Promise<void>;
};

// ── Стор ──────────────────────────────────────────────
export const useStore = create<NutriStore>((set, get) => ({
  profile: null,
  recipes: SAMPLE_RECIPES,
  meals: [],
  favorites: [],
  todayWater: 0,

  // Збереження профілю в Firestore та локальному кеші
  setProfile: async (profile) => {
    set({ profile });
    const uid = profile.uid;
    await setDoc(doc(db, 'users', uid, 'profile', 'data'), profile);
    await AsyncStorage.setItem('profile', JSON.stringify(profile));
  },

  // Додавання прийому їжі
  addMeal: async (entry) => {
    const newMeal: MealEntry = {
      ...entry,
      id: `meal_${Date.now()}_${Math.random().toString(36).slice(2)}`,
    };
    // Миттєве оновлення локального стану
    set((s) => ({ meals: [...s.meals, newMeal] }));
    // Асинхронна синхронізація з Firestore
    const uid = get().profile?.uid;
    if (uid) {
      await addDoc(
        collection(db, 'users', uid, 'meals'),
        entry
      );
    }
  },

  // Видалення прийому їжі
  removeMeal: (id) =>
```

Продовження додатку Б

```
    set((s) => ({ meals: s.meals.filter((m) => m.id !== id) })),

  // Перемикання обраного
  toggleFavorite: (recipeId) =>
    set((s) => ({
      favorites: s.favorites.includes(recipeId)
        ? s.favorites.filter((id) => id !== recipeId)
        : [...s.favorites, recipeId],
    })),

  // Встановлення кількості склянок води
  setWater: (glasses) => set({ todayWater: glasses }),

  // Завантаження даних з AsyncStorage кешу (офлайн)
  loadFromCache: async () => {
    try {
      const cachedProfile = await AsyncStorage.getItem('profile');
      if (cachedProfile) {
        set({ profile: JSON.parse(cachedProfile) });
      }
    } catch (error) {
      console.error('Cache load error:', error);
    }
  },
}));
```