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 МОБІЛЬНОГО ДОДАТКУ ДЛЯ ЗАНЯТЬ ЙОГОЮ З ВИКОРИСТАННЯМ FLUTTER

DEVELOPMENT OF A MOBILE APPLICATION FOR YOGA PRACTICE USING FLUTTER

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

Виконав: здобувач вищої освіти
групи ІПЗ-42
Лапюк Роман Андрійович

Керівник:
Гулієва Наталія Михайлівна

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

_____

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 2024р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Лапюку Роману Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка мобільного додатку для занять йогою з використанням Flutter»*
Керівник роботи: *к.т.н., доцент Гулієва Н. М.*
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: *Visual Studio Code, Flutter, Dart, Android Studio, SQLite, методичні вказівки до виконання кваліфікаційної роботи бакалавра*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *розробка інтерфейсу додатку з каталогом вправ і персональним кабінетом; Збереження прогресу тренувань у локальній базі даних; Адаптація інтерфейсу для різних екранів і платформ*
5. Перелік графічного матеріалу: *17 рисунків, 5 таблиць, 2 додатки*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Гулієва Н. М.*		
Специфікація вимог до розробленої системи	*Гулієва Н. М.*		
Розробка об'єкта проектування	*Гулієва Н. М.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	0,86 %		
Академічна доброчесність	*Гулієва Н. М.*		

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

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	Виконано
2	Аналіз проблеми розробки та впровадження об'єкту проектування	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 _____ Роман ЛАПЮК

Керівник кваліфікаційної роботи _____ Наталія ГУЛІЄВА

АНОТАЦІЯ

Лапюк Р. А. Розробка мобільного додатку для занять йогою з використанням Flutter. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі здійснено аналіз сучасного стану ринку мобільних додатків для йоги, виявлено їхні недоліки та визначено технічні проблеми, які постають при розробці таких рішень. Сформульовано завдання на розробку власного застосунку. У другому розділі обґрунтовано вибір архітектури, технологій та засобів реалізації, створено UML-діаграми, визначено функціональні, нефункціональні та технічні вимоги до системи. У третьому розділі реалізовано практичну частину додатку: побудовано інтерфейс, логіку взаємодії, базу даних, а також здійснено тестування й формування інсталяційного пакета.

Ключові слова: Flutter, йога, мобільний додаток, SQLite, інтерфейс, фітнес, Dart, MVC, кросплатформенність.

ABSTRACT

Lapiuk R. Development of a Mobile Application for Yoga Practice Using Flutter. Manuscript. Bachelor's qualification thesis in the study program "Software Engineering" of the specialty "Software Engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The first chapter presents an analysis of the current market for mobile yoga applications, identifies major shortcomings of existing solutions, and formulates the main objectives of the new application. The second chapter justifies the choice of architecture, technologies, and development tools, defines functional, non-functional, and technical requirements, and presents UML diagrams. The third chapter contains the practical implementation: user interface creation, interaction logic, local database structure, testing, and generation of the installation package. The conclusions summarize the development results, confirm the achievement of the set goals, and outline prospects for future improvement of the application.

Keywords: Flutter, yoga, mobile application, SQLite, interface, fitness, Dart, MVC, cross-platform.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі дедалі більше зростає потреба в інструментах, що сприяють підтримці фізичного та психічного здоров’я. Йога – це комплексна практика, що поєднує фізичні вправи, дихальні техніки та медитацію. Вона набуває популярності серед людей різного віку та рівня підготовки завдяки своїй здатності знижувати рівень стресу, покращувати гнучкість, зміцнювати м’язи та підвищувати концентрацію.

У цьому контексті актуальним є створення мобільного додатку для занять йогою, який би поєднував інтуїтивний інтерфейс, адаптивні тренування та можливість відстеження особистого прогресу. Використання технології Flutter дозволяє ефективно реалізувати кросплатформений додаток із гарною продуктивністю та привабливим дизайном.

Метою роботи є розробка мобільного додатку для занять йогою на базі Flutter, який би забезпечував зручний доступ до тренувальних програм, підтримував персоналізацію та сприяв залученню і утриманню користувачів.

Об’єктом є процес створення мобільних додатків для фітнесу.

Предметом виступають особливості розробки мобільного додатку для йоги з використанням Flutter, включно з технічними, функціональними та UX-рішеннями.

Для досягнення поставленої мети в межах кваліфікаційної роботи необхідно виконати наступні технічні та функціональні завдання:

– визначення вимог до розроблюваного додатку;

– аналіз та вибір засобів для створення додатку;

– розробка додатку засобами які були вибрані;

– забезпечити локальне збереження даних;

– впровадити адаптивний інтерфейс, що коректно відображається на різних розмірах екранів;

– провести комплексне тестування функціоналу, продуктивності та стабільності роботи;

– сформувати інсталяційний пакет APK для подальшого розповсюдження додатку.

Таким чином, розробка мобільного додатку для занять йогою буде виконуватись з використанням фреймворку Flutter, що забезпечить створення ефективного кросплатформеного рішення з сучасним дизайном та високою продуктивністю. У межах кваліфікаційної роботи буде реалізовано повний цикл розробки: від визначення вимог та вибору необхідних інструментів – до створення, тестування та підготовки додатку до розповсюдження.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ РІШЕНЬ У СФЕРІ МОБІЛЬНИХ ДОДАТКІВ ДЛЯ ЙОГИ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Сучасний ринок фітнес-додатків, зокрема для занять йогою, переживає період інтенсивного розвитку, що обумовлено зростанням попиту на дистанційні формати тренувань та персоніфіковані підходи до здоров'я [1]. Проте глибокий аналіз функціональних можливостей лідируючих рішень виявляє суттєві системні обмеження, які стримують розвиток цього сегменту.

Серед ключових продуктів, що домінують на ринку фітнес-додатків для йоги, варто виділити кілька найпопулярніших рішень, які вирізняються функціональністю, якістю контенту та підходами до взаємодії з користувачами.

Down Dog – один із найзавантажуваніших додатків для занять йогою, який використовує штучний інтелект для генерації унікальних тренувань. Завдяки цьому кожне заняття має індивідуальний сценарій, що мінімізує ефект повторюваності. Користувач може самостійно обрати рівень складності, тривалість, тип занять та інші параметри, що дозволяє адаптувати тренування під особисті потреби (рис. 1.1) [2].

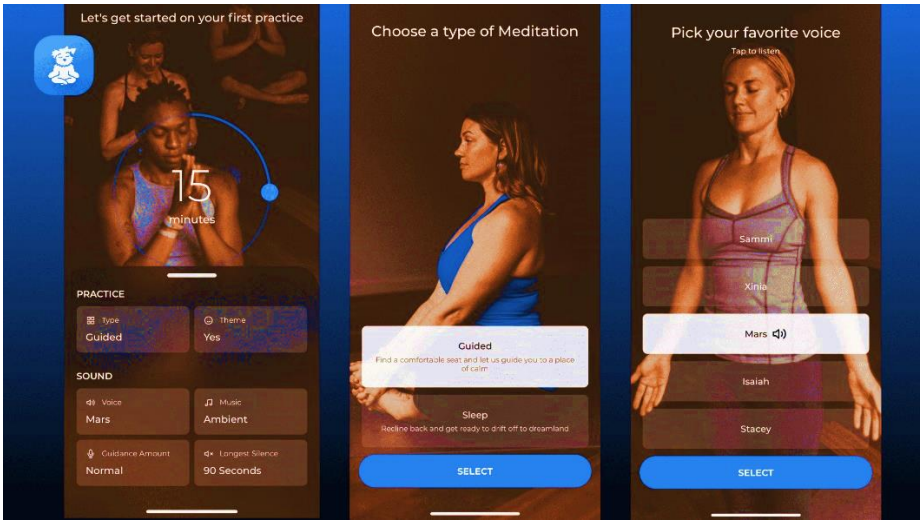

Рисунок 1.1 – Вигляд додатку Down Dog [3]

Daily Yoga пропонує комплексне рішення, орієнтоване не лише на фізичну активність, а й на соціальну взаємодію. Додаток має функціонал створення спільнот, участі в групових викликах, коментування та обміну досягненнями між користувачами. Крім стандартних тренувань, тут доступні курси з медитації та харчування (рис. 1.2) [4].

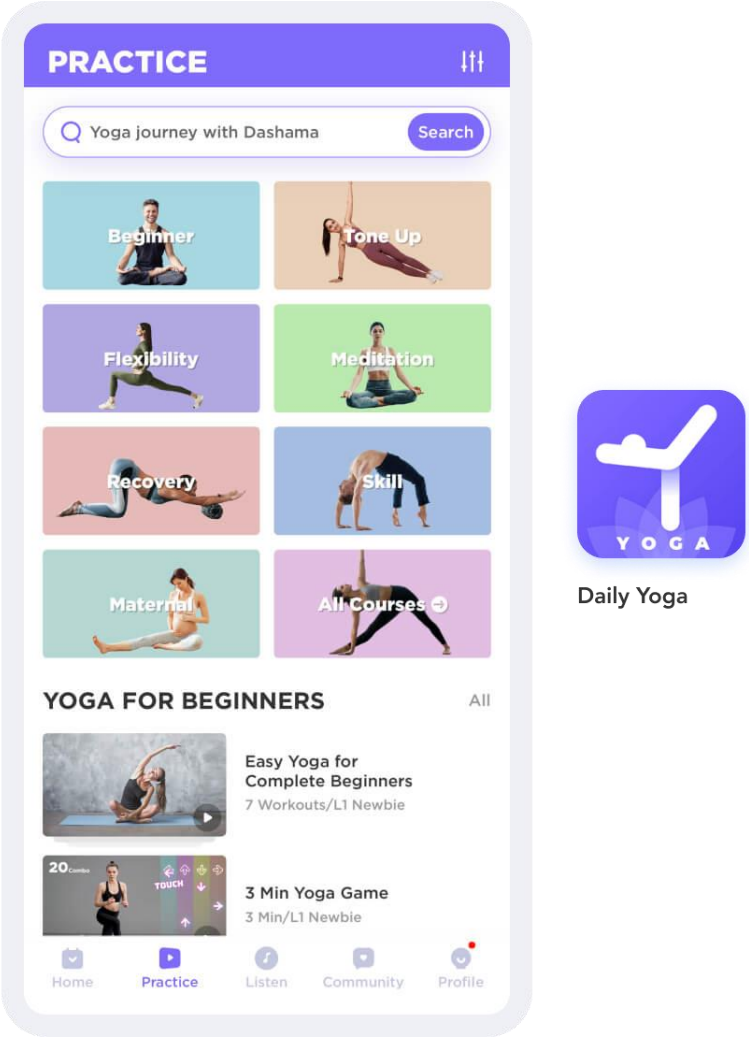

Рисунок 1.2 – Вигляд додатку Daily Yoga [5]

Yoga Studio позиціонується як професійний додаток з акцентом на візуальну якість та структурованість занять. У ньому представлений «кінотеатр вправ» – велика бібліотека відео у високій HD-якості, відзнятих за участю сертифікованих інструкторів. Користувачі можуть обирати програми за цілями (наприклад, покращення гнучкості чи зняття стресу) та створювати власні комплекси (рис. 1.3) [6].

