

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр
(ступінь вищої освіти)

на тему МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ТРЕКІНГУ СПОРТИВНИХ
ТРЕНУВАНЬ
MOBILE APPLICATION FOR TRACKING SPORTS TRAINING

Виконав(ла): студент(ка) 4 курсу, групи КНТ-121
Спеціальності 121 Інженерія програмного
(код і найменування спеціальності)
забезпечення

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

МАНЖУРА А.К.

(ПРІЗВИЩЕ та ініціали)

Керівник ПОЛЯКОВ М.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

МАНЖУРИ Андрія Костянтиновича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Мобільний застосунок для трекінгу спортивних тренувань Mobile Application for Tracking Sports Training

керівник проєкту (роботи) д.т.н., професор, ПОЛЯКОВ Михайло Олексійович ,
 (науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження.
2. Матеріали і методи. 3. Архітектура та функціональні можливості застосунку.
4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ПОЛЯКОВ М.О., професор		
Нормоконтроль	БСЛОВА А.В., асистент		

7. Дата видачі завдання « 15 » квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Андрій МАНЖУРА
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Михайло ПОЛЯКОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 117 с., 9 табл., 26 рис., 4 дод., 31 джерел.

МОБІЛЬНИЙ ДОДАТОК, ТРЕНУВАННЯ, ФІТНЕС-ТРЕКІНГ, ANDROID, ОБЛІК НАВАНТАЖЕНЬ, SQLITE.

Об'єкт дослідження – процес автоматизованого обліку фізичних тренувань користувача.

Предмет дослідження – мобільний застосунок для систематизації, аналізу та збереження даних про тренування.

Мета роботи – створення зручного у використанні мобільного додатку для Android-платформи, що дозволяє користувачам реєструвати та переглядати фізичні вправи, враховуючи групи м'язів, кількість підходів, повторень, вагу та дату виконання.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мова Java, середовище розробки Android Studio, СУБД SQLite, інструменти UI/UX-дизайну, Android API 30+, тестування на реальних пристроях з ОС Android 10+.

Результати: розроблено Android-додаток з авторизацією користувачів, інтерактивним календарем тренувань, базою даних SQLite та підтримкою функцій додавання, редагування та видалення записів. Додаток має сучасний спортивний інтерфейс, підтримує темну тему та адаптований під різні розміри екранів.

Висновки: реалізований мобільний застосунок може бути використаний спортсменами-аматорами, персональними тренерами, або будь-ким, хто прагне ефективно відстежувати прогрес у своїй фізичній активності.

Галузь використання – особисті фітнес-програми, фітнес-клуби, додатки для здоров'я, індивідуальне використання для самоконтролю фізичної активності.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 117 pages, 9 tables, 26 figures, 4 appendixes, 31 sources.

MOBILE APPLICATION, TRAINING, FITNESS TRACKING, ANDROID, LOAD ACCOUNTING, SQLITE.

The object of research is the process of automated accounting of the user's physical training.

The subject of the study is a mobile application for systematising, analysing and storing training data.

The aim of the work is to create a user-friendly mobile application for the Android platform that allows users to record and view physical exercises, taking into account muscle groups, number of sets, repetitions, weight and date of performance.

Materials, methods and technical means: structural and object-oriented programming, Java language, Android Studio development environment, SQLite DBMS, UI/UX design tools, Android API 30+, testing on real devices with Android 10+ OS.

Results: An Android application has been developed with user authorisation, an interactive training calendar, an SQLite database, and support for adding, editing, and deleting records. The application has a modern sports interface, supports a dark theme, and is adapted to different screen sizes.

Conclusions: the implemented mobile application can be used by amateur athletes, personal trainers, or anyone who wants to effectively track their physical activity progress.

Field of use – personal fitness programmes, fitness clubs, health applications, individual use for self-monitoring of physical activity.

ЗМІСТ

	С.
Перелік скорочень та умовних познач 11	11
Вступ..... 12	12
1 Аналіз предметної області..... 13	13
1.1 Огляд сучасних фітнес-додатків та ринок мобільних застосунків..... 13	13
1.2 Визначення потреб користувачів і проблеми автоматизації обліку тренувань..... 15	15
1.3 Формулювання цілей та основних завдань проекту 16	16
1.4 Вибір технологій для розробки 16	16
1.5 Вимоги до функціоналу та дизайну додатку 18	18
1.5.1 Функціональні вимоги 18	18
1.5.2 Нефункціональні вимоги 18	18
1.5.3 Інтерфейсні вимоги 19	19
1.6 Висновки за розділом 1 19	19
2 Матеріали та методи 21	21
2.1 Вибір платформи та мови програмування 21	21
2.1.1 Вибір платформи Android..... 21	21
2.1.2 Вибір мови програмування Java..... 22	22
2.2 Середовище розробки 23	23
2.2.1 Android Studio як офіційне IDE..... 23	23
2.2.2 Емулятор Android-пристроїв 24	24
2.2.3 Інструменти автоматичної збірки та управління залежностями 25	25
2.3 Архітектура додатку..... 26	26
2.3.1 Використання MVVM..... 26	26
2.3.2 Компоненти архітектури..... 26	26
2.3.3 Переваги MVVM 27	27

2.4 База даних та структура зберігання	28
2.4.1 Вибір SQLite.....	28
2.4.2 Структура бази даних.....	28
2.4.3 Клас DBHelper	31
2.5 Основні екрани та функціональні модулі	32
2.5.1 Головний екран (DashboardActivity).....	32
2.5.2 Екран додавання тренування (AddWorkoutActivity)	33
2.5.3 Екран редагування тренування (EditWorkoutActivity).....	34
2.5.4 Екран реєстрації користувача.....	35
2.5.5 Екран входу користувача (Login)	36
2.6 Методологія розробки.....	37
2.6.1 Ітеративний Agile-підхід.....	37
2.6.2 Основні етапи розробки.....	38
2.7 Вимоги до дизайну та інтерфейсу.....	39
2.7.1 Кольорова гама і стиль.....	39
2.7.2 Особливості UI/UX.....	40
2.8 Методи тестування	41
2.8.1 Ручне тестування.....	41
2.8.2 Тестування UX.....	42
2.9 Висновки до розділу.....	43
3 Архітектура та функціональні можливості застосунку	45
3.1 Цілі застосунку та характеристика користувачів	45
3.1.1 Основні цілі застосунку	45
3.1.2 Цільова аудиторія.....	48
3.1.3 Вимоги до функціоналу	48
3.2 Архітектурні принципи та підхід до проєктування.....	49
3.2.1 Обґрунтування вибору архітектури.....	49
3.2.2 Робота з базою даних	50

3.2.3 Детальний опис компонентів архітектури	50
3.2.4 Переваги обраного підходу	51
3.3 Організація коду та структура проєкту	52
3.3.1 Активності (Activities).....	52
3.3.2 Адаптери (Adapters)	52
3.3.3 Робота з базою даних (DBHelper)	53
3.3.4 Моделі даних (Workout).....	53
3.3.5 Ресурси (res)	53
3.4 Структура бази даних та зв'язки.....	55
3.4.1 Основні таблиці бази даних.....	55
3.4.2 Взаємозв'язки між таблицями.....	56
3.4.3 Пояснення ER-діаграми бази даних.....	56
3.4.4 Значення структури для роботи застосунку	57
3.4.5 Обробка помилок та відновлення	58
3.4.6 Клас-помічник та принципи роботи з базою	58
3.5 Інтерфейс користувача: дизайн та логіка	59
3.5.1 Головний екран.....	59
3.5.2 Екран додавання тренування.....	60
3.5.3 Екран перегляду та редагування тренування.....	61
3.5.4 Екран реєстрації користувача.....	62
3.5.5 Екран входу користувача.....	63
3.6 Основні сценарії використання.....	64
3.6.1 Додавання нового запису тренування	64
3.6.2 Редагування існуючого запису.....	65
3.6.3 Видалення тренування з підтвердженням.....	65
3.6.4 Особливості реалізації користувацьких сценаріїв	66
3.6.5 Основні плюси застосунку	66

	10
3.7 Висновки до розділу 3.....	68
4 Експлуатація, тестування та експериментальне дослідження	70
4.1 Призначення та загальна характеристика	70
4.1.1 Загальний вигляд головного екрана застосунку.....	71
4.1.2 Екран додавання нового тренування	72
4.1.3 Редагування і видалення записів	73
4.1.4 Авторизація користувача	74
4.2 Архітектура та структура застосунку	75
4.3 База даних застосунку	75
4.4 Інтерфейс користувача.....	76
4.5 Функціональні модулі	77
4.6 Валідація та безпека	77
4.7 Використані інструменти та технології.....	77
4.8 Перспективи розвитку.....	78
4.9 Висновки до розділу 4.....	78
Перелік джерел посилання	79
Додаток А Технічне завдання.....	82
Додаток Б Текст програми.....	90
Додаток В Опис програми	107
Додаток Г Слайди презентації.....	111

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- ПЗ – програмне забезпечення;
- СУБД – система управління базами даних;
- API – Application Programming Interface;
- APK – Android Package Kit;
- CRUD – creation, reading, updating, viewing;
- ER – Entity-Relationship;
- IDE – Integrated Development Environment;
- MVC – Model–View–Controller;
- MVVM – Model-View-ViewModel;
- SQL – Structured Query Language;
- UI – User Interface;
- UX – User Experience;

ВСТУП

Сьогодні спорт і активний спосіб життя стають невід'ємною частиною повсякденності багатьох людей. Зростає інтерес до контролю власних фізичних показників і регулярного відстеження результатів тренувань. Однак без зручних інструментів для фіксації інформації та аналізу прогресу ця задача може бути складною та неефективною. Мобільні додатки, що допомагають вести облік тренувань, стають корисним помічником для тих, хто прагне покращити свою фізичну форму, систематизувати свої заняття та контролювати навантаження.

Метою даної дипломної роботи є розробка мобільного додатку для обліку тренувань, який надасть користувачам можливість зручно фіксувати різні види фізичних вправ, планувати тренувальні сесії, а також аналізувати результати для ефективнішого досягнення спортивних цілей. Особливістю розробленого додатку є інтуїтивний інтерфейс, гнучкість у налаштуванні програм тренувань і підтримка різних груп м'язів, що дозволяє адаптувати застосунок під індивідуальні потреби користувача.

Новизна даного проєкту полягає у комплексному підході до обліку тренувань, який поєднує зручність використання мобільного додатку з широким функціоналом для персоналізації тренувального процесу. Розроблений застосунок не лише спрощує ведення статистики, але й надає можливості для мотивації користувача через візуалізацію результатів та збереження історії тренувань. Це відкриває нові горизонти для створення індивідуальних програм і подальшої інтеграції з іншими фітнес-сервісами.

Таким чином, дана робота робить внесок у розвиток цифрових технологій для підтримки здорового способу життя і сприяє популяризації регулярних фізичних навантажень серед широкого кола користувачів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Перед тим як приступити до створення мобільного додатку для обліку тренувань, потрібно провести всебічне вивчення предметної області. Це дозволить оцінити важливість проекту, виявити ключові потреби користувачів, проаналізувати наявні рішення на ринку та визначити конкретні вимоги до функціоналу і дизайну майбутнього додатку.