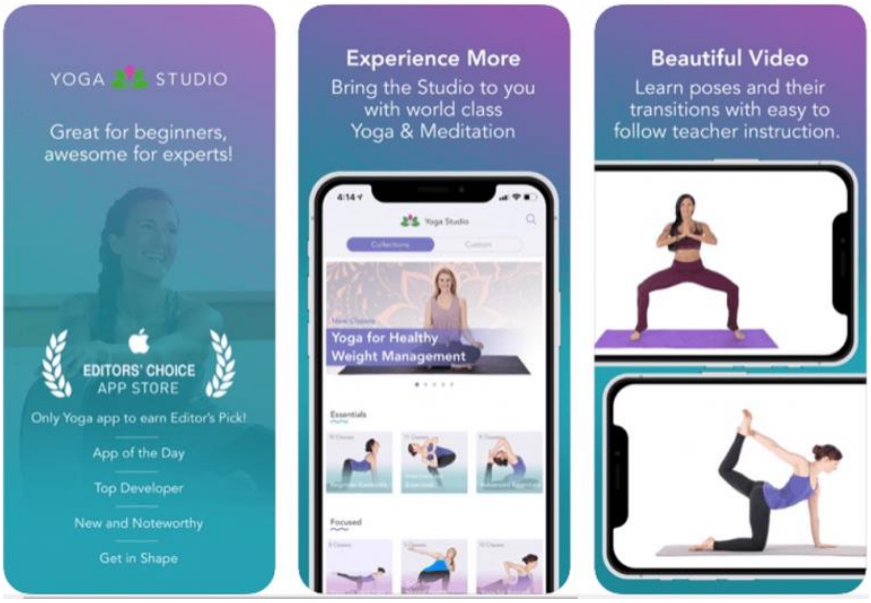

Рисунок 1.3 – Вигляд додатку Yoga Studio [7]

Asana Rebel поєднує елементи йоги та фітнесу, пропонуючи гібридну модель занять. Додаток орієнтований на тих користувачів, які хочуть підтримувати форму, розвивати силу й витривалість, а не лише займатися класичною йогою. Крім фізичних тренувань, додаток також включає контент для мотивації, трекінгу здоров'я та формування корисних звичок (рис. 1.4) [8].

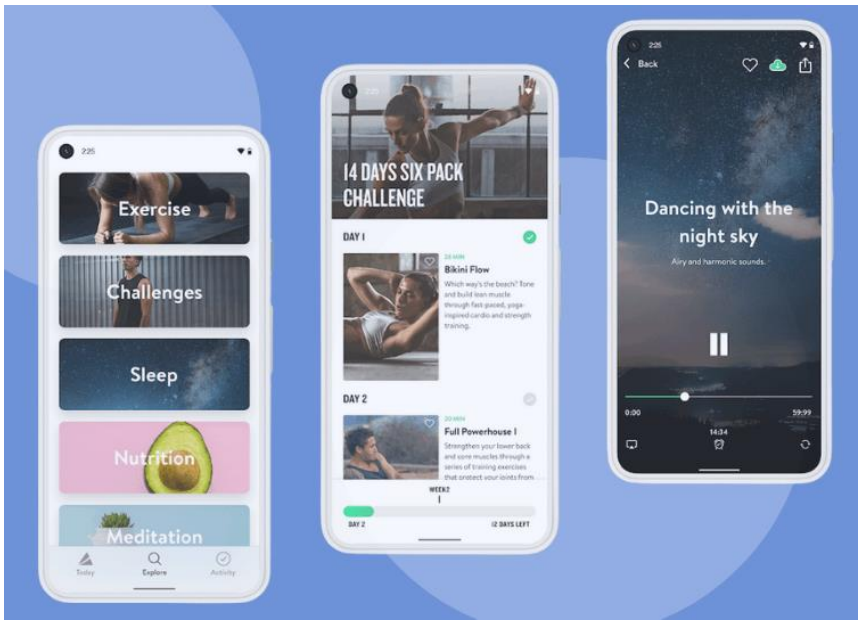

Рисунок 1.4 – Вигляд додатку Asana Rebel [9]

Технічна реалізація більшості сучасних додатків має низку критичних недоліків. Архітектурні обмеження, пов’язані з використанням нативних підходів до розробки, призводять до значного збільшення витрат на підтримку мультиплатформенності. Дослідження показують, що близько половини розробників стикаються з проблемами синхронізації функціоналу між iOS та Android версіями, що безпосередньо впливає на якість користувацького досвіду [10].

Особливу увагу заслуговує аналіз алгоритмів персоналізації, які використовуються в сучасних рішеннях. Більшість систем обмежуються простими фільтрами за рівнем складності або тривалістю тренувань, не враховуючи таких ключових факторів, як антропометричні характеристики користувача, наявність хронічних захворювань опорно-рухового апарату, динаміка прогресу в різних типах асан та біометричні показники, що фіксуються носимими пристроями.

Патентний пошук за останні 5 років свідчить про значний інтерес до технологій комп’ютерного зору для аналізу техніки виконання вправ (патенти US20210166421, EP3895594), проте комерційні реалізації цих рішень поки що обмежені досить вузькими нішами преміум-сегменту [11].

Ключовою проблемою сучасних додатків є відсутність цілісного підходу до побудови тренувальних програм. Більшість рішень працюють за принципом лінійних алгоритмів, тоді як сучасні дослідження в області спортивної медицини доводять ефективність адаптивних систем, що враховують фізіологічну реакцію організму під час тренування, психоемоційний стан користувача за даними сенсорів та довгострокову динаміку розвитку гнучкості і сили.

Враховуючи ці фактори, розробка нового рішення на базі Flutter відкриває унікальні можливості для створення по-справжньому інтелектуальної системи. Використання сучасних архітектурних підходів (BLoC, Provider) дозволяє реалізувати складну бізнес-логіку, зберігаючи високу продуктивність на обох платформах.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Було проведено аналіз предметної області та проаналізовано сучасний стан проблеми.

Для реалізації поставленої мети в межах кваліфікаційної роботи необхідно виконати наступні технічні та функціональні завдання:

– визначення вимог до розроблюваного додатку;

– аналіз та вибір засобів для створення додатку;

– розробка додатку засобами які були вибрані;

– забезпечити локальне збереження даних;

– впровадити адаптивний інтерфейс, що коректно відображається на різних розмірах екранів;

– провести комплексне тестування функціоналу, продуктивності та стабільності роботи;

– сформувати інсталяційний пакет APK для подальшого розповсюдження додатку.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

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

У процесі розробки мобільного додатку для занять йогою з використанням Flutter виникає потреба у чіткому плануванні архітектури програмного забезпечення. Це дозволяє забезпечити масштабованість, повторне використання компонентів, легкість у тестуванні та подальшій підтримці.

Для моделювання програмної системи обрано об’єктно-орієнтовану модель, що базується на принципах UML (Unified Modeling Language) [12]. Такий підхід є доцільним з огляду на те, що Flutter передбачає використання об’єктно-орієнтованого підходу, адже код пишеться мовою Dart, яка повністю підтримує ООП [13] (класи, спадкування, поліморфізм тощо).

Також UML дозволяє чітко візуалізувати елементи додатку – класи, взаємозв’язки між ними, сценарії взаємодії користувача з інтерфейсом. Об’єктна модель дозволяє легко модифікувати окремі компоненти системи, не зачіпаючи всю архітектуру.

UML є стандартом де-факто для моделювання систем, що робить створену модель зрозумілою для інших розробників, технічних експертів і рецензентів.

Також, враховуючи специфіку мобільного додатку (наявність користувацького інтерфейсу, роботи з базою даних, відображення тренувань, збереження прогресу), було обрано використання архітектурної моделі MVC (Model-View-Controller) у поєднанні з платформеною структурою Flutter (Widgets, State, BuildContext). Це дозволяє відокремити логіку від відображення, забезпечуючи кращу підтримку й тестованість.

Таким чином, для моделювання системи застосовуються UML як формальна мова моделювання, ООП як метод організації структури

програмного коду та MVC як архітектурна модель реалізації. Цей вибір забезпечує зручність у подальшому проєктуванні, реалізації й обслуговуванні мобільного додатку для йоги.

Для ефективної розробки мобільного додатку «Yoga Workout App» було застосовано комплекс методів і підходів, що дозволяють виявити, сформулювати та проаналізувати вимоги до програмного забезпечення. Це забезпечує максимальну відповідність продукту очікуванням користувачів, а також технічним і функціональним обмеженням платформи Flutter.

Основні методи, які були застосовані це опитування потенційних користувачів, Аналіз аналогічних рішень, виявлення сильних та слабких сторін.

Було проведено усне опитування серед представників цільової аудиторії (людей, які займаються йогою або шукають зручні мобільні фітнес-додатки), під час якого зібрано інформацію про бажаний функціонал і частоту використання додатку.

Виявлені побажання включають простий, інтуїтивний інтерфейс, можливість вибору вправ за рівнем складності, трекінг прогресу, підказки та відео до кожної вправи, а також роботу без інтернету.

Було проаналізовано функціонал трьох популярних фітнес-додатків: Yoga for Beginners (Leap Fitness Group), Down Dog та Daily Yoga. На основі аналізу виявлено загальні сильні сторони, такі як чітка структура вправ, візуальні демонстрації, ведення статистики та підтримка офлайн-режиму. Також визначено типові слабкі сторони – перевантаженість інтерфейсу, обов’язкова реєстрація і відсутність вільного редагування планів.

Для кожного ключового сценарію використання було розроблено короткий опис дій користувача (рис. 2.1), що дозволяє формалізувати основні функції системи.

Рисунок 2.1 – Узагальнена діаграма використання для користувача

В таблиці 2.1 описано кожен об’єкт діаграми використання для користувача.

Таблиця 2.1 – Опис об’єктів діаграми використання

Об’єкт	Опис
Користувач	основна дійова особа, яка взаємодіє з додатком через графічний інтерфейс
Перегляд списку вправ	користувач має доступ до загального переліку доступних вправ, згрупованих за категоріями або програмами
Вибір категорії тренувань	дає змогу відфільтрувати вправи за тематиками, наприклад: розтяжка, медитація, силова йога тощо
Перегляд детальної інформації про вправу	відкривається окремий екран із описом вправи, її користю, тривалістю та ілюстрацією
Запуск тренування	активується таймер або відеоінструкція, починається відлік виконання
Відмітити виконану вправу	після завершення користувач підтверджує виконання, і результат зберігається у прогрес
Переглянути прогрес	дає змогу переглянути історію виконаних вправ, відмічених днів, а також статистику

На основі попереднього аналізу було сформульовано функціональні, нефункціональні та технічні вимоги до мобільного додатку для занять йогою. В таблиці 2.2 описано функціональні вимоги.

Таблиця 2.2 – Функціональні вимоги додатку

№	Вимога	Опис
F1	Перегляд списку вправ	Користувач повинен мати можливість переглядати вправи, згруповані за категоріями
F2	Вибір категорії	Користувач може обирати тип тренування (ранкове, вечірнє, розтяжка тощо)
F3	Детальний перегляд вправи	Додаток має відображати опис, зображення, тривалість вправи
F4	Запуск тренування	Користувач повинен мати змогу запустити вправу з таймером або підказками
F5	Збереження прогресу	Після виконання вправи додаток має зберігати дату й статус виконання
F6	Пошук вправ	Має бути можливість швидко знаходити вправу за назвою
F7	Робота в офлайн-режимі	Додаток повинен працювати без доступу до Інтернету

В таблиці 2.3 зібрані нефункціональні вимоги, які визначають, наскільки система буде швидкою, надійною і зручною для користувача додатку які точно варто врахувати.

Таблиця 2.3 – Нефункціональні вимоги додатку

№	Вимога	Опис
NF1	Кросплатформеність	Додаток має однаково працювати на Android та iOS
NF2	Локальне збереження даних	Уся інформація про вправи та прогрес зберігається в SQLite
NF3	Продуктивність	Час запуску додатку не повинен перевищувати 2 секунд
NF4	Юзабіліті	Інтерфейс повинен бути інтуїтивно зрозумілим, з мінімалістичним дизайном
NF5	Масштабованість	Архітектура має дозволяти легке додавання нових вправ та розділів
NF6	Безпека	Дані користувача не повинні передаватися третім сторонам

В таблиці 2.4 описано технічні вимоги додатку що задають базу для всієї системи які визначають, на чому і як буде будуватись наш додаток.

Таблиця 2.4 – Технічні вимоги додатку

№	Вимога	Опис
T1	Мова програмування	Dart (у межах Flutter SDK)
T2	База даних	SQLite, з використанням пакету sqflite
T3	Мінімальна версія Android	Android Android 6.0 (API 23)
T4	Мінімальна версія iOS	iOS 12.0
T5	Зберігання медіа	Зображення вправ зберігаються в assets/images/

Проєктування програмного забезпечення розпочнімо з діаграми класів (рис. 2.2).

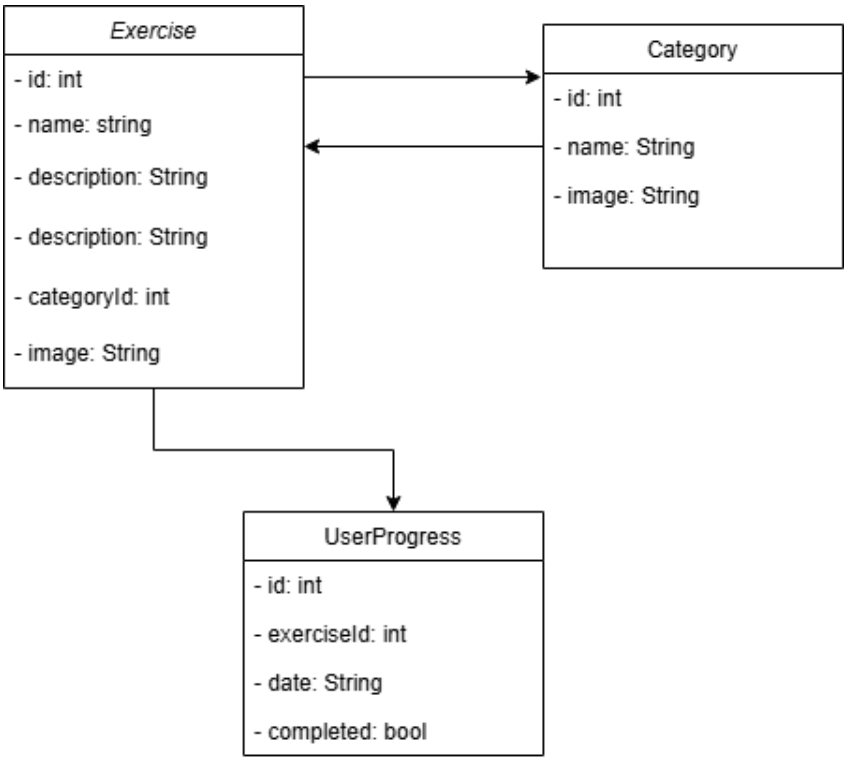

Рисунок 2.2 – Діаграма класів

Опис класів і зв’язків діаграми починається з класу Exercise, який представляє окрему вправу та містить такі поля, як id – ідентифікатор вправи, name – назва, description – текстовий опис, duration – тривалість у секундах, categoryId – зовнішній ключ до категорії, а також image як ілюстрацію до вправи.

Клас Category описує тип вправи або її розділ (наприклад, «Розтяжка» чи «Баланс») і має власні поля: id – ідентифікатор, name – назву категорії та image як фонову іконку.

Для відстеження прогресу користувача використовується модель UserProgress, яка зберігає історію виконаних вправ, включаючи id – унікальний запис, exerciseId – ідентифікатор вправи, дату виконання (date) та булеве значення completed, що вказує, чи була вправа виконана. Таким чином, ці класи пов’язані між собою через поля categoryId та exerciseId, забезпечуючи

структуру для організації інформації про вправи та відстеження користувацького прогресу.

Також для проектування було створено діаграму активностей (рис. 2.3).

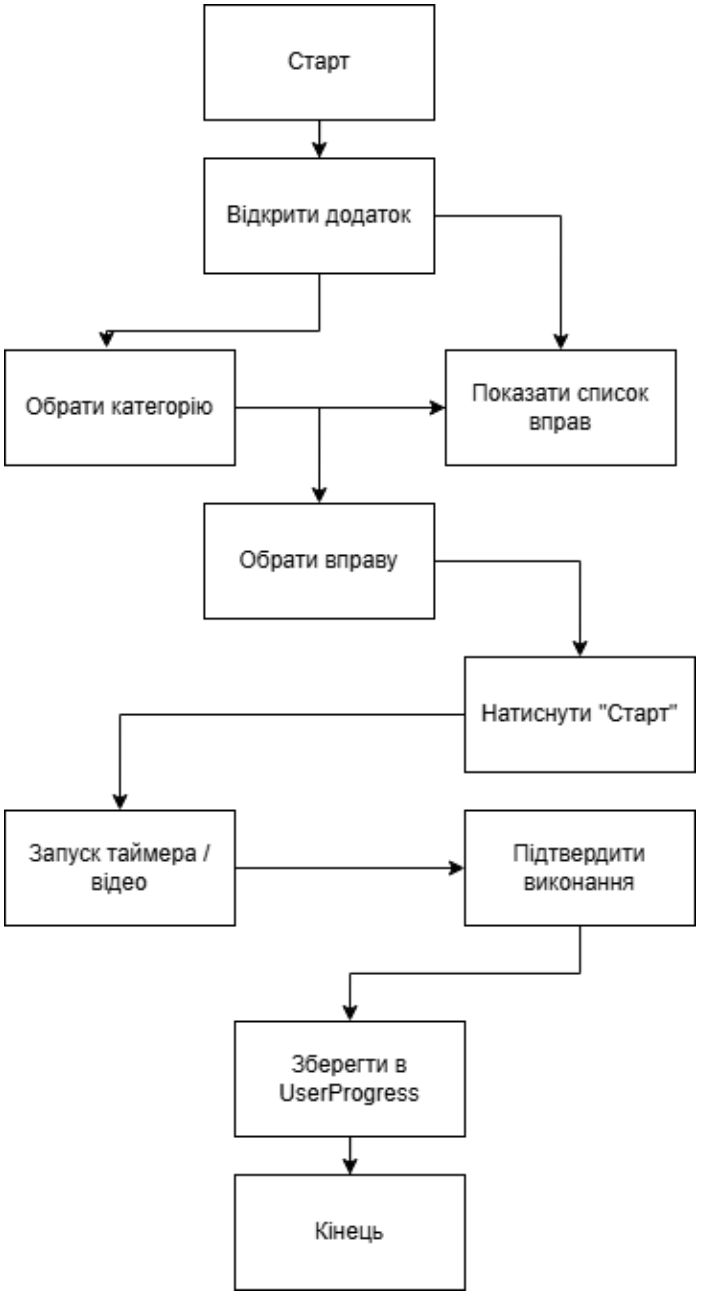

Рисунок 2.3 – Діаграма активностей

Завершальною діаграмою в процесі проектування є діаграма послідовностей (рис. 2.4).

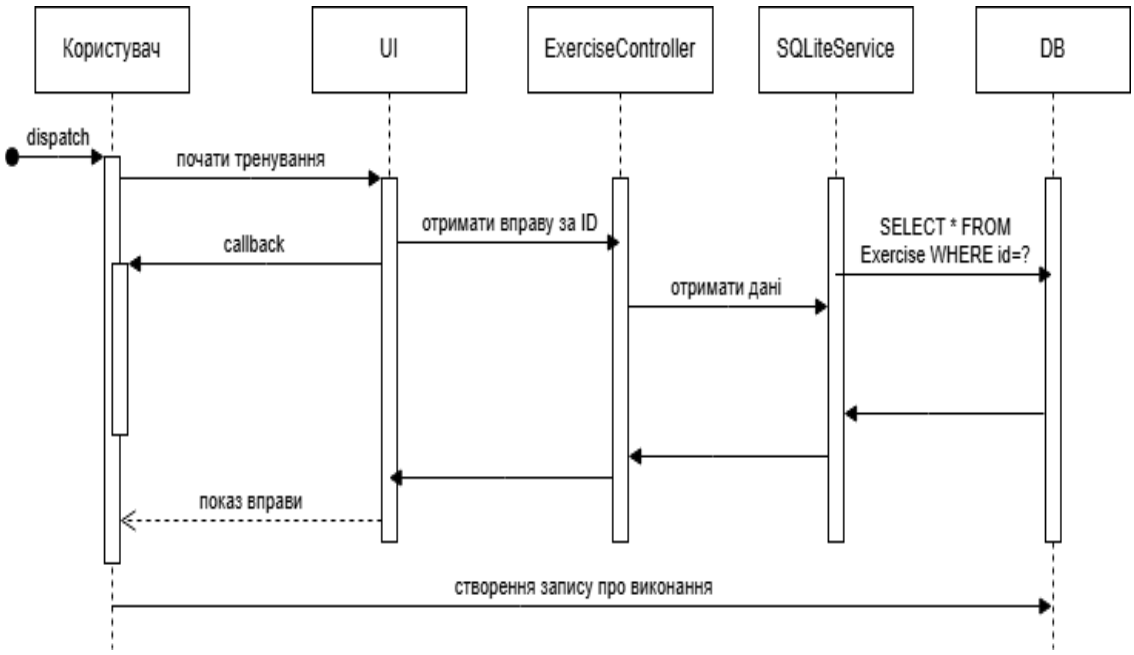

Рисунок 2.4 – Діаграма послідовностей

Також було проведено опис об’єктів діаграми послідовностей у таблиці 2.5.

Таблиця 2.5 – Опис об’єктів діаграми послідовностей

Об’єкт	Опис
Користувач	ініціатор дії через інтерфейс
UI	графічна оболонка, що реагує на взаємодію
ExerciseController	логіка, яка обробляє запити до БД
SQLiteService	сервісний шар, що працює з локальною БД
DB	сама база даних SQLite

У результаті побудови UML-моделей програмного забезпечення мобільного додатку для занять йогою, ми успішно досягли всіх цілей, які ставилися ще на етапі проєктування, що значно покращило загальний процес розробки. Проведене моделювання дало змогу не лише виявити ключові компоненти системи та оптимізувати їх взаємодію, а й ефективно уникнути дублювання логіки на ранньому етапі розробки, що заощадило час та ресурси.

Основні результати моделювання включають те, що діаграма варіантів використання (Use Case) дозволила виявити повний набір функцій, які очікує

кінцевий користувач, та правильно розставити пріоритети при реалізації. Діаграма класів допомогла чітко визначити основні сутності системи – вправи, категорії, прогрес, спроєктувати структуру бази даних SQLite, а також продумати відповідні Dart-моделі для Flutter. Діаграма активностей візуалізувала логіку роботи користувача з додатком: від відкриття до виконання вправи й збереження результату, що дозволило уникнути зайвих переходів у навігації. Діаграма послідовностей продемонструвала, як компоненти програми взаємодіють під час виконання основних дій, і дозволила оптимізувати структуру запитів до бази даних.