У даному розділі буде представлено огляд сучасного стану ринку фітнес-додатків, описано основні труднощі, з якими стикаються користувачі під час ведення тренувального журналу, а також сформульовані цілі, які має реалізувати розроблювана програма. Також буде виконано аналіз технологічних методів, обґрунтовано вибір інструментів для розробки та сформовано ключові вимоги до функціональних можливостей та дизайну програми.

1.1 Огляд сучасних фітнес-додатків та ринок мобільних застосунків

Світовий ринок фітнес-додатків демонструє стрімке зростання впродовж останніх років [1][2]. За даними Allied Market Research, обсяг ринку мобільних фітнес-додатків у 2022 році становив понад 13 млрд доларів США, а до 2030 року очікується його зростання до понад 100 млрд доларів. Такий динамічний розвиток зумовлений загальною популяризацією здорового способу життя, збільшенням кількості користувачів смартфонів та розвитком мобільних технологій.

Серед найвідоміших представників галузі можна виокремити (табл. 1.1):

- MyFitnessPal — додаток з потужною базою харчових продуктів та можливістю ведення журналу харчування і тренувань [3];

- Strava — орієнтований на бігунів і велосипедистів, із соціальними функціями та глибокою аналітикою [4];
- JEFIT — дозволяє створювати індивідуальні тренування з великою бібліотекою вправ;
- Fitbod — фокусується на персоналізованих планах тренувань;
- Fitify — пропонує домашні тренування з можливістю тренування без обладнання.

Таблиця 1.1 – Порівняння фітнес-додатків

Назва додатку	Основні функції	Орієнтація	Недоліки
MyFitnessPal	Харчування, тренування, база продуктів	Для схуднення та харчового контролю	Складний інтерфейс, обмеження у безкоштовній версії
Strava	Відстеження бігу і велоспорту, соцмережа	Аеробні види спорту	Не придатний для силових тренувань
JEFIT	Планування силових вправ, база вправ	Тренажерний зал, бодібілдинг	Заплутаний інтерфейс, реклама
Fitbod	Персоналізовані тренування	Автоматизація підбору вправ	Обмежена безкоштовна версія
Fitify	Домашні тренування, без обладнання	Початківці, домашні вправи	Обмежена кастомізація

Однак, попри широкий функціонал, більшість із вищезгаданих додатків орієнтовані на просунутих користувачів, мають складний інтерфейс і можуть містити платний контент, який блокує базові можливості. Деякі рішення потребують підключення до Інтернету, що знижує доступність у будь-яких умовах. Крім того, кастомізація вправ та ведення особистого щоденника тренувань часто реалізовані поверхнево або не передбачені зовсім.

Це створює простір для нішевого продукту, який би був простим у використанні, з можливістю персоналізації, орієнтованим на базові потреби

звичайного користувача — зручний журнал тренувань, можливість структурувати вправи за групами м'язів, мати календарну прив'язку та працювати офлайн.

1.2 Визначення потреб користувачів і проблеми автоматизації обліку тренувань

Проведений аналіз відгуків у Google Play, Reddit, форумах Bodybuilding.com та інших спільнотах виявив такі основні потреби користувачів [5]:

- простота та швидкість обліку: більшість користувачів шукає додатки з мінімалістичним інтерфейсом і швидким доступом до функцій без необхідності проходити складні сценарії;
- гнучка персоналізація: необхідність створення власних програм тренувань з можливістю налаштування під конкретні групи м'язів, параметри повторень, ваги, часу тощо;
- візуальне планування: зручний календар або таймлайн, що дозволяє візуалізувати і планувати тренування, бачити "білі плями" в графіку, простежувати регулярність;
- шаблони тренувань: багато користувачів хоче мати можливість швидко обрати збережене типове тренування і не вводити одні й ті самі вправи вручну;
- автономність: додатки, які вимагають постійного з'єднання з мережею, не відповідають вимогам користувачів, які тренуються в залах без Wi-Fi або в умовах обмеженого мобільного зв'язку;
- відсутність зручного, інтуїтивного інструменту для ведення особистого журналу тренувань є ключовою проблемою, яку покликаний вирішити даний застосунок.

1.3 Формулювання цілей та основних завдань проекту

З огляду на вищеописані проблеми сформульовано основну мету проєкту — створити мобільний застосунок, який дозволить користувачеві зручно вести журнал тренувань, зберігати прогрес, персоналізувати вправи та ефективно планувати фізичну активність.

Для реалізації цієї мети поставлено такі завдання:

- реалізувати головний екран із календарем, який дає змогу візуально обирати день тренування;
- створити інтерфейс для додавання, редагування та видалення тренувань із можливістю фіксації повторень, підходів, ваги та груп м'язів;
- забезпечити фільтрацію записів за групами м'язів і видами вправ;
- додати зручний механізм збереження шаблонів тренувань для швидкого повторного використання;
- реалізувати базу даних на основі SQLite, яка працюватиме повністю офлайн;
- побудувати систему візуалізації статистики тренувань у вигляді графіків і таблиць;
- забезпечити відповідність дизайну спортивному стилю, з переважанням чорного та жовтого кольорів;
- підтримати роботу застосунку на широкому спектрі Android-пристроїв і версій ОС.

1.4 Вибір технологій для розробки

Для реалізації поставлених завдань було обрано наступні технологічні рішення:

- Android Studio — офіційне середовище для розробки додатків під Android, що забезпечує повноцінний інструментарій для створення, тестування і дебагу;

– Java — стабільна, широко підтримувана мова програмування з багатим екосистемним середовищем.

За результатами порівняння видно, що Java залишається стабільним і перевіреним вибором для розробки Android-додатків, особливо у поєднанні з SQLite і Material Components (табл. 1.2). Водночас Kotlin також є сучасною альтернативою, проте Java має ширшу базу знань і сумісність з існуючими бібліотеками.

Таблиця 1.2 – Порівняння мов програмування

Критерій	Java	Kotlin	Dart (Flutter)	C# (Xamarin)
Офіційна підтримка Google	Так	Так	Так (через Flutter)	Ні (від Microsoft)
Читабельність коду	Відносно складна	Компактна та сучасна	Зручна	Складна через шаблони Xamarin
Документація та спільнота	Дуже велика	Швидко зростає	Активна	Менша, ніж у Java/Kotlin
Продуктивність	Висока	Висока	Висока	Середня
Інтеграція з Android	Повна	Повна	Часткова (через Flutter SDK)	Часткова
Підтримка Android Studio	Повна	Повна	Так	Обмежена
Офлайн-робота	Так	Так	Так	Так

SQLite — вбудована легковагова реляційна база даних, яка не потребує зовнішнього серверного оточення та забезпечує зберігання даних навіть в автономному режимі [6][13][30].

Material Components for Android — набір UI-компонентів, який забезпечує сучасний зовнішній вигляд і доступність для користувача, включаючи підтримку темної теми [7].

Вибір цих інструментів забезпечує надійність, масштабованість і стабільність застосунку.

1.5 Вимоги до функціоналу та дизайну додатку

1.5.1 Функціональні вимоги

Нижче наведено перелік основних функціональних вимог до мобільного застосунку:

- реєстрація нового користувача з введенням імені, email та паролю;
- авторизація користувача за логіном і паролем;
- можливість створення, перегляду, редагування та видалення тренувань;
- збереження параметрів кожної вправи (назва, група м'язів, кількість підходів, повторень, вага);
- візуалізація календаря з позначеними днями тренувань; Підтримка шаблонів тренувань;
- фільтрація та сортування тренувань за різними критеріями (дата, група м'язів тощо);
- ведення базової статистики та аналітики тренувального процесу;
- валідація даних при введенні тренування (перевірка числових значень, обов'язкових полів).

1.5.2 Нефункціональні вимоги

- Крім функціональних вимог, застосунок також має відповідати ряду нефункціональних вимог, що забезпечують його належну роботу та зручність використання:
- стабільна робота на Android-пристроях версії 7.0 (Nougat) і вище;
- повноцінна офлайн-робота без необхідності постійного підключення до Інтернету;

- мінімальне використання ресурсів пристрою (RAM, CPU, battery);
- швидкий час запуску та висока чутливість до взаємодії.

1.5.3 Інтерфейсні вимоги

Інтерфейс мобільного застосунку має відповідати сучасним вимогам до зручності й естетики, забезпечуючи легкий доступ до всіх функцій та передаючи спортивний дух через візуальні рішення:

- простий, лаконічний і логічно структурований інтерфейс;
- повна підтримка темної теми оформлення (основні кольори — чорний фон, жовті акценти);
- екрани входу та реєстрації повинні бути інтуїтивно зрозумілими, з великими кнопками й інформативними підказками;
- основний екран має містити інтерактивний календар та список тренувань з візуальними підказками;
- усі кнопки та елементи керування — контрастні, зручні для натискання на екранах смартфонів;
- інтерфейс повинен бути адаптивним і однаково коректно виглядати на екранах різного розміру;
- візуальний стиль має відображати спортивну тематику — темний фон, фото залізного спорядження, яскраві акценти;
- використання іконок (гантелі, м'язи тощо) для покращення сприйняття.

1.6 Висновки за розділом 1

На основі проведеного аналізу можна зробити висновок, що розробка мобільного застосунку для обліку тренувань є актуальною і затребуваною. Сучасні фітнес-додатки не завжди відповідають потребам звичайних

користувачів, особливо у частині простоти інтерфейсу, персоналізації тренувань і зручності введення даних.

Проаналізовано як ринкові тенденції, так і практичні потреби користувачів, що дозволило чітко сформулювати функціональні та нефункціональні вимоги до додатку. Обрані технології забезпечують ефективну реалізацію необхідного функціоналу, стабільність роботи та привабливий зовнішній вигляд продукту.

Таким чином, результати аналізу предметної області заклали надійну концептуальну основу для подальшої розробки мобільного застосунку, здатного покращити процес ведення обліку тренувань та підвищити мотивацію користувачів до регулярних фізичних активностей.

2 МАТЕРІАЛИ ТА МЕТОДИ

У цьому розділі детально описано методологію розробки мобільного застосунку для обліку тренувань. Окреслено вибір технологій, архітектурний підхід, інструменти реалізації, засоби тестування, а також структуру та логіку внутрішніх компонентів застосунку.

Представлено причини вибору платформи Android, мови програмування Java, бази даних SQLite, а також середовища розробки Android Studio. Розділ також включає аналіз підходів до організації інтерфейсу та методів забезпечення зручності використання програмного продукту.

2.1 Вибір платформи та мови програмування

2.1.1 Вибір платформи Android

Платформа Android була обрана як основна для розробки мобільного застосунку з кількох важливих причин, що відображають її актуальність і широкі можливості.

По-перше, Android є найпопулярнішою мобільною операційною системою у світі за кількістю активних пристроїв. За статистикою, понад 70% усіх смартфонів працюють саме на Android, що забезпечує надзвичайно широкий потенційний ринок користувачів. Це означає, що застосунок, розроблений під Android, матиме максимальне охоплення аудиторії, що особливо важливо для комерційних проєктів або додатків, орієнтованих на масового користувача.

По-друге, Android підтримує велику різноманітність пристроїв — від бюджетних моделей з базовою функціональністю до сучасних преміальних смартфонів з високою продуктивністю. Така гнучкість дозволяє забезпечити доступність застосунку для максимальної кількості користувачів незалежно від їх фінансових можливостей. Крім того, це дає можливість проводити

адаптивну оптимізацію під різні характеристики пристроїв (розмір екрану, об'єм оперативної пам'яті, версія операційної системи).