Переваги обраного підходу полягають у його високій формалізованості та структурованості. Використання UML-моделей дозволяє забезпечити уніфікований спосіб опису архітектури програмної системи, який є зрозумілим для розробників незалежно від їхнього досвіду чи специфіки проєкту. Це значно спрощує процес проєктування, підтримки та подальшої модифікації коду. Прозора структура системи, що досягається завдяки побудові діаграм, надає змогу чітко визначити межі між компонентами, виявити та мінімізувати залежності між ними, а також уникнути надмірного зв’язування, що особливо важливо для забезпечення стабільності та підтримуваності додатку. Крім того, модульна архітектура створює гнучке підґрунтя для розширення функціонал – у майбутньому можна легко додавати нові вправи, категорії занять, статистичні модулі чи елементи персоналізації без необхідності масштабних змін у базовій структурі. З метою досягнення ефективної організації логіки додатку було обрано архітектурний шаблон MVC (Model-View-Controller), який добре зарекомендував себе в мобільній розробці, а для збереження даних – легку, але потужну локальну базу даних SQLite, що забезпечує швидкий доступ до інформації та не потребує підключення до мережі.

Інтерфейс мобільного додатку для занять йогою розроблений відповідно до сучасних стандартів UI/UX (рис. 2.5). Його головна мета – забезпечити

інтуїтивно зрозумілу навігацію, привабливий візуальний стиль та швидкий доступ до функцій.

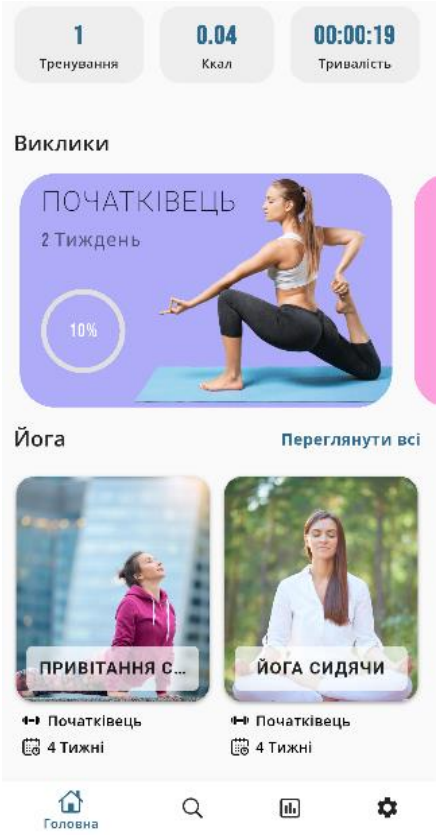

Рисунок 2.5 – Інтерфейс програми

Було реалізовано мінімалістичний дизайн з акцентом на візуальний контент (зображення тренувань, іконки, ілюстрації) та плавний користувацький досвід. Для цього використано дизайн-патерни мобільних застосунків у сфері фітнесу та медитації.

На старті користувача супроводжує серія екранів із поясненням функціоналу додатку:

– медитація;

– щоденна йога;

– практика йоги.

Головна сторінка відображає ключові метрики: кількість тренувань, витрачені калорії, тривалість.

Блок «Challenges» показує поточну програму, наприклад Beginner 7 Week, із відсотком виконання. Блок «Yoga» містить добірку вправ (Sun Salutation, Seated Yoga), з категорією складності та тривалістю. Мета створити швидкий доступ до прогресу й актуального контенту.

Каталог тренувань розділений за стилями (Hatha Yoga, Power Yoga), сезонами (Winter Season) та частинами тіла (Abs, Chest, Arms). Кожен пункт містить назву, опис, кількість вправ.Мета допомогти користувачу знайти саме ті вправи, які йому потрібні. На вікні тренування зображено вправи та кнопка «Go».

Потоки даних відбуваються між інтерфейсом, контролерами, моделями даних і базою даних (SQLite) з відображенням актуального стану UI.

Технології обслуговування інформаційних потреб Flutter забезпечує кросплатформену розробку з єдиною кодовою базою. SQLite дозволяє зберігати прогрес локально, що дає змогу використовувати додаток без Інтернету.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для розв’язання задачі створення мобільного додатку для занять йогою необхідно було визначити оптимальний інструментарій, який би забезпечував зручну реалізацію функціоналу, відповідність сучасним стандартам, а також мав потенціал для масштабування. Під час аналізу існуючих технологій були розглянуті як нативні, так і кросплатформенні рішення. Однак саме Flutter, як сучасний фреймворк від Google, показав себе як найбільш ефективний для поставлених цілей [14].

Вибір Flutter обумовлений його здатністю створювати високопродуктивні мобільні додатки з єдиною кодовою базою для Android та iOS. Це дозволяє значно зменшити час і вартість розробки, а також забезпечує стабільність і однаковий вигляд інтерфейсу на обох платформах. Flutter також

має зручну структуру, засновану на використанні віджетів, що дає змогу створювати кастомізовані й гнучкі UI-рішення [15].

Мовою програмування, що використовується в межах Flutter, є Dart. Вона поєднує в собі простоту синтаксису та підтримку об’єктно-орієнтованого програмування, що дозволяє ефективно організовувати логіку застосунку. Для зберігання даних, зокрема інформації про вправи, прогрес користувача та конфігурацію тренувань, застосовується локальна база даних SQLite. Її інтеграція з Flutter відбувається за допомогою спеціального плагіна sqflite, який підтримує основні CRUD-операції, транзакції та асинхронну обробку запитів [16].

Окрему увагу було приділено проєктуванню архітектури додатку. Основу склала модель MVC, у якій представлено чітке розділення між логікою взаємодії, візуалізацією та збереженням даних. Це дозволило організувати структуру коду в окремі модулі моделі, контролери, сервіси та представлення, що значно полегшує тестування та підтримку проєкту.

Для реалізації функціоналу було застосовано ряд методів, серед яких – обробка станів, навігація між екранами, валідація введених даних, а також побудова інформаційних потоків між шарами системи. Наприклад, при запуску тренування користувач обирає певну вправу, інформація про яку витягується з бази даних, і передається на екран відображення. Після завершення заняття результат зберігається знову у локальну БД, а головний екран оновлює статистику.

Розробка додатку не передбачала складної математичної моделі, однак було реалізовано базові алгоритми для розрахунку прогресу та калорій, витрачених під час тренування. Наприклад, для кожної вправи задається умовна інтенсивність та тривалість, на основі яких розраховується кількість витраченої енергії. Усі ці обчислення виконуються локально, без залучення зовнішніх сервісів.

РОЗДІЛ 3
РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ЗАНЯТЬ ЙОГОЮ З ВИКОРИСТАННЯМ FLUTTER

3.1 Практична реалізація об’єкта проєктування

Розробка мобільного додатку для занять йогою здійснювалася на базі SDK Flutter з використанням мови програмування Dart. У процесі реалізації було побудовано модульну архітектуру, де кожен окремий «.dart-файл» відповідає за певну частину функціоналу або представлення користувацького інтерфейсу. Це дозволило забезпечити високу організованість проєкту, простоту в підтримці та легкість у тестуванні.

Точка входу в додаток – файл main.dart. Саме тут ініціалізується основний віджет програми MyApp, де визначається тема оформлення, початковий маршрут (SplashScreen), а також базова конфігурація екранів. Нижче представлено фрагмент коду (ліст. 3.1), який демонструє запуск програми.

Лістинг 3.1 – Запуск програми

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  runApp(
    MyApp(),
  );
}
```

кінець лістингу 3.1

Файл SplashScreen.dart реалізує заставку, яка показується при запуску (рис. 3.1). Тут реалізована логіка переходу до основного вмісту після затримки або ініціалізації даних. Завдяки цьому користувач отримує візуальний сигнал про запуск програми, що покращує сприйняття швидкодії. Крім того, екран заставки забезпечує плавний перехід між запуском додатку та

основним інтерфейсом, створюючи більш професійний та завершений користувацький досвід.

Рисунок 3.1 – Заставка програми

Важливу роль у візуальній складовій проєкту відіграє Constants.dart, де централізовано зберігаються стилі, кольори, відступи, що забезпечує єдиний стиль усього додатку. Це дозволяє швидко вносити зміни у вигляд застосунку без потреби змінювати код в окремих компонентах. Також в Constants.dart було реалізована конвертація сантиметри в дюйми та фути (ліст. 3.2).

Лістинг 3.2 – Реалізація конвертації сантиметрів в дюйми та фути

```
static double feetAndInchToCm(double feet, double inch) {
    double meter;
```

```
    double f1 = (feet / 3.281);
    double i1 = (inch / 39.37);
    meter = f1 + i1;
    return meter*100;
  }

  static void meterToInchAndFeet(double cm, TextEditingController
ftController,
      TextEditingController inchController) {
    double meter = cm / 100;
    double inch = (meter * 39.37);
    double ft1 = inch / 12;
    int n = ft1.toInt();
    double in1 = ft1 - n;
    double in2;
    in2 = in1 * 12;
    ftController.text = n.round().toString();
    inchController.text = in2.round().toString();
  }
  static String meterToInchAndFeetText(double cm) {
    double meter = cm / 100;
    double inch = (meter * 39.37);
    double ft1 = inch / 12;
    int n = ft1.toInt();
    double in1 = ft1 - n;
    double in2;
    in2 = in1 * 12;
    return n.round().toString()+" ft "+in2.round().toString()+" in";
  }
```

кінець лістингу 3.2

Окремо варто виділити файл ModelExerciseList.dart, який містить структуру даних для опису тренувань. У ньому представлено клас моделі, що містить інформацію про вправу: зображення, опис, тривалість, рівень складності тощо. Ці моделі зв’язуються з відповідними віджетами, які відображають їх у вигляді карток на екрані. Приклад структури показаний в лістингу 3.3.

Лістинг 3.3 – Структура даних ModelExerciseList.dart

```
class ModelExerciseList {
  int? id;
  int? mainCatId;
  int? exerciseId;
```