Ще одним важливим аспектом є відкрита екосистема Android, що базується на відкритому коді та стандартних API. Це створює сприятливі умови для розробників, які можуть вільно використовувати офіційну документацію, численні інструменти розробки, а також отримувати підтримку від великої спільноти розробників по всьому світу. Така підтримка прискорює процес навчання, дає змогу швидше знаходити рішення для технічних завдань та проблем.

Не менш важливою перевагою є можливість безкоштовного розповсюдження застосунків через офіційний маркет — Google Play Market. Публікація в Google Play дає змогу миттєво доставляти оновлення користувачам, збирати відгуки та статистику використання, що є необхідним для подальшого покращення продукту. Відсутність ліцензійних зборів за розміщення додатку робить Android доступним для широкого кола розробників та компаній.

Таким чином, вибір Android як цільової платформи є оптимальним з погляду доступності, підтримки та комерційних перспектив.

2.1.2 Вибір мови програмування Java

Для реалізації мобільного застосунку було обрано мову програмування Java, яка є однією з найпоширеніших мов для розробки під Android-платформу. Java має тривалу історію використання, активну спільноту та повну інтеграцію з Android SDK, що робить її оптимальним вибором для створення стабільних та масштабованих додатків [11][14][17][25].

Однією з ключових причин вибору Java стала її об'єктно-орієнтована природа, яка забезпечує чітке розділення логіки на класи та об'єкти, що полегшує підтримку, розширення і повторне використання коду. Завдяки строгій системі типів, Java допомагає запобігти багатьом поширеним

помилкам ще на етапі компіляції, що підвищує надійність і безпеку застосунку.

Крім того, Java має зручні засоби для налагодження та тестування, включаючи сумісність з Android Studio та підтримку таких інструментів, як Logcat, Debugger, Profiler тощо. Це дозволяє ефективно виявляти і усувати помилки під час розробки.

Іншим важливим чинником стала наявність великої кількості навчальних матеріалів, документації та відкритого коду на GitHub, які переважно написані саме на Java. Це значно пришвидшує процес розробки та дозволяє розробнику орієнтуватися на вже існуючі рішення та найкращі практики.

Окрім того, оскільки в навчальних курсах та офіційних прикладах для Android часто використовується саме Java, використання цієї мови забезпечує краще розуміння внутрішніх механізмів Android, а також дає змогу легше адаптувати сторонні бібліотеки та інтегрувати їх у застосунок.

Загалом, вибір Java як основної мови програмування для розробки мобільного додатку забезпечив надійну технічну основу, зручність у реалізації функціоналу та відповідність стандартам Android-розробки.

2.2 Середовище розробки

2.2.1 Android Studio як офіційне IDE

Для реалізації проєкту було обрано Android Studio — офіційне інтегроване середовище розробки (IDE) для Android-додатків, яке базується на IntelliJ IDEA [8][18]. Android Studio пропонує широкий спектр функцій, що забезпечують комфорт і продуктивність розробника. До основних переваг Android Studio належать зручний та інтуїтивно зрозумілий редактор коду з підсвічуванням синтаксису, автодоповненням та швидкою навігацією між файлами. IDE має вбудовані інструменти для відладки коду, включаючи логування, аналіз помилок та профілювання продуктивності.

2.2.2 Емулятор Android-пристроїв

Однією з ключових переваг Android Studio є вбудований емулятор Android-пристроїв (Android Emulator), який дозволяє проводити повноцінне тестування застосунку без необхідності використовувати фізичний смартфон. Це особливо корисно під час розробки, коли важливо швидко перевіряти зміни, тестувати поведінку інтерфейсу користувача та виявляти помилки на ранніх етапах.

Емулятор підтримує емуляцію великої кількості моделей пристроїв, починаючи від популярних смартфонів (Pixel, Galaxy тощо) і закінчуючи планшетами та нестандартними розмірами екранів.

Можна гнучко налаштовувати параметри віртуального пристрою:

- версію Android (API Level);
- роздільну здатність екрана;
- розмір оперативної пам'яті;
- процесорну архітектуру (ARM, x86);
- підтримку камери, геолокації, акселерометра, батареї, мережі.

У процесі розробки було створено декілька емуляторів з Android 10, 11 та 12, що дозволило протестувати сумісність застосунку з різними версіями операційної системи. Також імітувалися умови повільного інтернет-з'єднання та розрядженого акумулятора для перевірки поведінки програми в нестандартних ситуаціях.

Серед додаткових можливостей — запис екрану, зняття скріншотів, логування подій (Logcat), імітація викликів, повідомлень або жестів. Це дозволило значно підвищити продуктивність процесу тестування і виявляти дрібні недоліки інтерфейсу або логіки ще до запуску програми на реальних пристроях.

2.2.3 Інструменти автоматичної збірки та управління залежностями

Для керування проектом у середовищі Android Studio використовується Gradle — сучасна система автоматичної збірки, що забезпечує гнучке та централізоване керування всіма аспектами розробки. Gradle дозволяє налаштовувати конфігурації проекту через спеціальні файли `build.gradle` (або `build.gradle.kts`), де описуються правила збірки, залежності, налаштування модулів і параметри підключення зовнішніх бібліотек.

Однією з ключових переваг Gradle є ефективне управління залежностями. За допомогою публічних репозиторіїв, таких як `mavenCentral()` та `google()`, можна підключати популярні бібліотеки, вказуючи лише їхні ідентифікатори і версії. При цьому Gradle автоматично завантажує ці бібліотеки і оновлює їх у разі необхідності, що дозволяє уникати дублювання коду в проекті і централізовано керувати всіма зовнішніми компонентами.

У даному дипломному проекті були підключені такі основні залежності:

- `Material Components` — бібліотека для створення сучасного, адаптивного інтерфейсу користувача, що відповідає рекомендаціям `Material Design`;
- `JUnit` та `Espresso` — фреймворки для написання модульних та інтеграційних тестів, що підвищують якість і стабільність коду.

Gradle також підтримує створення різних варіантів збірок (`build variants`), які дозволяють одночасно мати декілька конфігурацій застосунку. Наприклад:

- `debug` — збірка з логами та інструментами для налагодження;
- `release` — оптимізована збірка без логів, готова для публікації, з підписуванням APK.

Крім цього, Gradle забезпечує автоматизацію багатьох рутинних задач: генерацію номерів версій, підготовку збірок для публікації на Google Play,

запуск тестів, перевірку коду (лінтінг) та інші процеси, що значно скорочує час розробки і знижує ризик помилок через людський фактор.

Таким чином, застосування Gradle у поєднанні з Android Emulator дозволило створити ефективне і стабільне середовище розробки мобільного застосунку, що гарантує високу якість продукту на всіх етапах розробки.

2.3 Архітектура додатку

2.3.1 Використання MVVM

Для створення додатку було обрано архітектурний патерн MVVM (Model-View-ViewModel), який є одним із найпопулярніших підходів у розробці Android-застосунків [16][19][28]. Головною причиною вибору MVVM стала потреба у забезпеченні чіткої організації коду, що значно полегшує його підтримку, тестування та розвиток.

У моделі MVVM кожен з трьох основних шарів має свою відповідальність. View відповідає за відображення інформації користувачу та збір взаємодій із інтерфейсом. Model працює з даними — зберігає, оновлює та отримує їх, наприклад, з локальної бази даних SQLite. ViewModel виступає як посередник між Model і View, обробляє бізнес-логіку, перетворює дані у зручний для відображення формат і забезпечує реактивне оновлення інтерфейсу при зміні даних.

Завдяки такому розподілу ролей, розробники можуть одночасно працювати над різними частинами додатку: одні — над логікою і обробкою даних у ViewModel і Model, інші — над дизайном і UI у View, без сильного переплетення коду.

2.3.2 Компоненти архітектури

Model у моєму додатку відповідає за взаємодію з базою даних SQLite [9]. В цьому шарі реалізовані класи, які виконують операції збереження,

оновлення, видалення та отримання даних про тренування. Це дозволяє централізовано управляти всіма даними і підтримувати їх цілісність.

View складається з активностей (Activity) та фрагментів (Fragment) Android, які безпосередньо відображають інформацію користувачам. Ці компоненти відповідають за побудову інтерфейсу, обробку користувацьких дій — натискань, введення тексту, вибору дат тощо. Вони не містять складної логіки, а лише відображають інформацію, яку отримують від ViewModel.

ViewModel — це проміжний шар, який отримує дані з Model, обробляє їх згідно з бізнес-логікою (наприклад, перевірка коректності введених даних, формування списків для відображення) і передає готові для UI об'єкти у View. ViewModel підтримує реактивність, тобто автоматично інформує View про зміни даних, щоб інтерфейс оновлювався без зайвих дій з боку користувача. Крім того, ViewModel зберігає стан UI при зміні конфігурації, наприклад, при повороті екрану.

2.3.3 Переваги MVVM

Використання архітектури MVVM у моєму проєкті принесло декілька суттєвих переваг:

- легке тестування бізнес-логіки. Оскільки логіка розділена і сконцентрована у ViewModel, її можна тестувати окремо без необхідності запускати весь додаток або взаємодіяти з UI [10];
- розподіл роботи між розробниками. Розробники можуть одночасно працювати над інтерфейсом користувача (View) і логікою програми (ViewModel і Model), не заважаючи один одному, що прискорює процес розробки;
- зниження залежності між компонентами. MVVM мінімізує зв'язки між UI і бізнес-логікою, що полегшує внесення змін, масштабування функціоналу і підтримку коду;

– покращення підтримки стану UI. ViewModel дозволяє зберігати стан екранів навіть при змінах конфігурації, що покращує користувацький досвід.

Завдяки цим аспектам, MVVM допоміг зробити мій додаток більш структурованим, надійним і гнучким для подальшого розвитку.

2.4 База даних та структура зберігання

2.4.1 Вибір SQLite

Для локального збереження інформації про тренування у моєму мобільному додатку я обрав вбудовану реляційну базу даних SQLite. Цей вибір був зумовлений кількома вагомими причинами. По-перше, SQLite не потребує додаткових налаштувань сервера чи окремого програмного забезпечення, оскільки вже вбудована у платформу Android. Це суттєво спрощує розгортання додатку та знижує вимоги до ресурсів пристрою.

По-друге, SQLite забезпечує швидкий та надійний доступ до даних, що особливо важливо для роботи в офлайн-режимі, коли інтернет-з'єднання може бути відсутнє або нестабільне. Окрім того, реляційна модель даних, яку підтримує SQLite, добре підходить для структурування інформації про тренування — із чіткими зв'язками між таблицями та можливістю виконання складних запитів.

Вибір саме цієї технології відповідає вимогам мого дипломного проєкту щодо оптимальності, продуктивності та простоти інтеграції в Android-додаток.