```
  String? duration;
  ModelExerciseList.fromMap(dynamic objects) {
    id = objects['id'];
    mainCatId = objects['main_cat_id'];
    exerciseId = objects['exercise_id'];
    duration = objects['duration'];
  }
  Map<String, dynamic> toMap() {
    var map = new HashMap<String, dynamic>();
    map['duration'] = duration;
    map['exercise_id'] = exerciseId;
    map['main_cat_id'] = mainCatId;
    map['id'] = id;
    return map;
  }
}
```

кінець лістингу 3.3

Також був реалізований калькулятор індексу маси тіла (рис. 3.2). Індекс маси тіла це величина, що дозволяє оцінити ступінь відповідності маси людини до її зросту, й тим самим, непрямо оцінити, чи є маса недостатньою, нормальною, надмірною. У додатку А надано код з реалізацією цього калькулятора.

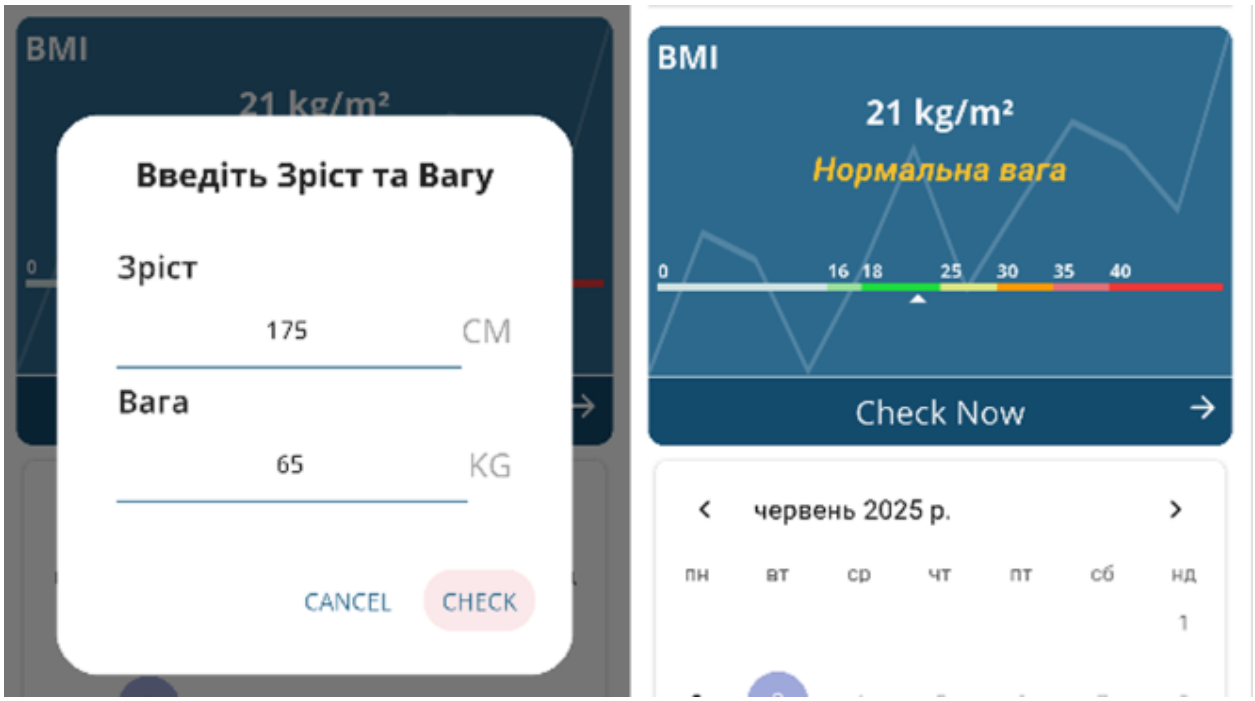

Рисунок 3.2 – Реалізація ІМТ

У рамках покращення взаємодії з користувачем у додатку реалізовано вступну частину (onboarding), яка допомагає новому користувачеві швидко ознайомитись із можливостями застосунку. За це відповідає файл IntroPage.dart, де за допомогою PageView реалізовано горизонтальний перегортуваний список екранів, кожен з яких представляє певну ідею – наприклад, користь йоги, щоденну практику або дихальні вправи. Користувач має можливість переглядати ці екрани, натискати кнопку Next, або ж Skip, щоб одразу перейти до основного інтерфейсу. Також для локального відображення всіх фото та тексту на IntoPage.dart було використано DataFile.dart в якому описано всі списки з локальними моделями. Всі локальні моделі описані в додатку Б.

Для зберігання даних про те, чи проходив користувач уже вступну частину, використовується файл PrefData.dart, що працює з локальним сховищем на пристрої (через бібліотеку shared_preferences). При першому запуску встановлюється булеве значення isIntroScreen = true, після чого при наступному відкритті додаток автоматично переспрямовує користувача вже не на IntroPage, а на головний екран. Це реалізовано так (ліст. 3.4).

Лістинг 3.4 – Реалізація IntroScreen