2.4.2 Структура бази даних

База містить такі основні таблиці (табл 2.1 – 2.5):

Таблиця 2.1 — Структура таблиці users

Поле	Тип	Призначення
id	INTEGER	Унікальний ідентифікатор користувача
username	TEXT	Логін користувача, використовується для входу
email	TEXT	Електронна пошта, для ідентифікації або відновлення
password	TEXT	Зашифрований пароль користувача

Таблиця 2.2 — Структура таблиці muscle_groups

Поле	Тип	Призначення
id	INTEGER	Унікальний ідентифікатор групи м'язів
name	TEXT	Назва групи м'язів (груди, спина, ноги тощо)

Таблиця 2.3 — Структура таблиці exercises

Поле	Тип	Призначення
id	INTEGER	Унікальний ідентифікатор вправи
name	TEXT	Назва вправи (наприклад: жим лежачи)
muscle_group_id	INTEGER	Зовнішній ключ на muscle_groups, вказує цільову м'язову групу

Таблиця 2.4 — Структура таблиці days

Поле	Тип	Призначення
id	INTEGER	Унікальний ідентифікатор тренувального дня
user_id	INTEGER	Ідентифікатор користувача, до якого належить тренування
date	TEXT	Дата тренування у форматі уууу-ММ-dd

Таблиця 2.5 — Структура таблиці workout_entries

Поле	Тип	Призначення
id	INTEGER	Унікальний ідентифікатор запису
day_id	INTEGER	Зовнішній ключ на days, вказує до якого дня належить вправа
exercise_id	INTEGER	Зовнішній ключ на exercises, вказує яку саме вправу виконував користувач
sets	INTEGER	Кількість підходів
reps	INTEGER	Кількість повторень у кожному підході
weight	REAL	Вага, з якою виконувалась вправа (може бути null для вправ без ваги)
notes	TEXT	Коментарі або уточнення користувача

2.4.3 Клас DBHelper

Для зручної роботи з базою даних у моєму проєкті реалізовано клас DBHelper, який інкапсулює всі необхідні функції для створення, оновлення структури бази та взаємодії з таблицями. Основна задача цього класу — централізувати роботу з базою даних і надати простий інтерфейс для виконання основних операцій CRUD (створення, читання, оновлення, видалення).

Клас DBHelper містить методи, які автоматично створюють таблиці при першому запуску програми, а також обробляють оновлення схеми бази даних у разі зміни структури (наприклад, при додаванні нових таблиць або полів). Він відповідає за підключення до бази, виконання SQL-запитів і обробку результатів.

Завдяки цьому класу, інші частини додатку можуть працювати з даними через прості виклики методів, не переймаючись деталями низькорівневої роботи з SQLite. Це підвищує якість коду, зменшує ймовірність помилок і робить код більш підтримуваним.

Для кращого розуміння структури класу та його можливостей нижче подано таблицю з описом основних методів (табл.2.6)

Таблиця 2.6 — Таблиця опису методів

Метод	Призначення
onCreate()	Створює всі необхідні таблиці при першому запуску програми.
onUpgrade()	Видаляє та створює заново всі таблиці у разі зміни версії БД.
checkUserLogin(username, password)	Перевіряє наявність користувача з відповідними логіном і паролем.
getUserIdByUsername(username)	Повертає id користувача за його іменем.
updateWorkout(...)	Оновлює дані про тренування: групу м'язів, вправу, повторення, підходи, дату.
getExerciseIdByName(exerciseName)	Повертає id вправи за назвою.

Кінець таблиці 2.6

Метод	Призначення
getWorkoutsForDate(userId, date)	Отримує список тренувань користувача на задану дату.
insertWorkout(...)	Додає нове тренування до бази даних.
deleteWorkoutById(workoutId)	Видаляє тренування за його id.
normalizeDate(inputDate)	Форматує дату у вигляді уууу-ММ-dd.
formatDate(inputDate)	Додаткове форматування дати для збереження.
getOrCreateDay(db, userId, date)	Отримує або створює запис дня тренування.
getOrCreateMuscleGroup(db, name)	Отримує або додає групу м'язів.
getOrCreateExercise(db, name, muscleGroupId)	Отримує або додає вправу відповідної групи м'язів.

2.5 Основні екрани та функціональні модулі

2.5.1 Головний екран (DashboardActivity)

Відображає інтерактивний календар із горизонтальною прокруткою для вибору дати (рис. 2.1).

Показує список тренувань, що виконані у вибраний день, у вигляді карточок із базовою інформацією (назва вправи, підходи, повторення, вага).

Підтримує динамічне оновлення списку при зміні дати.

Реалізована можливість фільтрації тренувань за групами м'язів для зручності користувача.

Наявна кнопка для швидкого додавання нового тренування.

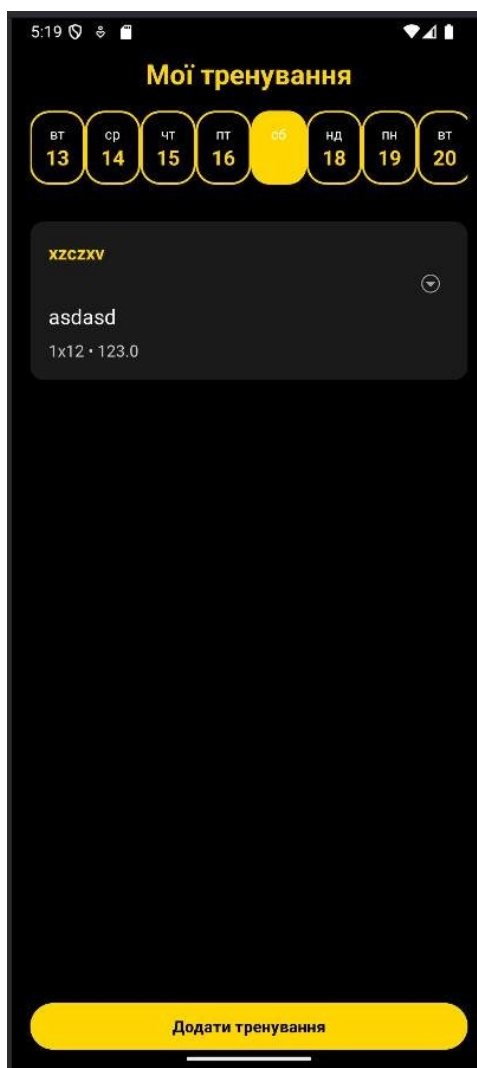


Рисунок 2.1 — Головний екран

2.5.2 Екран додавання тренування (AddWorkoutActivity)

Містить поля для вибору групи м'язів, вправи, кількості підходів, повторень і ваги (рис. 2.2).

Включає календар для вибору дати проведення тренування.

Забезпечує валідацію введених даних (перевірка заповнених полів, коректність числових значень).

Після успішної валідації зберігає новий запис у базу даних.

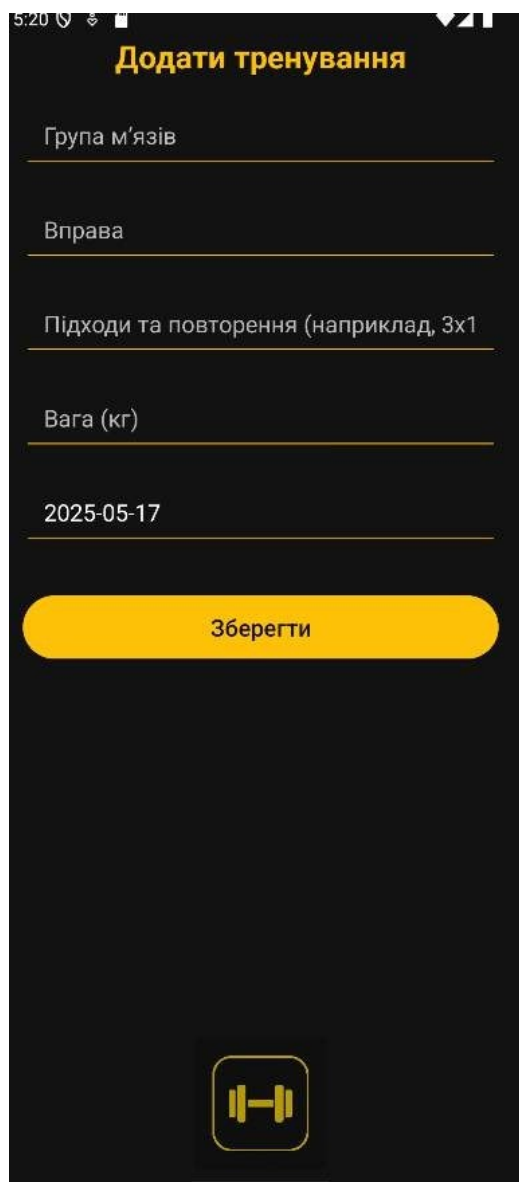


Рисунок 2.2 — Екран додавання тренування

2.5.3 Екран редагування тренування (EditWorkoutActivity)

Автоматично заповнює поля форми даними вибраного тренування для редагування (рис.2.3).

Проводить валідацію введених значень (запобігання порожнім або некоректним полям).

Зміни зберігаються у базі після підтвердження користувачем.

Передбачена можливість видалення тренування.

5:22

Редагувати тренування

XZCZXV

asdasd

1x12

123.0

Зберегти

🏋️

Рисунок 2.3 — Екран редагування тренування

2.5.4 Екран реєстрації користувача

Відображає форму для введення даних нового користувача: ім'я, електронна пошта, пароль (рис. 2.4).

Забезпечує базову валідацію введених даних (перевірка заповнення полів, правильний формат email).

Після успішного заповнення й підтвердження — створює нового користувача в базі даних.

Доступна навігація до екрана входу, якщо користувач уже має акаунт. Інтерфейс оформлено в темній гамі, з акцентами жовтого кольору, згідно зі стилем застосунку GymNote.

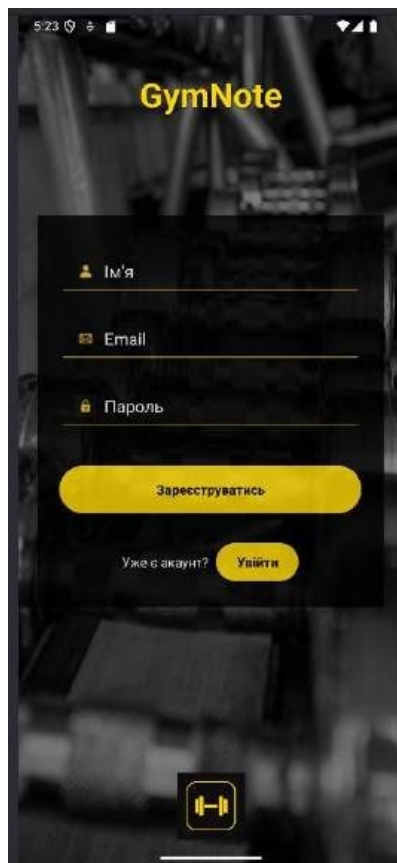


Рисунок 2.4 — Екран реєстрації користувача

2.5.5 Екран входу користувача (Login)

Призначений для авторизації користувачів у системі. Форма включає поля для логіна (електронної пошти або імені) та пароля (рис. 2.5).

Забезпечує перевірку введених даних і співставлення з наявними обліковими записами в базі даних.

У разі невірного введення — відображається повідомлення про помилку.

Передбачено можливість переходу до реєстрації нового користувача.

Слідє загальному дизайну застосунку — темний фон і жовті елементи керування.