```
static setIsIntro(bool sizes) async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  await prefs.setBool(isIntro, sizes);
}
```

кінець лістингу 3.4

Самі дані для кожного вступного слайду зберігаються у вигляді об’єктів моделі IntroModel, яка описує заголовок, підзаголовок та ілюстрацію для кожного слайда. Це дозволяє легко масштабувати вступну частину – додавати нові слайди або локалізовані версії, не змінюючи логіку відображення.

Таким чином, початковий досвід користувача продуманий до деталей: показ інтро, збереження факту його перегляду, автоматичне перенаправлення

на основний інтерфейс – усе це підвищує зручність використання додатку, а також дає змогу новачкам швидко зорієнтуватись.

Одним із ключових функціональних елементів додатку є відображення списку вправ, які згруповані за категоріями або програмами тренувань. Для цього використовується структура на основі віджетів ListView, а також моделі ModelExerciseList, які описують кожну вправу окремо. У моделі зберігаються такі дані, як: назва вправи, шлях до зображення, тип (наприклад, «Seated Yoga», «Sun Salutation»), кількість тижнів у програмі, рівень складності тощо.

Відповідні віджети формують картки вправ, які динамічно будуються на основі даних. Картки містять зображення, текстову інформацію, а також значки тривалості або прогресу. Завдяки використанню компонентів GestureDetector реалізована логіка натискання на кожну вправу: при виборі картки користувач переходить до екрану детальної інформації або безпосередньо до вікна тренування.

У файлах, які відповідають за відображення інтерфейсу (HomeWidget.dart, ConstantWidget.dart), застосовано принцип розділення: є окремі методи для побудови елементів інтерфейсу (текстові блоки, іконки, контейнер з обводкою) та для обробки логіки переходу. Наприклад, компонент для побудови зображення з підписом і міткою рівня реалізується як окрема функція, що дозволяє повторно використовувати його в різних частинах програми.

Після вибору вправи, на відповідному екрані користувач може ознайомитися з інструкцією, побачити загальний таймер або прогрес по тижнях. Реалізація таймера або зворотного відліку в окремих вправах здійснюється за допомогою Timer із Dart core. Після завершення вправи програма автоматично оновлює статус у локальній базі або у сховищі SharedPreferences, зберігаючи інформацію про те, що дана вправа виконана.

Кожен тиждень тренування містить набір вправ, які позначаються цифрами (дні) та іконкою трофея, коли завершено. Це створює відчуття

поступового прогресу, що мотивує користувача не пропускати дні й завершувати програму.

Завдяки архітектурі, побудованій на віджетах, моделях і станах, додаток зберігає стабільну роботу при будь-яких переходах між екранами, а зміна стану (наприклад, виконано/не виконано) оновлюється миттєво й відображається без перезапуску.

У додатку реалізовано гнучку та зручну навігацію між екранами за допомогою стандартного механізму Navigator у Flutter. Використовуючи методи Navigator.push() і Navigator.pushReplacement(), користувач може переходити від стартового екрана до вступних слайдів, далі – до головної сторінки, екранів вправ або перегляду прогресу. Усі переходи виконуються плавно, із використанням анімацій, що покращує UX. Крім того, структура маршрутизації дозволяє легко розширювати проєкт новими екранами без потреби глобального переписування логіки.

Одним з ключових елементів реалізації є зберігання та відображення прогресу користувача. Для цього використовуються локальні сховища. Просту інформацію (наприклад, чи переглянуто вступний екран, або який тиждень пройдено) зберігає файл PrefData.dart через API SharedPreferences. Це дозволяє зберігати дані навіть після перезапуску програми. Прогрес у вправах або вибрані плани тренувань можуть бути також частково збережені у вигляді масивів JSON, що читаються/записуються локально. Такий підхід не потребує складної серверної частини, що важливо для офлайн-додатків.

Особливу увагу під час реалізації було приділено адаптивності інтерфейсу. Оскільки додаток орієнтований на використання на різних пристроях з різними екранами, було реалізовано систему масштабування через файл SizeConfig.dart. У ньому визначаються пропорційні коефіцієнти відносно ширини та висоти екрана. Наприклад, замість фіксованого розміру елемента задається blockSizeHorizontal * 4, що автоматично адаптує елемент під розмір дисплея. Завдяки цьому, застосунок однаково добре виглядає як на невеликих смартфонах, так і на планшетах.

Усі ці елементи – продумана навігація, локальне збереження, адаптивний дизайн – у комплексі забезпечують зручний, функціональний і стабільний мобільний додаток для щоденних занять йогою. Реалізація вказаних рішень підтверджує, що автор не тільки володіє мовою Dart і середовищем Flutter, але й здатен ефективно застосовувати їх на практиці для вирішення реальних прикладних задач.

У додатку для занять йогою використовується локальна база даних flutter_workout_ui_db.db, реалізована на основі SQLite. Вона забезпечує зберігання всіх основних даних: вправ, категорій, історії виконаних завдань, популярних програм, швидких тренувань тощо. Завдяки цьому додаток повністю функціонує в офлайн-режимі, що є критично важливим для мобільних користувачів.

База даних програми налічує 17 таблиць, кожна з яких виконує свою унікальну функцію. Серед них ключовими є tbl_main_category, яка зберігає перелік основних категорій тренувань, таких як «Йога для початківців» або «Силові вправи». tbl_exercise_list містить повний список вправ, пов’язаних з цими категоріями, включаючи назву, зображення, тип та тривалість.

Готові програми тренувань описані в tbl_workout_list, а tbl_workout_exercise_list слугує для зв’язування вправ з конкретними тренуваннями.

Для формування контенту головного екрану використовуються таблиці tbl_challenges_list, tbl_quick_workout, tbl_top_picks та tbl_discover, що відповідають за челенджі, швидкі тренування та рекомендовані добірки. Журнал активності користувача, що містить дату, вправу та статус виконання, зберігається у tbl_history.

Нарешті, tbl_exercise_detail надає додаткову інформацію про кожну вправу, таку як техніка виконання, рівень складності та детальний опис. Для прикладу на рисунку 3.3 зображено приклад таблиці з списком челенджів та їхніми фільтрами.

Таблиця: tbl_challenges_list

Filter in any column

	id	title	image	weeks	description	background
	Філ...	Фільтр	Фільтр	Фільтр	Фільтр	Фільтр
1	1	Початківець	full_body_challenge.png	7	Згинання коліна, також відоме як …	yoga.png
2	3	Розширений	plank_challenge.png	6	Виконайте це завдання, щоб виконати…	yoga.png
3	2	Середній …	push_up_challenge.png	7	Програма «300 присідань» дає вам …	yoga.png

Рисунок 3.3 – Приклад таблиці tbl_challenges_list

Зв’язки між таблицями організовано за допомогою зовнішніх ключів (exercise_id, workout_id, category_id), що дозволяє реалізувати реляційну структуру даних. Наприклад, користувач відкриває категорію – додаток вибирає відповідні вправи з tbl_exercise_list, а далі деталі – з tbl_exercise_detail.

Дані зчитуються й обробляються за допомогою бібліотеки sqflite у Flutter. Запити виконуються в асинхронному режимі, що не блокує UI. Отримані результати відображаються у вигляді списків, карток або інтерфейсних блоків. Усі зміни, зокрема завершення вправи або оновлення історії, також зберігаються до цієї бази даних.

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

У процесі розробки мобільного додатку для занять йогою було проведено комплексне тестування з метою перевірки коректності роботи функціоналу, стабільності інтерфейсу, правильності взаємодії з базою даних, а також сумісності з різними пристроями. Особливу увагу приділено тестуванню навігації, роботи з локальним сховищем, інтеграції бази SQLite та відображенню контенту на різних розмірах екранів.

Тестування проводилося як вручну, так і за допомогою інструментів, вбудованих у середовище розробки Android Studio. Зокрема, використовувався емулятор Android, а також реальні пристрої на базі Android 10 і Android 13 для перевірки адаптивності та стабільності додатку.

Було проведено комплексне тестування ключових функціональних сценаріїв, а саме ми перевірили ініціалізацію застосунку після інсталяції, включаючи коректне відображення SplashScreen. Також ми дослідили перегляд вступних слайдів та забезпечили збереження прогресу проходження. Навігація між основними екранами (вправи, програми, деталі) була ретельно протестована для підтвердження безперебійного користувацького досвіду, як і вибір тренування та запуск вправи. Також перевірено збереження виконаної вправи в історію користувача. Вкінці було протестовано операції читання та запису у базу даних SQLite (таблиці tbl_history, tbl_exercise_list) та збереження налаштувань за допомогою механізму SharedPreferences. Ці тести підтвердили стабільність та коректність роботи всіх основних компонентів системи.

Під час ручного тестування ми виявили кілька незначних недоліків. На екранах з невеликою роздільною здатністю, до 720p, деякі елементи інтерфейсу накладалися один на одного. При першому запуску спостерігалася затримка в 1-2 секунди під час завантаження вступних слайдів. Також, після проходження онбордингу, програма іноді не зберігала стан, що було спричинено відсутністю await у методі запису SharedPreferences.

Ми успішно усунули всі ці недоліки. Для адаптації інтерфейсу користувача ми доопрацювали логіку SizeConfig.dart. Щоб оптимізувати завантаження вступних слайдів, ми використали precacheImage для зображень з assets. А для коректного збереження стану проходження онбордингу, ми додали необхідний await у відповідний метод.

Мінімальні системні вимоги:

– Android 6.0 або iOS 12.0;
– 1 ГБ оперативної пам’яті;
– 100 МБ вільного місця;
– екран з мінімальною шириною 320 dp.

Рекомендовані параметри:

– Android 9.0 або вище;
– 2+ ГБ оперативної пам’яті;

– 200+ МБ вільного місця;

– роздільна здатність 720p або вища.

У результаті проведеного тестування було підтверджено, що мобільний додаток функціонує стабільно, виконує всі основні завдання відповідно до поставлених вимог, а виявлені недоліки були своєчасно виправлені.

3.3 Реалізація інтерфейсу

Реалізація користувацького інтерфейсу (UI) мобільного додатку для занять йогою була спрямована на створення інтуїтивно зрозумілого, візуально привабливого та функціонального середовища, що сприяє ефективній взаємодії користувача з програмою. Основний акцент було зроблено на забезпеченні послідовного дизайну, адаптивності макетів та зручної навігації між основними функціями.

Інтерфейс додатку побудований на компонентному підході, де кожен елемент UI є віджетом, що дозволяє досягти високої модульності та гнучкості.

На рівні кореневого віджета (MyApp у main.dart) встановлюється глобальна тема оформлення, яка визначає основні кольори (primaryColor, accentColor) та застосовується до всіх компонентів UI. Це забезпечує візуальну єдність та спрощує подальші зміни стилів. Також тут налаштовується локалізація та система обробки локальних сповіщень, використовуючи реактивні потоки (BehaviorSubject) для відображення повідомлень користувачу (наприклад, через CupertinoAlertDialog).

Основна навігація додатку реалізована за допомогою нижньої навігаційної панелі (BottomNavigationBar), що є частиною віджета HomeWidget. HomeWidget виступає контейнером для чотирьох основних розділів додатку як головний екран (TabHome), екран «Discover» (TabDiscover), екран активності/статистики (TabActivity) та екран налаштувань (TabSettings). Кожен з цих розділів є самостійним віджетом, що дозволяє ефективно керувати їхнім станом та вмістом. Обробка натискання кнопки «назад» реалізована за

допомогою WillPopScope, що дозволяє контролювати повернення на головний екран або вихід з додатку.

Для динамічного відображення контенту в інтерфейсі використовуються спеціально розроблені моделі даних. Наприклад, IntroModel (визначена в IntroModel.dart) інкапсулює заголовок, опис та шлях до зображення для кожного слайда вступного екрану. Аналогічно, інші частини інтерфейсу (списки тренувань, виклики, стилі йоги) отримують свої дані з файлів, що містять відповідні моделі та методи їхнього завантаження (наприклад, з DataFile.dart, який надає challengesMainCat(), getWorkoutList(), getDiscoverList() тощо). Це забезпечує чисте розділення між логікою даних та їхнім візуальним представленням.

Дизайн інтерфейсу додатку має ключове значення для забезпечення привабливості та зручності використання. Він базується на принципах чистоти, інтуїтивності та візуальної привабливості.

Кольори додатка визначені централізовано (ColorCategory.dart, Constants.dart), що забезпечує єдиний візуальний стиль. Використання шістнадцяткових кодів та функцій для динамічної зміни відтінків кольорів (getColorFromHex, darken, brighten у Constants.dart) надає гнучкість у створенні візуальних ефектів та адаптації елементів UI.

Для забезпечення візуальної ієрархії та читабельності використовується набір визначених шрифтів (OpenSans, SecularOne, MeriendaOne, Bebasneue з Constants.dart). Допоміжні віджети та методи (getCustomTextFont, getTextWidgetWithFont у ConstantWidget.dart) дозволяють легко застосовувати ці шрифти з різними розмірами, вагою та кольорами по всьому додатку, підтримуючи консистентність типографіки.

Адаптивний дизайн є фундаментальним для роботи на різних пристроях. Додаток реалізує систему масштабування інтерфейсних елементів через SizeConfig.dart, який обчислює відносні одиниці виміру на основі розмірів екрану (screenWidth, screenHeight, blockSizeHorizontal, safeBlockVertical). ConstantWidget.dart використовує ці пропорційні коефіцієнти для створення

адаптивних віджетів та розрахунку розмірів (наприклад, getScreenPercentSize), що гарантує коректне відображення макетів на будь-якому пристрої, включаючи врахування системних відступів.

Інтерфейс додатку використовує різноманітні візуальні елементи та композиції віджетів для створення привабливого та функціонального досвіду користувача. Переважає чистий, мінімалістичний дизайн з використанням карток, заокруглених кутів та чітких візуальних розділень між секціями.

Вступний екран, як видно на рисунку 3.4, використовує PageView.builder для горизонтально прокручуваних слайдів. Кожен слайд має фонову ілюстрацію та текстовий контент (заголовок та опис), накладені за допомогою Stack. Прогрес онбордингу візуалізується серією круглих індикаторів, де активний слайд виділяється заповненою акцентним кольором крапкою. Навігація здійснюється кнопкою «Next» або «Skip», що дозволяє перейти до основного інтерфейсу після першого запуску додатка.

Рисунок 3.4 – Екран онбордингу

Головний екран є центральним елементом взаємодії, як показано на рисунку 3.5. У верхній частині відображаються картки зі статистикою («Тренування», «Ккал», «Тривалість»), забезпечуючи швидкий огляд прогресу. Розділи «Виклики» та «Йога» реалізовані як горизонтально прокручувані списки (CarouselSlider для викликів та GridView для йоги). Картки в цих списках містять зображення, заголовки, описи та, для викликів, кругові індикатори прогресу (CircularPercentIndicator). Нижня навігаційна панель надає доступ до основних розділів додатка.

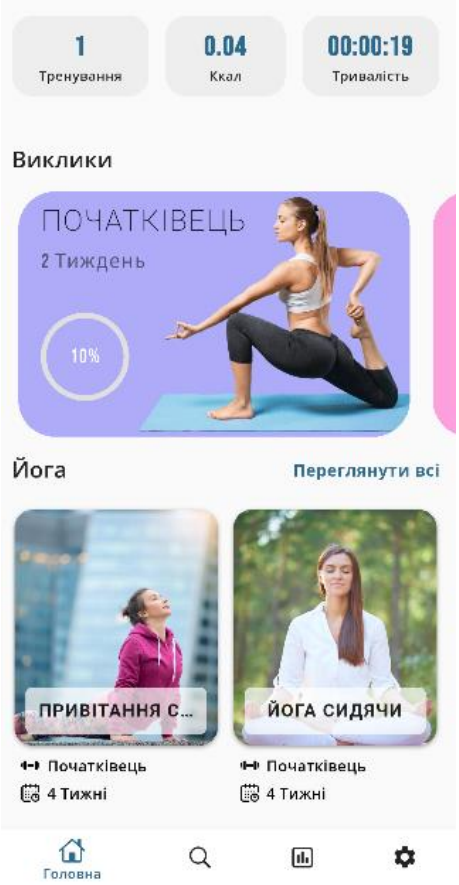

Рисунок 3.5 – Головний екран

Рисунок 3.6 демонструє екран, що відображає ключові показники активності. Круговий індикатор прогресу візуалізує щоденну ціль калорій. Розділ індексу маси тіла (BMI) показує значення BMI, його інтерпретацію («Недостатня вага», «Нормальна вага») та візуальну шкалу (MyAssetsBar), що

дозволяє користувачеві швидко оцінити свій стан. TableCalendar забезпечує зручну навігацію по датах та перегляд статистики за конкретні дні.

Рисунок 3.6 – Екран Активності та статистики

Екран вправ демонструє логічно структуровану організацію тренувального процесу, розподіленого за тижнями та днями. Такий підхід дозволяє користувачеві легко орієнтуватися у своєму прогресі та плануванні занять. Завершені дні позначаються зеленим кольором, що наочно сигналізує про виконання вправ у конкретну дату, а повністю завершені тижні відзначаються спеціальною іконкою кубка, яка слугує мотивувальним елементом і додатковим візуальним підкріпленням досягнень користувача. Це створює ефект гейміфікації, заохочуючи регулярні заняття та послідовне проходження тренувальної програми (рис. 3.7).

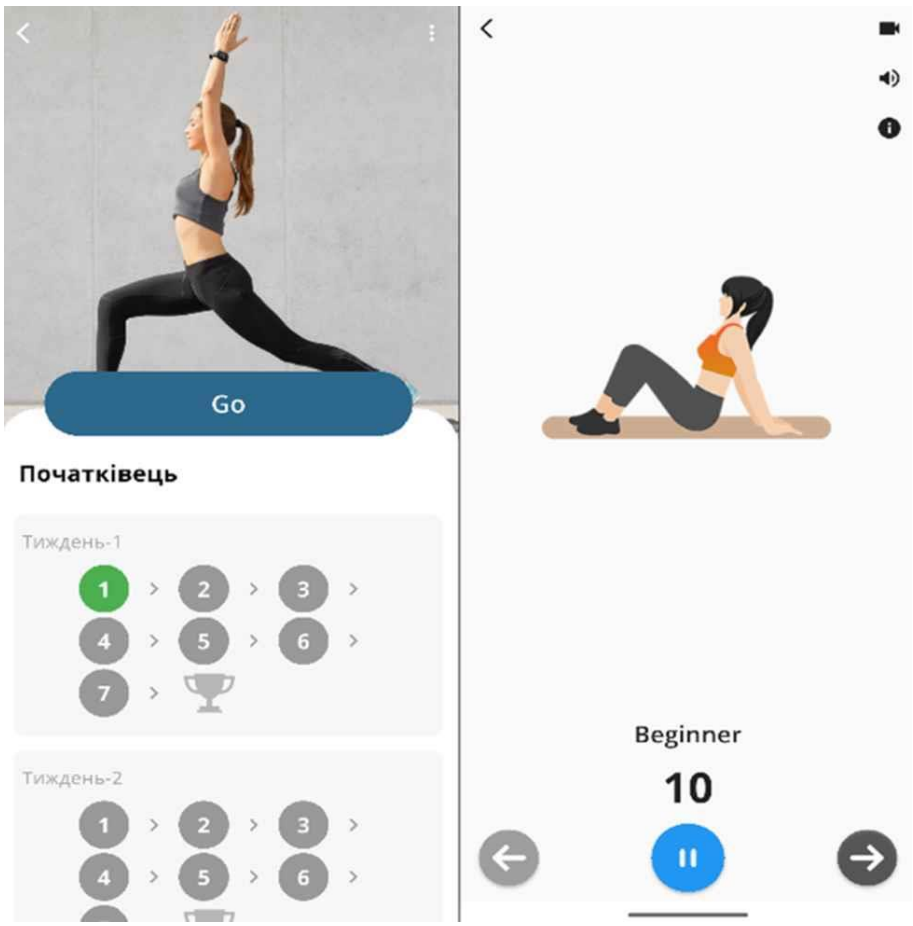

Рисунок 3.7 – Екран деталей тренування та виконання вправи

Екран налаштувань надає користувачеві можливість персоналізувати додаток. Серед ключових налаштувань тривалість відпочинку між вправами, щоденна ціль калорій, налаштування звукового супроводу та озвучування інструкцій (Text-to-Speech), вибір системи одиниць виміру (метрична/імперська), функція «тримати екран увімкненим», а також можливості для зворотного зв’язку та поширення додатку. Всі ці налаштування зберігаються локально за допомогою PrefData.

Для введення даних (наприклад, зріст, вага, щоденна мета калорій) та системних сповіщень використовуються стандартні AlertDialog та CupertinoAlertDialog, що забезпечують відповідний нативний вигляд на різних платформах. Ці діалоги мають послідовний дизайн із заокругленими кутами та чіткими кнопками «Відміна» та «Підтвердження».

Для коротких, неблокуючих повідомлень користувачеві (наприклад, підтвердження дії) застосовується Fluttertoast.

Більшість кнопок та інтерактивних елементів (InkWell, TextButton) мають візуальні ефекти (наприклад, тіні, зміна кольору при натисканні), що підвищує відчуття інтерактивності.

Таким чином, реалізація користувацького інтерфейсу в мобільному додатку для йоги базується на принципах модульності, адаптивності та візуальної привабливості, забезпечуючи інтуїтивно зрозумілий та функціональний досвід взаємодії.

3.4 Розробка інсталяційного пакета

Розгортання мобільного додатку, створеного за допомогою Flutter, здійснюється шляхом формування інсталяційного пакета у форматі APK (для Android) або IPA (для iOS). У цьому проєкті реалізовано генерацію Android-пакета, оскільки саме ця платформа використовувалась як основна цільова.

Процес складання інсталяційного пакета здійснюється через вбудовані засоби Flutter, зокрема команду «flutter build apk –release» [17].

У результаті формується файл app-release.apk, який розміщується в каталозі build/app/outputs/flutter-apk/. Цей файл є самостійним інсталяційним пакетом, який можна встановити на будь-який Android-пристрій без потреби у Google Play.

Структура проекти складається з компонентів, які розміщені в межах папки app. У корені проєкту знаходяться службові каталоги .gradle та .kotlin, що відповідають за конфігурацію середовища розробки та інструментів.

Основна логіка додатку зосереджена в директорії src. Вона містить дві папки debug – для файлів, що використовуються під час налагодження, та main – основну структуру застосунку. Усередині main розміщено каталоги java та kotlin, де зберігається програмний код, а також папку res, яка містить ресурси інтерфейсу користувача, зображення, рядки локалізації тощо. Окремо

виділений файл AndroidManifest.xml, який визначає загальну конфігурацію додатку, зокрема його компоненти, дозволи та точки запуску.

Папка .cxx, присутня в структурі, вказує на можливість використання нативного коду в проєкті. Крім того, директорія profile може бути призначена для розміщення конфігурацій або даних, пов'язаних із профілюванням та оптимізацією продуктивності. Така структура є стандартною для Android-проєктів і забезпечує зручну організацію як для розробки, так і для супроводу застосунку. На рисунку 3.8 показано структуру проєкту.

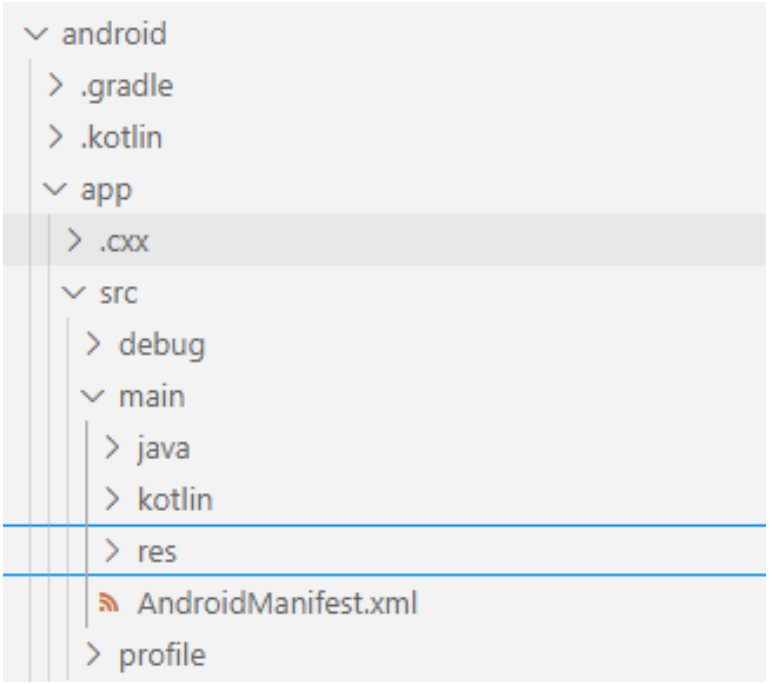

Рисунок 3.8 – Структура Проекту після flutter build apk

Усі Dart-файли, ресурси та залежності автоматично включаються в APK під час збірки. Flutter збирає інтерпретований код і транслює його в нативний ARM-код, який виконується напряму на пристрої.

Для створення ярлика на головному екрані Android використовується налаштування в AndroidManifest.xml, де визначено назву (ліст. 3.5) [18].

Лістинг 3.5 – Налаштування назви та іконки програми