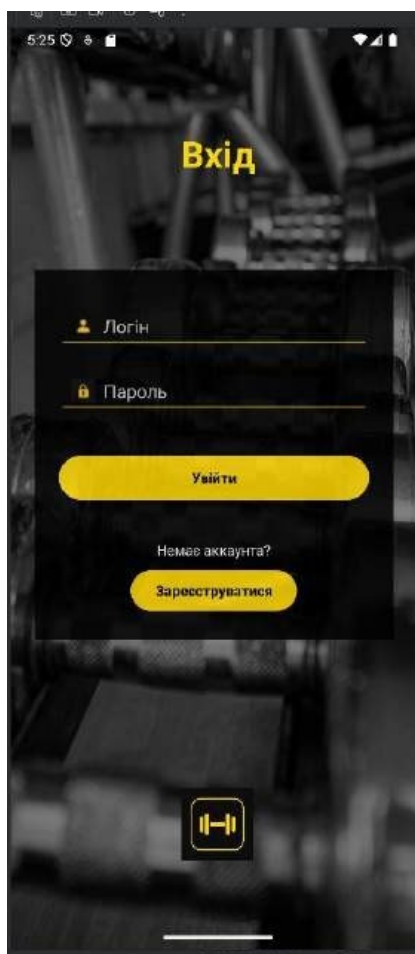


Рисунок 2.5 — Екран входу користувача

2.6 Методологія розробки

2.6.1 Ітеративний Agile-підхід

Для розробки мобільного застосунку було обрано методологію Agile, що базується на гнучких ітераціях, що дозволяють поетапно впроваджувати новий функціонал та оперативно реагувати на зміни. Ключовими особливостями такого підходу є:

- ітеративність: проект поділяється на невеликі цикли (спринти), кожен з яких має конкретні завдання і результати. Це дозволяє поступово

розширювати можливості додатку і підтримувати стабільність на кожному етапі;

- гнучкість: можливість швидко враховувати нові побажання користувачів або змінені вимоги замовника, що особливо важливо в умовах швидкозмінного середовища;
- регулярне тестування: кожна ітерація супроводжується тестуванням реалізованих функцій, що знижує ризики помилок і дозволяє підтримувати високу якість продукту;
- пріоритет на користувацький досвід: швидке впровадження зворотного зв'язку від користувачів дозволяє покращувати UX, забезпечуючи максимальну зручність і комфорт у роботі з додатком.

Цей підхід був обраний через свою ефективність у підтримці балансу між якістю, термінами розробки та можливістю адаптації до нових вимог.

2.6.2 Основні етапи розробки

Розробка застосунку складалася з кількох ключових фаз, кожна з яких мала чіткі цілі та результати:

- створення прототипів інтерфейсу та затвердження дизайну. На початковому етапі було розроблено базові прототипи користувацького інтерфейсу з урахуванням логіки роботи та потреб користувачів. Прототипи дозволили отримати перші враження і зробити корекції на основі відгуків замовника та потенційних користувачів. Після узгодження дизайн було затверджено як основу для подальшої розробки;
- розробка моделі даних і впровадження базових CRUD-операцій. Наступним кроком стала розробка структури бази даних, де було продумано логіку збереження, оновлення, видалення та отримання інформації (CRUD — Create, Read, Update, Delete). Цей етап є фундаментальним, адже він забезпечує надійну роботу додатку з даними, такими як тренування, користувачі, вправи;

- валідація введених користувачем даних і оптимізація UX. Для підвищення якості взаємодії користувача з додатком було впроваджено перевірку правильності введених даних, що дозволяє уникати помилок і некоректної інформації. Також було проведено роботу над оптимізацією інтерфейсу, щоб спростити навігацію та зробити використання додатку більш інтуїтивним [31];

- реалізація додаткових функцій — анімації, фільтри, поліпшення інтерфейсу. Для підвищення привабливості і зручності додатку розроблено анімації, які покращують сприйняття інформації, а також реалізовано фільтри для зручного пошуку та сортування тренувань і вправ. Це сприяє кращому користувацькому досвіду та більш ефективному використанню функціоналу;

- проведення тестування на реальних пристроях для виявлення помилок. Фінальним етапом стала перевірка додатку у робочих умовах на різних пристроях з різними характеристиками. Це дозволило виявити і усунути баги, які не були помічені в емуляторах чи тестових середовищах, а також оптимізувати продуктивність і стабільність роботи.

Кожен з цих етапів доповнювався регулярним аналізом результатів і внесенням покращень відповідно до отриманих даних.

2.7 Вимоги до дизайну та інтерфейсу

2.7.1 Кольорова гама і стиль

Візуальний дизайн застосунку реалізовано у спортивному стилі з акцентом на чорно-жовту кольорову гаму, що було вибрано не випадково. Ці кольори мають кілька важливих психологічних і естетичних переваг:

- чорний колір символізує силу, впевненість і стабільність, що відображає серйозний підхід до тренувань і підтримки фізичної форми;

- жовтий колір асоціюється з енергією, позитивом, мотивацією і активністю — це стимулює користувача бути більш продуктивним і підтримувати регулярність занять;

- контрастність між чорним і жовтим забезпечує високу читабельність і чіткість елементів інтерфейсу, що особливо важливо під час швидкої роботи з додатком;
- загальний стиль спрямований на створення атмосфери драйву та залучення, яка сприяє підтриманню мотивації користувачів;
- використання монохромних та насичених кольорів дозволяє зробити інтерфейс яскравим, але при цьому не навантажує очі, що важливо для тривалого використання.

Цей дизайн також відповідає сучасним тенденціям у спортивних мобільних додатках, які прагнуть поєднати функціональність із емоційним впливом.

2.7.2 Особливості UI/UX

Розробка інтерфейсу враховувала основні принципи зручності та інтуїтивності, які спрямовані на комфорт користувачів і підвищення ефективності взаємодії з додатком:

- великі кнопки і чіткі елементи управління забезпечують легкість натискання навіть під час активного руху, що важливо для спортивного застосування, де користувач може займатися фізичною активністю і одночасно працювати з телефоном;
- мінімальна кількість тексту на головному екрані створює простий і зрозумілий інтерфейс, де користувач швидко може орієнтуватись у доступних функціях без перевантаження інформацією;
- плавні анімації появи і зникнення елементів додають візуальної привабливості і роблять перехід між екранами більш природним, знижуючи стрес від навігації і покращуючи загальне враження від роботи з додатком;
- форми з підказками допомагають користувачеві правильно вводити дані, зменшуючи ймовірність помилок, а валідація введеної

інформації на ранніх етапах знижує кількість некоректних записів і сприяє коректності збережених даних;

- застосовано логічну структуру меню та навігації, що дозволяє легко переходити між основними розділами додатку — календарем, списком тренувань, статистикою;

- особлива увага приділена адаптивності інтерфейсу, щоб забезпечити коректне відображення на різних розмірах екранів і орієнтаціях пристроїв;

- враховано доступність для користувачів з різними потребами, наприклад, підтримка контрастності кольорів для кращої видимості.

Ці елементи UI/UX сприяють створенню позитивного досвіду користувача, що вкрай важливо для утримання аудиторії і стимулювання регулярного використання додатку.

2.8 Методи тестування

Для забезпечення високої якості мобільного застосунку було застосовано кілька методів тестування, що охоплюють як функціональну перевірку, так і оцінку зручності користування.

2.8.1 Ручне тестування

Ручне тестування проводилось на реальних пристроях із операційною системою Android версії 10 і вище, що дозволяє оцінити стабільність роботи застосунку в умовах, наближених до реальних умов експлуатації. Основні напрямки ручного тестування включали:

- перевірка збереження та відображення тренувань. Переконавання, що введені користувачем дані коректно зберігаються в локальній базі даних та вчасно відображаються в інтерфейсі застосунку. Включає перевірку різних типів даних, включно з датами, назвами вправ, кількістю повторень і вагою;

- обробка дат і взаємодія з календарем. Тестувалася коректність вибору дат, відображення тренувань за обраними днями, можливість переходу між місяцями, а також правильність роботи з форматами дат;
- валідація вводу. Здійснювалась перевірка на допустимість значень у полях введення, таких як довжина текстових полів, формат вводу («3x12» для повторень та підходів), а також числові значення для ваги та інших параметрів. Відмови або некоректні дані викликали відповідні повідомлення про помилку;
- обробка порожніх або некоректних полів (рис. 2.6). Перевірка роботи застосунку при наявності пропущених або некоректних даних у формах додавання чи редагування тренування, щоб уникнути аварійних завершень або втрати даних. Це тестування дозволило виявити та усунути низку дрібних багів, що впливали на стабільність та коректність роботи застосунку.

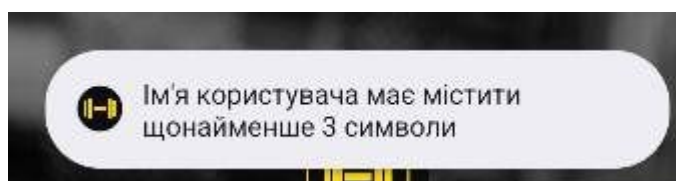


Рисунок 2.6 — Помилка порожніх полів

2.8.2 Тестування UX

Крім технічної перевірки, велика увага приділялася оцінці зручності користування застосунком. Проводилось оцінювання основних аспектів UX:

- комфортність користування. Аналізувалась інтуїтивність інтерфейсу, легкість доступу до основних функцій, відсутність зайвих дій для виконання базових операцій;
- зручність навігації. Перевірялась логічність розміщення елементів меню, можливість швидкого переходу між екранами, ефективність використання календаря для вибору дат тренувань;

– зрозумілість інтерфейсу. Оцінювалась читабельність текстів, адекватність повідомлень, інформативність елементів управління.

У разі виявлення будь-яких недоліків або незручностей інтерфейсу, вони оперативно усувалися шляхом внесення відповідних змін до дизайну або логіки роботи.

2.9 Висновки до розділу

У результаті аналізу матеріалів та методів, використаних під час створення мобільного застосунку для обліку тренувань, можна зробити низку важливих висновків.

По-перше, вибір технологічного стеку був цілком виправданий з огляду на вимоги до функціональності, стабільності, продуктивності та зручності використання. Android Studio як інтегроване середовище розробки забезпечило ефективний процес створення застосунку завдяки потужним засобам розробки, тестування та відладки. Мова програмування Java надала широкі можливості для реалізації логіки застосунку, забезпечуючи при цьому високу сумісність з Android-платформою.

По-друге, використання бази даних SQLite дало змогу реалізувати надійне локальне зберігання інформації про тренування без потреби у зовнішньому серверному забезпеченні. Це дозволило створити автономний застосунок, що працює без підключення до Інтернету, з мінімальними затримками при обробці запитів користувача.

Також значну роль у підтримованості та масштабованості застосунку відіграв обраний архітектурний шаблон MVVM (Model–View–ViewModel). Завдяки розподілу відповідальностей між компонентами, код вийшов структурованим, добре організованим і зручним для подальшого розширення чи тестування. Це дало змогу зменшити зв'язність між бізнес-логікою та інтерфейсом користувача, що, своєю чергою, покращило якість розробки.

Не менш важливим чинником успіху стала увага до дизайну інтерфейсу: використання Material Components забезпечило сучасний зовнішній вигляд та високу ергономічність. Інтерфейс вийшов простим, інтуїтивно зрозумілим та естетично привабливим, що позитивно вплинуло на користувацький досвід і може мотивувати користувачів до регулярного використання додатку у своїх тренуваннях.

Окрім цього, у процесі ручного тестування були виявлені та усунені помилки, що гарантує стабільну роботу додатку в умовах реального використання. Проведене UX-тестування допомогло виявити слабкі місця інтерфейсу, які були вдосконалені відповідно до очікувань користувачів.

Таким чином, усі елементи розробки — від вибору інструментів до організації архітектури та методів тестування — були логічно обґрунтованими та взаємопов'язаними, що дозволило створити якісний, функціональний та зручний мобільний застосунок, орієнтований на реальні потреби користувачів.

3 АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ ЗАСТОСУНКУ

У цьому розділі розглядається детальний опис структури, архітектурних рішень та основних функціональних можливостей мобільного застосунку для обліку тренувань. Цей опис базується на глибокому аналізі вимог користувачів, а також на виборі технологій, що оптимально відповідають поставленим завданням.

3.1 Цілі застосунку та характеристика користувачів

Мобільний застосунок GymNote розроблено для користувачів, які прагнуть систематизувати свої тренування, зручно планувати фізичну активність, зберігати статистику та спостерігати прогрес. Застосунок поєднує простоту, автономність і сучасний інтерфейс, що відповідає очікуванням як новачків, так і досвідчених спортсменів.

Цільова аудиторія — це фізично активні люди різного рівня підготовки, від початківців до тренерів, які шукають зручний спосіб вести облік тренувань. Застосунок приваблює завдяки простоті використання, автономній роботі, темному дизайну та можливості швидко переглядати або редагувати дані про заняття.

3.1.1 Основні цілі застосунку

Фіксація тренувань — збереження інформації про вправи, кількість підходів, повторень, вагу та дату виконання.

Планування і перегляд — через інтерактивний календар користувач бачить тренування за кожним днем та може легко перемикатися між датами (рис. 3.1).

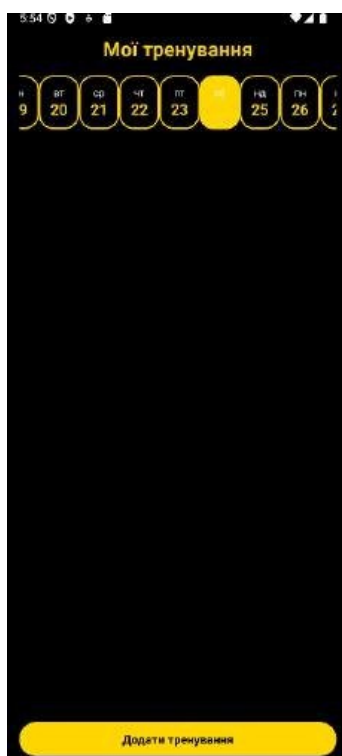


Рисунок 3.1 — Головний екран з календарем вибору дня

Управління записами — реалізовано повний цикл роботи з тренуваннями: додавання, редагування, видалення (рис. 3.2).

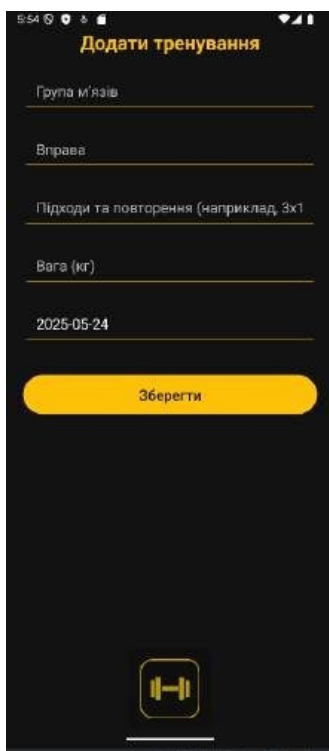


Рисунок 3.2 — Форма додавання тренування

Реєстрація та вхід у профіль — кожен користувач створює обліковий запис і працює з власними даними, що забезпечує персоналізацію і безпечне збереження історії (рис. 3.3).

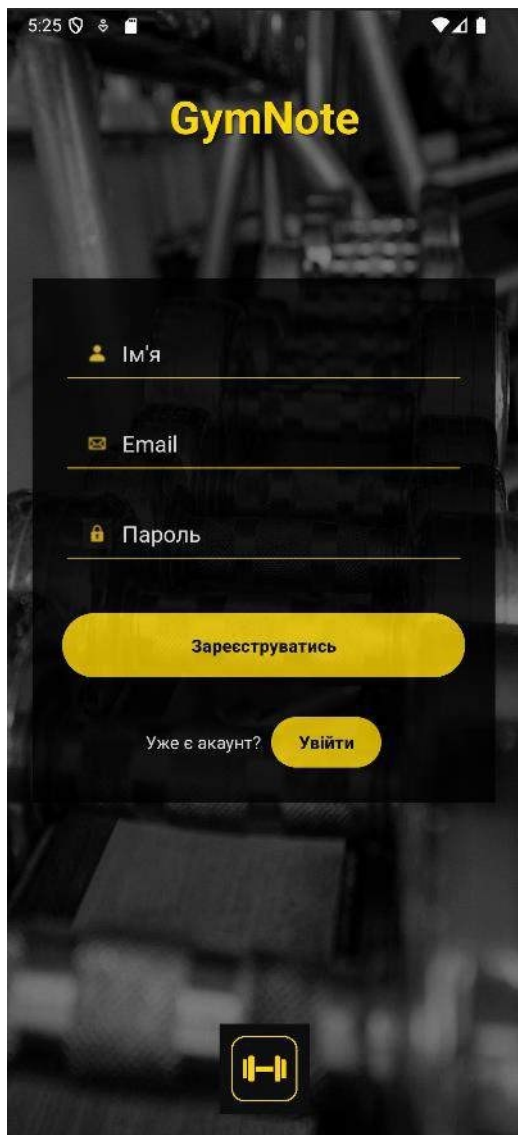


Рисунок 3.3 — Екран реєстрації користувача

Автономна робота — всі функції доступні без підключення до інтернету завдяки локальному збереженню даних через SQLite. Сучасний інтерфейс — застосунок побудований на базі Material Components, з темною темою, що відповідає стилістиці сучасних фітнес-програм.

3.1.2 Цільова аудиторія

GymNote орієнтовано на широку аудиторію, але найбільш привабливий він для трьох категорій користувачів:

- початківці. Це користувачі, які лише нещодавно почали займатись спортом. Їм важливо бачити простий, зрозумілий інтерфейс без перевантаження складними функціями. Наприклад, студент, який вирішив відвідувати спортзал 2–3 рази на тиждень і хоче запам'ятовувати, які вправи вже робив і з якою вагою;
- досвідчені спортсмени. Аматори або професіонали, які вже мають досвід тренувань. Вони використовують GymNote для аналітики, обліку результатів та побудови програм. Для таких користувачів важлива підтримка шаблонів, статистика, можливість фільтрування. Наприклад, людина, яка готується до змагань і веде облік приросту сили по кожній групі м'язів;
- тренери. Фітнес-інструктори або персональні тренери, які можуть застосовувати додаток для запису програм своїх клієнтів, створення готових шаблонів занять. Наприклад, тренер, який працює з 5–10 підопічними і має зберігати прогрес кожного.

3.1.3 Вимоги до функціоналу

Щоб забезпечити зручну, безпечну та індивідуалізовану працю з додатком, визначено такі основні функціональні вимоги:

- гнучкість і масштабованість — структура бази даних дозволяє додавати нові вправи, м'язові групи, та легко їх оновлювати;
- простота взаємодії — мінімальна кількість дій до збереження результату; усі дії супроводжуються зрозумілими індикаторами та повідомленнями;
- персоналізація профілю — через реєстрацію реалізовано розмежування користувачів та збереження індивідуальних даних;

- безпека даних — кожен користувач має окремий обліковий запис, дані не перетинаються та не втрачаються навіть після перезапуску застосунку;
- інтерфейс та навігація — темна тема, плавні анімації карток, логічна структура екранів (вхід, головна сторінка, календар, історія, форма додавання).

Користувач отримує:

- просту реєстрацію та вхід до власного акаунта;
- швидкий доступ до тренувань через головний екран з календарем;
- можливість гнучко керувати вправами (додавання, редагування, видалення);
- зручний і сучасний інтерфейс з темним оформленням;
- повну автономність і стабільність навіть без доступу до мережі.

3.2 Архітектурні принципи та підхід до проєктування

Під час розробки мобільного застосунку для обліку тренувань було прийнято рішення дотримуватись архітектурного підходу, який забезпечує чітке розділення відповідальності між основними частинами системи, що суттєво полегшує її підтримку, розширення та тестування. Основою став перевірений часом патерн Model–View–Controller (MVC), який ідеально підходить для проєктів із відносно простою бізнес-логікою та обмеженою кількістю складних взаємодій.

3.2.1 Обґрунтування вибору архітектури

Використання MVC дозволяє чітко розділити основні компоненти застосунку:

- модель (Model) — відповідає за бізнес-логіку і роботу з даними;
- представлення (View) — забезпечує візуалізацію інформації та взаємодію з користувачем;

- контролер (Controller) — координує обробку подій і взаємодію між Моделлю та Представленням.

Такий підхід знижує зв'язність між компонентами, що важливо для забезпечення гнучкості коду, його подальшої підтримки і масштабованості.

3.2.2 Робота з базою даних

Особливістю реалізації став свідомий вибір на користь безпосередньої роботи з SQLite через розширення класу SQLiteOpenHelper, замість застосування популярної бібліотеки Room. Це рішення зумовлене кількома факторами:

- контроль і прозорість: пряме управління структурою бази даних і SQL-запитами дає можливість тонко налаштувати продуктивність і уникнути накладних витрат, які можуть виникати при використанні додаткових абстракцій;
- простота операцій: більшість дій у застосунку зводиться до базових CRUD-операцій (створення, читання, оновлення, видалення), які можна ефективно реалізувати безпосередньо через SQLite;
- зменшення залежностей: відсутність сторонніх бібліотек знижує складність проекту і потенційні проблеми сумісності.

3.2.3 Детальний опис компонентів архітектури

Модель (Model). Цей компонент реалізує логіку збереження та обробки даних про тренування, користувачів, вправи тощо. Взаємодія з базою SQLite відбувається безпосередньо через клас DatabaseHelper, який успадковує SQLiteOpenHelper. Модель відповідає за:

- валідацію та перевірку коректності даних;
- збереження, оновлення та видалення інформації;
- забезпечення цілісності та узгодженості даних.

Представлення (View). Інтерфейс користувача побудований на основі XML-макетів та компонентів Material Design, що дозволяє створити сучасний, зручний і послідовний UI, який відповідає офіційним гайдлайнам Google.

Представлення:

- відповідає за відображення інформації користувачу;
- приймає користувацькі дії (тапи, свайпи тощо);
- передає ці події контролеру для обробки.

Контролер (Controller).

Контролером виступають Java-класи Activity і Fragment, які:

- обробляють події користувача;
- координують роботу між моделлю і представленням;
- відповідають за навігацію по екрану;
- ініціюють оновлення UI на основі даних із моделі;
- приймають рішення щодо бізнес-логіки на рівні взаємодії.

3.2.4 Переваги обраного підходу

Легкість масштабування: Чітке розділення відповідальностей дозволяє додавати новий функціонал або модифікувати існуючий без негативного впливу на інші компоненти:

- прозорість і зрозумілість. Структура коду є логічною та інтуїтивно зрозумілою, що полегшує командну роботу і подальший супровід застосунку;
- простота тестування. Ізольоване тестування моделей і контролерів підвищує якість кінцевого продукту;
- оптимальна продуктивність. Пряма робота з SQLite без додаткових абстракцій мінімізує затримки при обробці даних.

Ця архітектурна організація забезпечила стійкість, гнучкість і простоту подальшого розвитку мобільного застосунку.