```
<application
        android:label="Yoga Workout"
```

```
        android:name="${applicationName}"
        android:icon="@mipmap/ic_launcher">
```

кінець лістингу 3.5

Іконка ic_launcher створюється автоматично або налаштовується вручну за допомогою утиліти flutter_launcher_icons. Вона генерує усі необхідні розміри іконок для різних щільностей екранів і додає їх у папки res/mipmap-*.

Оскільки Flutter-додатки не взаємодіють з системним реєстром Android напряму (на відміну від Windows-додатків), збереження налаштувань та конфігурацій користувача реалізовано через локальні сховища [19]:

– SharedPreferences – для простих налаштувань (булеві значення, строки);

– SQLite – для структурованих даних (історія тренувань, вправи, прогрес).

Ці дані зберігаються в системних директоріях Android (/data/shared_prefs/ або /databases/), куди має доступ лише сам додаток.

Остаточний .apk-файл можна надіслати користувачу для інсталяції напряму або підготувати для розміщення на Google Play (через .aab пакет). У майбутньому додаток може бути підписаний цифровим сертифікатом для офіційного публічного релізу.

ВИСНОВКИ

У межах даної кваліфікаційної роботи було успішно реалізовано повний цикл розробки мобільного додатку для занять йогою з використанням Flutter, що відповідає поставленим цілям та завданням. Проведена робота охопила всі етапи від аналізу предметної області до формування інсталяційного пакета, демонструючи практичне застосування сучасних підходів до розробки програмного забезпечення.

Відповідно до завдання, було проведено глибокий аналіз предметної області та визначено всі функціональні, нефункціональні та технічні вимоги до розроблюваного додатку. Це дозволило сформувати чітке бачення кінцевого продукту, врахувати потреби потенційних користувачів та забезпечити відповідність системи актуальним стандартам якості та безпеки.

На основі проведеного аналізу було обґрунтовано вибір засобів та методів реалізації. Використання Flutter та мови Dart забезпечило кросплатформенність рішення, високу продуктивність та ефективність розробки. Архітектурна модель MVC у поєднанні з локальною базою даних SQLite була обрана для забезпечення модульності, гнучкості та легкості подальшої підтримки системи.

Практична реалізація додатку була здійснена відповідно до обраного технологічного стека. Було побудовано модульну структуру проекту з чітким розділенням відповідальності між файлами. Реалізовано основні функціональні елементи, такі як каталог вправ, персональний кабінет та механізми відстеження тренувального процесу, що підтверджує досягнення поставлених цілей розробки.

Особливу увагу було приділено локальному збереженню даних. Для цього застосовано гібридний підхід. SQLite використовується для зберігання структурованих даних про вправи, категорії, історії тренувань, а SharedPreferences – для невеликих користувацьких налаштувань (наприклад,

стан онбордингу). Такий підхід забезпечує повну функціональність додатку в офлайн-режимі, що є значною перевагою для користувачів.

Впровадження адаптивного інтерфейсу, що коректно відображається на різних розмірах екранів, стало ключовим аспектом UX-дизайну. Завдяки розробленій системі масштабування елементів через SizeConfig.dart та ConstantWidget.dart, інтерфейс автоматично адаптується до різних роздільних здатностей та орієнтацій пристроїв, зберігаючи пропорції та естетичний вигляд. Візуальний дизайн характеризується мінімалізмом, акцентом на візуальний контент та інтуїтивно зрозумілою навігацією.

Проведено комплексне тестування функціоналу, продуктивності та стабільності роботи додатку. Тестування охоплювало перевірку всіх ключових сценаріїв взаємодії, включаючи навігацію, збереження даних та відображення контенту на різних екранах. Виявлені на етапі тестування недоліки були оперативно усунені, що гарантувало стабільну та коректну роботу системи.

На завершальному етапі роботи було сформовано інсталяційний пакет APK для подальшого розповсюдження додатку на платформі Android. Це підтверджує готовність продукту до впровадження та використання кінцевими користувачами.

Таким чином, у ході кваліфікаційної роботи було повністю реалізовано мобільний додаток для занять йогою, який відповідає всім заявленим вимогам та демонструє високий рівень практичних навичок у розробці мобільних рішень за допомогою Flutter.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Statista. Fitness App Market Overview. URL: https://www.statista.com/ topics/ 11491/fitness-apps-worldwide (дата звернення: 11.02.2025).

2. Down Dog Official Website. URL: https://www.downdogapp.com (дата звернення: 12.02.2025).

3. The #1 Rated Yoga App, Down Dog, Releases Free Meditation App. URL: https://cdn.nwe.io/files/x/09/bf/8efb95104671063a869b43052a34.jpg (дата звернення 12.02.2025).

4. Daily Yoga App Features. URL: https://www.dailyyoga.com (дата звернення: 12.02.2025).

5. Daily Yoga. URL: https://www.applovin.com/wpcontent/uploads/2022/08/img_successstories_header_dailyyoga.png (дата звернення: 12.02.2025).

6. Yoga Studio by Gaiam. URL: https://yogastudioapp.com (дата звернення: 12.02.2025).

7. 5 Best Yoga Apps while working from home. URL: https://employmenthero.com/sg/wp- content/uploads/2020/06/YogaStudio-min-768x529-1.png (дата звернення: 12.02.2025).

8. Asana Rebel – Yoga Inspired Fitness. URL: https://www.asanarebel.com (дата звернення: 12.02.2025).

9. Asana Rebel Review for 2024. URL: https://fitnessdrum.com/wp-content/uploads/2022/02/Asana-Rebel-min.png (дата звернення: 12.02.2025).

10. Developer Economics. State of Mobile App Development. URL: https://developereconomics.net/reports (дата звернення: 05.03.2025).

11. Google Patents. Exercise recognition using computer vision (US20210166421A1, EP3895594A1). URL: https://patents.google.com/ patent/EP3895594A1/en (дата звернення: 07.03.2025).

12. Object Management Group (OMG). Unified Modeling Language (UML) Specification. URL: https://www.omg.org/spec/UML (дата звернення: 07.05.2025).

13. Flutter Documentation. Dart Language Tour. URL: https://dart.dev/guides/language/language-tour (дата звернення: 10.04.2025).

14. Android Authority. Best Yoga Apps for Android. URL: https://www.androidauthority.com/best-yoga-apps-android-347888 (дата звернення: 10.04.2025).

15. Google Developers. Flutter – Build apps for any screen. URL: https://flutter.dev (дата звернення: 11.04.2025).

16. Android Developers. Manifest file – application element. URL: https://developer.android.com/guide/topics/manifest/application-element (дата звернення: 14.04.2025).

17. Flutter Documentation. Widgets Catalog. URL: https://docs.flutter.dev/development/ui/widgets (дата звернення: 20.04.2025).

18. Pub.dev. sqflite: Flutter plugin for SQLite. URL: https://pub.dev/packages/sqflite (дата звернення: 20.04.2025).

19. Flutter Documentation. Persistence: Saving data. URL: https://docs.flutter.dev/cookbook/persistence (дата звернення: 20.04.2025).

ДОДАТКИ

Додаток А – Код калькулятора індексу маси тіла