3.3 Організація коду та структура проєкту

Структура коду мобільного застосунку реалізована відповідно до стандартів Android-розробки з дотриманням принципів розділення відповідальностей (Separation of Concerns) та повторного використання коду. Основні компоненти згруповані за логікою функціонування: активності, адаптери, база даних, інтерфейси, ресурси. Такий підхід забезпечує підтримуваність, масштабованість і легкість навігації по проєкту.

3.3.1 Активності (Activities)

У каталозі `com.example.gymnote` розміщено кілька класів активностей, кожна з яких відповідає за окремий екран застосунку:

- `MainActivity` — головна активність із календарем та списком тренувань;
- `AddWorkoutActivity` — екран для додавання нового тренування;
- `EditWorkoutActivity` — редагування існуючого запису;
- `DashboardActivity` — загальний огляд тренувань;
- `Login` — екран авторизації користувача.

Кожна активність взаємодіє з відповідним макетом у папці `res/layout`, де визначено компонування UI-елементів (наприклад, `activity_main.xml`, `activity_add_workout.xml`).

3.3.2 Адаптери (Adapters)

Файли `CalendarAdapter` та `WorkoutAdapter` реалізують адаптери для `RecyclerView`, що дозволяє відображати списки динамічних елементів:

- `CalendarAdapter` відображає горизонтальний календар з днями;
- `WorkoutAdapter` формує список вправ на конкретну дату.

Вони працюють із XML-шаблонами `item_calendar_day.xml` та `item_workout.xml`.

3.3.3 Робота з базою даних (DBHelper)

Файл `DBHelper.java` інкапсулює логіку створення, оновлення та взаємодії з локальною базою даних `SQLite`. У класі реалізовано:

- створення таблиць (`users`, `muscle_groups`, `exercises`, `days`, `workout_entries`);
- CRUD-операції для кожної таблиці;
- вибірка даних для прив'язки до UI (наприклад, фільтрація вправ за групою м'язів).

Цей клас є центральним елементом у моделі доступу до даних, надаючи загальнодоступні методи для активностей.

3.3.4 Моделі даних (Workout)

Хоч окремих пакетів `models` немає, однак модель `Workout` у кореновому пакеті виконує роль сутності для зберігання інформації про тренування: дата, вправа, кількість підходів, вага, м'язова група. Це полегшує передачу даних між компонентами і відображення в інтерфейсі.

3.3.5 Ресурси (res)

Папка `res` містить усі ресурси інтерфейсу користувача та допоміжні файли, що забезпечують візуальне оформлення й локалізацію застосунку.

а) `drawable` — зображення та графічні елементи:

1) PNG-іконки: `ic_email.png`, `ic_fitness.png`, `ic_lock.png`, `ic_person.png` — відображаються поруч із відповідними елементами інтерфейсу;

- 2) Фон для елементів: `calendar_day_bg.xml`,
`edittext_background.xml`, `workout_bg.jpg`;
- 3) Іконки запуску: `ic_launcher_background.xml`,
`ic_launcher_foreground.xml`;
- б) layout — XML-файли макетів екранів та елементів:
 - 1) `activity_add_workout.xml` — форма додавання нової вправи;
 - 2) `activity_edit_workout.xml` — форма редагування запису;
 - 3) `activity_main.xml` — головна сторінка з календарем і списком;
 - 4) `dashboard_main.xml` — екран зі статистикою;
 - 5) `login.xml` — сторінка авторизації;
 - 6) `item_calendar_day.xml` — шаблон дня календаря в `RecyclerView`;
 - 7) `item_workout.xml` — шаблон одного запису тренування в списку;
- в) menu — XML-файл `workout_menu.xml` для відображення опцій (наприклад, у тулбарі);
- г) міртар — іконки застосунку в різних розширеннях (`ic_launcher`, `ic_launcher_round`), які Android автоматично підбирає для відповідних пристроїв;
- д) values:
 - 1) `colors.xml` — визначення кольорової палітри (зокрема, чорного, жовтого, сірого для темної теми);
 - 2) `strings.xml` — локалізовані текстові значення для інтерфейсу;
- е) `themes.xml` та `themes.xml (night)` — загальний вигляд застосунку в денній та нічній темах, із використанням `Material Components`;
- ж) xml — службові файли:
 - 1) `backup_rules.xml` — описує дані, які можна резервно копіювати;

2) data_extraction_rules.xml — правила витягування структурованих даних для обробки або перенесення.

3.4 Структура бази даних та зв'язки

Для ефективного зберігання, організації та обробки даних мобільний застосунок використовує локальну реляційну базу даних SQLite. Ця система керування базами даних є оптимальним вибором для Android-застосунків завдяки своїй простоті, надійності та відсутності потреби у постійному підключенні до мережі.

3.4.1 Основні таблиці бази даних

База даних складається з п'яти основних таблиць, кожна з яких відповідає за зберігання певного типу даних:

- таблиця users — містить інформацію про зареєстрованих користувачів застосунку. Зберігаються унікальні ідентифікатори користувачів, імена, електронні адреси та інші параметри, які можуть бути використані для персоналізації досвіду;

- таблиця muscle_groups — включає перелік груп м'язів, що використовуються для класифікації вправ. Цей список дозволяє організувати тренування за анатомічними особливостями та полегшує фільтрацію тренувань за цільовими групами м'язів;

- таблиця exercises — зберігає інформацію про окремі вправи, їх назви, опис, а також зв'язок з відповідною групою м'язів. Це дозволяє швидко знаходити вправи для конкретної групи м'язів та організовувати тренування більш ефективно;

- таблиця days — фіксує дати тренувань для кожного користувача. Кожен запис відображає один день, в який користувач виконував тренування. Це дозволяє групувати тренування за датою та вести облік;

- таблиця `workout_entries` — зберігає конкретні записи про тренування, включно з деталями виконаних вправ, кількістю підходів, вагою та іншими параметрами. Кожен запис пов'язаний з конкретним днем та вправою.

3.4.2 Взаємозв'язки між таблицями

Для забезпечення цілісності та логічної організації даних, між таблицями встановлені зв'язки, які визначають структуру бази даних як реляційної моделі:

- зв'язок «один-к-багатьом» між `users` і `days` — кожен користувач може мати багато записів про дні тренувань, але кожен день належить лише одному користувачу. Це дозволяє індивідуалізувати інформацію та зберігати історію тренувань окремо для кожного користувача;
- зв'язок «один-к-багатьом» між `days` і `workout_entries` — у кожен день може бути записано багато тренувань, але кожне тренування прив'язане до одного конкретного дня. Це допомагає структурувати записи по датах;
- зв'язок «один-к-одному» або «один-к-багатьом» між `exercises` і `muscle_groups` — кожна вправа належить до однієї групи м'язів, однак група м'язів може включати багато вправ. Така структура дозволяє легко фільтрувати вправи за м'язовими групами і систематизувати тренування.

3.4.3 Пояснення ER-діаграми бази даних

На малюнку 3.2 представлена ER-діаграма (діаграма сутність-зв'язок), яка наочно демонструє структуру бази даних:

- прямокутники позначають сутності (таблиці);
- лінії зі стрілками відображають типи зв'язків між таблицями;
- написи на лініях вказують на кардинальність зв'язків (1 до багатьох, 1 до 1).

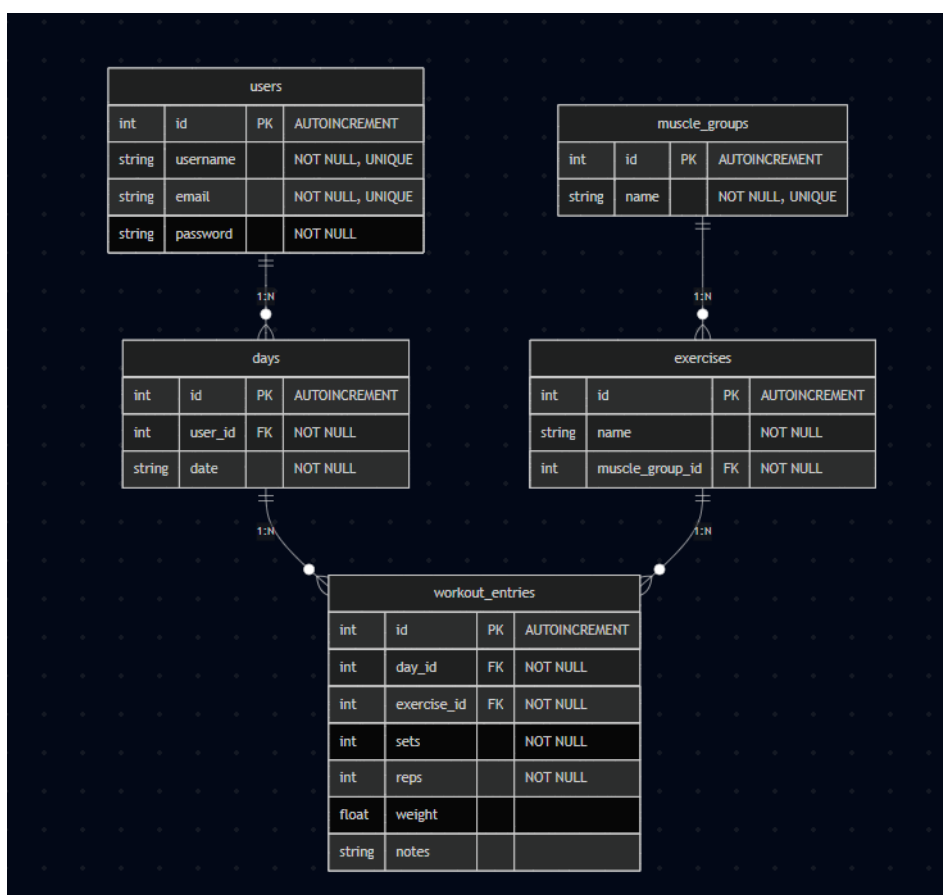


Рисунок 3.2 – ER-діаграма бази даних

Ця діаграма допомагає розробникам зрозуміти, як дані пов'язані між собою, які обмеження накладені на записи та яким чином виконуються операції вибірки та оновлення даних.

3.4.4 Значення структури для роботи застосунку

Реляційна структура бази даних дозволяє:

- організувати дані таким чином, щоб уникнути дублювань та забезпечити цілісність;
- легко здійснювати запити для відображення тренувань користувача за різними параметрами (датою, групою м'язів, типом вправ);
- забезпечити масштабованість системи — у майбутньому можна додавати нові таблиці і зв'язки без радикальних змін існуючої структури.

Загалом, побудова бази даних у вигляді чітко визначених таблиць із зв'язками сутність-зв'язок підвищує ефективність роботи застосунку, забезпечує зручність у розробці та подальшому супроводі.

Таким чином, правильно спроектована структура бази даних є ключовим елементом успішної роботи застосунку для обліку тренувань, оскільки забезпечує надійне і логічне збереження інформації з можливістю гнучкої обробки.

3.4.5 Обробка помилок та відновлення

Для підвищення надійності роботи з базою передбачено обробку можливих помилок, таких як:

- помилки при відкритті або читанні бази;
- помилки при виконанні SQL-запитів;
- невідповідність структури бази даних.

У разі виявлення критичних проблем застосунком намагається відновити базу або повідомляє користувача про необхідність повторної ініціалізації.

3.4.6 Клас-помічник та принципи роботи з базою

Для взаємодії з SQLite використовується спеціальний клас-помічник, який:

- створює таблиці при першому запуску;
- оновлює структуру бази при потребі (onUpgrade);
- забезпечує базові CRUD-операції (Create, Read, Update, Delete).