```
void showWeightHeightDialog(BuildContext contexts) async {
    return showDialog(
      context: contexts,
      builder: (context) {
        return StatefulBuilder(
          builder: (context, setState) {
            return AlertDialog(
              title: getMediumBoldTextWithMaxLine(
                  "Введіть Зріст та Вагу", Colors.black87, 1),
              content: Container(
                width: 300.0,
                padding:
                    EdgeInsets.only(top: 15, bottom: 15, left: 15,
right: 15),
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.start,
                  crossAxisAlignment: CrossAxisAlignment.start,
                  mainAxisSize: MainAxisSize.min,
                  children: <Widget>[
                    getCustomText("Зріст", Colors.black87, 1,
TextAlign.start,
                        FontWeight.w600, 20),
                    SizedBox(
                      height: 2,
                    ),
                    Row(
                      mainAxisSize: MainAxisSize.max,
                      children: [
                        Expanded(
                          child: TextField(
                            keyboardType: TextInputType.number,
                            textAlign: TextAlign.center,
                            cursorColor: accentColor,
                            decoration: InputDecoration(
                                enabledBorder: UnderlineInputBorder(
                                  borderSide: BorderSide(color:
accentColor),
                                ),
                                focusedBorder: UnderlineInputBorder(
                                  borderSide: BorderSide(color:
accentColor),
                                )),
                            style: TextStyle(
                                fontWeight: FontWeight.normal,
                                fontSize: 15,
                                color: Colors.black,
                                decorationColor: accentColor,
```

```
                                fontFamily: Constants.fontsFamily),
                            controller: myController,
                          ),
                          flex: 1,
                        ),
                        Visibility(
                          child: Text(
                            " , ",
                            style: TextStyle(
                                fontWeight: FontWeight.normal,
                                fontSize: 15,
                                color: Colors.black,
                                decorationColor: accentColor,
                                fontFamily: Constants.fontsFamily),
                          ),
                          visible: (!isKg) ? true : false,
                        ),
                        Visibility(
                          child: Expanded(
                            child: TextField(
                              keyboardType: TextInputType.number,
                              textAlign: TextAlign.center,
                              cursorColor: accentColor,
                              decoration: InputDecoration(
                                  enabledBorder: UnderlineInputBorder(
                                    borderSide: BorderSide(color:
accentColor),
                                  ),
                                  focusedBorder: UnderlineInputBorder(
                                    borderSide: BorderSide(color:
accentColor),
                                  )),
                              style: TextStyle(
                                  fontWeight: FontWeight.normal,
                                  fontSize: 15,
                                  color: Colors.black,
                                  decorationColor: accentColor,
                                  fontFamily: Constants.fontsFamily),
                              controller: myControllerIn,
                            ),
                            flex: 1,
                          ),
                          visible: (!isKg) ? true : false,
                        ),
                        getMediumNormalTextWithMaxLine((isKg) ? "CM" :
"FT/In",
                            Colors.grey, 1, TextAlign.start)
                      ],
                    ),
                    SizedBox(
                      height: 7,
```

```
                    ),
                    getCustomText("Bara", Colors.black87, 1,
TextAlign.start,
                        FontWeight.w600, 20),
                    SizedBox(
                      height: 2,
                    ),
                    Row(
                      mainAxisSize: MainAxisSize.max,
                      children: [
                        Expanded(
                          child: TextField(
                            keyboardType: TextInputType.number,
                            textAlign: TextAlign.center,
                            cursorColor: accentColor,
                            decoration: InputDecoration(
                                enabledBorder: UnderlineInputBorder(
                                  borderSide: BorderSide(color:
accentColor),
                                ),
                                focusedBorder: UnderlineInputBorder(
                                  borderSide: BorderSide(color:
accentColor),
                                )),
                            style: TextStyle(
                                fontWeight: FontWeight.normal,
                                fontSize: 15,
                                color: Colors.black,
                                decorationColor: accentColor,
                                fontFamily: Constants.fontsFamily),
                            controller: myControllerWeight,
                          ),
                          flex: 1,
                        ),
                        getMediumNormalTextWithMaxLine(
                            "KG", Colors.grey, 1, TextAlign.start)
                      ],
                    ),
                  ],
                ),
              ),
              actions: [
                new TextButton(
                    child: Text(
                      'Відміна',
                      style: TextStyle(
                          fontFamily: Constants.fontsFamily,
                          fontSize: 15,
                          color: accentColor,
                          fontWeight: FontWeight.normal),
                    ),
```

```
                    onPressed: () {
                      Navigator.pop(context);
                    }),
                new TextButton(
                    // color: lightPink,
                    style: TextButton.styleFrom(backgroundColor:
lightPink),
                    child: Text(
                      'Перевірка',
                      style: TextStyle(
                          color: accentColor,
                          fontFamily: Constants.fontsFamily,
                          fontSize: 15,
                          fontWeight: FontWeight.normal),
                    ),
                    onPressed: () {
                      if (myController.text.isNotEmpty) {
                        if (isKg) {
                          height = double.parse(myController.text);


                        } else {
                          double inch = 0;
                          if (myControllerIn.text.isNotEmpty) {
                            inch = double.parse(myControllerIn.text);
                          }
                          double feet =
double.parse(myController.text);
                          double cm = Constants.feetAndInchToCm(feet,
inch);
                          height = cm;
                        }
                        PrefData().addHeight(height);
                      }

                      if (myControllerWeight.text.isNotEmpty) {

                        double weight1 =
double.parse(myControllerWeight.text);
                        if (isKg) {
                          weight = weight1;
                          PrefData().addWeight(weight1);
                        } else {
                          weight = Constants.poundToKg(weight1);
                          PrefData().addWeight(weight);
                        }
                      }

                      Navigator.pop(context, weight);
                    }),
              ],
```

```
          );
        },
      );
    },
  ).then((value) {
    getBmiVal();
  });
}
```

Додаток Б – Код всіх моделей з DataFile

```
class DataFile {
  static List<IntroModel> getIntroModel(BuildContext context) {
    List<IntroModel> introList = [];

    IntroModel mainModel = new IntroModel();
    mainModel.id = 1;
    mainModel.name = "Щоденна йога";
    mainModel.image = "intro_1.png";
    mainModel.desc = "Чи ваші фізичні вправи та релаксація сприяють
здоров'ю?";
    introList.add(mainModel);

    mainModel = new IntroModel();
    mainModel.id = 2;
    mainModel.name = "Заняття йогою";
    mainModel.image = "intro_2.png";
    mainModel.desc = "Медитація – ключ до продуктивності, щастя та
довголіття";
    introList.add(mainModel);

    mainModel = new IntroModel();
    mainModel.id = 3;
    mainModel.name = "Практикуйте йогу";
    mainModel.image = "intro_3.png";
    mainModel.desc = "Виконуйте фізичні вправи та розслаблення";
    introList.add(mainModel);

    return introList;
  }

  static List<ModelChallengesMainCat> challengesMainCat() {
    List<ModelChallengesMainCat> list = [];

    ModelChallengesMainCat modelChallengesMainCat =
        new ModelChallengesMainCat();
    modelChallengesMainCat.image = "asthanga_yoga.jpg";
    modelChallengesMainCat.icon = "yoga_images.png";
    modelChallengesMainCat.background = "asthanga_yoga.jpg";
    modelChallengesMainCat.title = "Початківець";
    modelChallengesMainCat.weeks = 2;
    modelChallengesMainCat.id = 1;
    list.add(modelChallengesMainCat);

    modelChallengesMainCat = new ModelChallengesMainCat();
    modelChallengesMainCat.image = "asthanga_yoga.jpg";
    modelChallengesMainCat.background = "asthanga_yoga.jpg";
    modelChallengesMainCat.title = "Середній рівень";
```

```
    modelChallengesMainCat.weeks = 7;
    modelChallengesMainCat.id = 2;
    modelChallengesMainCat.icon = "hathha_yoga.png";
    list.add(modelChallengesMainCat);

    modelChallengesMainCat = new ModelChallengesMainCat();
    modelChallengesMainCat.image = "asthanga_yoga.jpg";
    modelChallengesMainCat.icon = "yoga_imges5.png";
    modelChallengesMainCat.background = "asthanga_yoga.jpg";
    modelChallengesMainCat.title = "Розширений";
    modelChallengesMainCat.weeks = 6;
    modelChallengesMainCat.id = 3;
    list.add(modelChallengesMainCat);

    return list;
  }

  static List<ModelDiscover> getDiscoverList() {
    List<ModelDiscover> list = [];

    ModelDiscover model = new ModelDiscover();
    model.id = 1;
    model.title = "Зимні тренування";
    model.description =
        "Це практика, що поєднує м’які розтяжки та дихальні вправи для
підтримки тепла, енергії й внутрішнього балансу в холодну пору року.";
    model.image = "yoga_imges5.png";

    list.add(model);

    model = new ModelDiscover();
    model.id = 2;
    model.title = "Осінні тренування";
    model.description =
        "сприяє заземленню, зміцненню імунітету й адаптації до змін
природи";
    model.image = "mediation.png";
    list.add(model);

    model = new ModelDiscover();
    model.id = 3;
    model.title = "Літні тренування";
    model.description =
        "Зосереджується на охолодженні тіла, гнучкості й легкості
дихання в спеку";
    model.image = "leg_1.png";
    list.add(model);
    model = new ModelDiscover();
    model.id = 4;
    model.title = "Весняні тренування";
    model.description =
```

```
        "Оновлює тіло й розум, допомагаючи пробудити енергію після
зимової сплячки";
    model.image = "mediation_2.png";
    list.add(model);

    return list;
  }

  static List<ModelPopularWorkout> getPopularList() {
    List<ModelPopularWorkout> list = [];

    ModelPopularWorkout model = new ModelPopularWorkout();
    model.id = 1;
    model.image = "abs_workout.jpg";
    model.name = "Прес";
    list.add(model);

    model = new ModelPopularWorkout();
    model.id = 2;
    model.image = "glutes_workout.jpg";
    model.name = "Сідниці";
    list.add(model);

    model = new ModelPopularWorkout();
    model.id = 3;
    model.image = "legs_workout.jpg";
    model.name = "Ноги";
    list.add(model);

    model = new ModelPopularWorkout();
    model.id = 4;
    model.image = "back_workout.jpg";
    model.name = "Спина";
    list.add(model);

    model = new ModelPopularWorkout();
    model.id = 5;
    model.image = "chest_workout.jpg";
    model.name = "Груди";
    list.add(model);

    model = new ModelPopularWorkout();
    model.id = 6;
    model.image = "arms_workout.jpg";
    model.name = "Руки";
    list.add(model);

    return list;
  }
```