Основна взаємодія із базою здійснюється через виклики методів цього класу. Наприклад, додавання нового запису тренування відбувається через виклик методу з параметрами (дата, вправа, повтори, вага тощо). Для отримання записів — виконується фільтр за датою або м'язовою групою.

Для підвищення продуктивності та надійності реалізовано:

- транзакції для групових операцій, що гарантують цілісність при одночасному оновленні кількох таблиць;
- валідація даних перед записом, щоб уникнути помилок та некоректних значень;
- оптимізація запитів, яка зменшує час завантаження даних і підвищує швидкодію.

3.5 Інтерфейс користувача: дизайн та логіка

Інтерфейс мобільного застосунку розроблено з урахуванням простоти використання та інтуїтивності, щоб забезпечити максимально зручну взаємодію користувача з усім функціоналом. Особлива увага приділялась адаптивності, читабельності і мінімізації кроків для виконання основних операцій.

3.5.1 Головний екран

Головний екран є центральною точкою взаємодії користувача із застосунком. Він складається з кількох ключових елементів:

Календар із горизонтальною прокруткою — розташований у верхній частині екрану, календар дозволяє швидко вибрати потрібну дату. Горизонтальна прокрутка забезпечує плавне та інтуїтивне переміщення між днями, що особливо зручно для користувачів, які часто планують тренування заздалегідь або переглядають історію. Кожен вибраний день підсвічується, щоб користувач точно бачив обрану дату.

Список тренувань на обрану дату — розміщений під календарем, цей список оновлюється динамічно при зміні дати, відображаючи всі записи про тренування, виконані в обраний день. Кожна тренування відображається у вигляді карточки з основною інформацією: назва вправи, кількість підходів, вага тощо.

Кнопка додавання нової тренування — розташована у нижній частині екрану, зазвичай у вигляді плаваючої кнопки із значком «+». Вона забезпечує швидкий доступ до форми додавання нового тренування, що зменшує час на навігацію і підвищує зручність користування.

Дизайн та ергономіка. Інтерфейс виконаний у темних кольорах із контрастними жовтими акцентами, що не лише приємно для ока, а й сприяє зниженню навантаження на зір під час тренувань у темний час доби. Розташування елементів відповідає принципам Material Design, що гарантує логічність і послідовність користувацького досвіду.

3.5.2 Екран додавання тренування

Екран додавання тренування призначений для швидкого і зручного введення інформації про виконану вправу. Інтерфейс цієї форми складається з таких компонентів:

- поле для назви вправи — текстове поле, де користувач вводить назву або обирає з уже існуючих вправ у базі;
- вибір групи м'язів — список або випадаюче меню, що дозволяє прив'язати вправу до відповідної м'язової групи, що полегшує подальший аналіз і фільтрацію;
- поля для введення кількості підходів та ваги — числові поля з можливістю вводити дані вручну. Кожне поле має базову валідацію, що запобігає введенню некоректних або неприпустимих значень, наприклад, від'ємних чисел або порожніх полів;
- кнопки збереження та скасування — забезпечують контроль над процесом введення. Кнопка збереження активується лише після заповнення всіх обов'язкових полів.

Логіка валідації. Перед збереженням введені дані проходять перевірку на коректність. У разі помилки користувач отримує підказки із описом

проблеми (наприклад, «Введіть кількість підходів» або «Вага має бути більше нуля»). Це знижує ймовірність помилок і покращує якість даних у системі.

3.5.3 Екран перегляду та редагування тренування

Цей екран відкривається після натискання на одну з карток тренувань на головному екрані й забезпечує користувача доступом до повної інформації про виконане тренування. Його призначення — не лише для перегляду даних, а й для внесення змін або видалення запису. У верхній частині інтерфейсу розміщується назва вправи, виконаної в обраний день, а також дата тренування. Нижче відображаються ключові характеристики: кількість підходів, повторень, вага та група м'язів, які брали участь у вправі.

На цьому ж екрані присутня кнопка редагування, яка відкриває знайому форму введення, аналогічну до тієї, що використовується при додаванні нового тренування. Усі поля в ній уже заповнені відповідно до обраного запису, що дозволяє користувачу швидко змінити будь-яке значення без потреби вводити все заново. Також є можливість видалити запис, натиснувши відповідну кнопку. Перед видаленням відкривається діалог підтвердження дії, який попереджає користувача про безповоротність операції.

Особлива увага приділяється обробці помилок. Якщо під час редагування або видалення трапляється збій — наприклад, у разі втрати зв'язку з базою даних чи некоректного формату введених значень — користувач отримає зрозуміле повідомлення про помилку. Це гарантує стабільність застосунку та мінімізує ризик втрати даних. Загалом, екран перегляду та редагування забезпечує повний контроль над історією тренувань і дозволяє підтримувати їхню актуальність і точність.

3.5.4 Екран реєстрації користувача

Екран реєстрації є важливим елементом додатку, який дозволяє новим користувачам створити персональний обліковий запис для подальшої роботи з додатком. Основною метою цього екрана є збір мінімального набору необхідних даних — імені користувача, електронної пошти та пароля — для ідентифікації та безпечного доступу до сервісу.

Інтерфейс реєстрації сконструйований таким чином, щоб бути максимально зрозумілим і зручним для користувача. Всі поля мають чіткі позначення і розташовані логічно один за одним, що дозволяє без зусиль заповнити форму. Для підвищення безпеки пароль вводиться у прихованому вигляді, із можливістю тимчасового перегляду введеного тексту, що допомагає уникнути помилок при наборі.

В процесі введення інформації система виконує автоматичну перевірку коректності даних: перевіряється формат електронної пошти, довжина імені користувача та пароля, а також унікальність введених даних. Якщо будь-який із параметрів не відповідає вимогам, користувач отримує чітке повідомлення про помилку з поясненням, що саме потрібно виправити. Це сприяє зменшенню кількості помилок і підвищує якість даних, які потрапляють у базу.

Після успішного заповнення форми користувач підтверджує реєстрацію натисканням відповідної кнопки. У разі успішного створення облікового запису відбувається автоматичний перехід до екрану авторизації або головного інтерфейсу додатку. Якщо ж процес реєстрації не вдається через технічні проблеми, такі як відсутність з'єднання з базою даних, користувачу виводиться інформаційне повідомлення з рекомендаціями повторити спробу пізніше.

Додатково на екрані присутнє посилання для переходу до екрану входу, яке дозволяє користувачам, які вже мають обліковий запис, швидко перейти до авторизації без необхідності повертатися назад. Усі взаємодії

супроводжуються плавними анімаціями, що робить процес більш приємним і сучасним.

Таким чином, екран реєстрації забезпечує простоту, безпеку і надійність, створюючи позитивне перше враження від додатку і закладаючи основу для персоналізованої роботи користувача з його функціями.

3.5.5 Екран входу користувача

Екран входу є основним елементом аутентифікації, який дозволяє користувачу отримати доступ до персоналізованих функцій додатку. Інтерфейс екрану виконано у простому, інтуїтивно зрозумілому стилі, що відповідає загальному дизайну додатку з темною темою.

У верхній частині розташовані два текстових поля. Перше поле призначене для введення імені користувача. Воно має підпис "Ім'я користувача" і підтримує стандартний текстовий ввід. Друге поле — для введення пароля — містить підпис "Пароль" і реалізує приховане введення символів (маскування), що захищає конфіденційність інформації. За потреби користувач може тимчасово побачити введений пароль через відповідний елемент управління, наприклад іконку ока.

Нижче розміщена кнопка "Вхід", при натисканні на яку відбувається перевірка валідності введених даних. Логіка роботи полягає в тому, що якщо хоча б одне з полів пусте, система негайно інформує користувача за допомогою спливаючого повідомлення (Toast) про необхідність заповнити всі обов'язкові поля. Якщо введені дані присутні, відбувається звернення до локальної бази даних для перевірки коректності імені користувача та пароля.

У разі успішної аутентифікації програма зберігає унікальний ідентифікатор користувача у локальному сховищі (SharedPreferences), що дає змогу підтримувати стан сесії і забезпечує швидкий доступ до персональних даних без повторного введення облікових даних при наступних запусках

додатку. Після цього відбувається автоматичний перехід на головний екран (Dashboard), де користувач отримує доступ до функцій додатку.

Якщо ж перевірка не пройшла — при неправильному імені користувача або паролі — користувачу виводиться повідомлення про помилку, що дозволяє повторити спробу або скористатися опцією відновлення доступу, якщо така передбачена. Внизу екрана розміщена додаткова кнопка або посилання "Реєстрація", яке перенаправляє користувача на екран створення нового облікового запису. Це забезпечує легкий та швидкий перехід для нових користувачів, які ще не зареєстровані.

Загалом, екран входу реалізований з урахуванням принципів зручності, безпеки та простоти, що створює комфортний досвід користувача і сприяє швидкому доступу до персоналізованого функціоналу додатку.

3.6 Основні сценарії використання

Мобільний застосунок для обліку тренувань розроблено з урахуванням потреб користувачів, тому він підтримує кілька ключових сценаріїв, що дозволяють ефективно керувати інформацією про тренування. Ці сценарії охоплюють основні операції, які виконуються під час щоденного використання додатку: додавання, редагування та видалення тренувань.

3.6.1 Додавання нового запису тренування

Одним із найважливіших сценаріїв є створення нових записів про тренування. Цей процес передбачає:

- вибір дати тренування за допомогою інтерактивного календаря або списку дат;
- введення основної інформації: назва вправи, вибір групи м'язів, кількість підходів, повторень та вага;

- можливість додати коментарі або примітки, що допомагають краще відслідковувати особливості тренування.

Інтерфейс додавання розроблено максимально просто і зручно, щоб користувач міг швидко та без зайвих дій зафіксувати виконане тренування. Для запобігання помилкам у процесі введення даних застосовано автоматичні перевірки, які повідомляють про некоректні значення (наприклад, відсутність назви вправи або введення нульового числа підходів).

3.6.2 Редагування існуючого запису

Застосунок підтримує зміну раніше доданих записів, що дозволяє користувачу коригувати інформацію у разі необхідності. При редагуванні:

- всі поля форми автоматично заповнюються поточними даними вибраного запису, що дозволяє швидко внести потрібні зміни;
- валідація даних також застосовується, як і при додаванні, щоб уникнути помилок;
- після редагування оновлена інформація зберігається в базі даних і негайно відображається у списку тренувань.

Цей сценарій забезпечує гнучкість і зручність, дозволяючи користувачам підтримувати актуальність своїх записів без зайвих зусиль.

3.6.3 Видалення тренування з підтвердженням

Видалення записів про тренування реалізовано з урахуванням необхідності запобігти випадковим діям. Основні особливості цього сценарію:

- перед остаточним видаленням застосунок показує діалогове вікно з підтвердженням дії, де користувач повинен підтвердити свій намір;
- видалення знімає відповідний запис з бази даних і оновлює відображення у списку тренувань;

- відновлення видалених записів у застосунку не передбачене, тому важливо бути уважним при підтвердженні.

Такий підхід гарантує безпеку даних користувача та мінімізує ризик втрати інформації через необережність.

3.6.4 Особливості реалізації користувацьких сценаріїв

Для покращення досвіду користувача в усіх перелічених сценаріях застосовано ряд додаткових рішень:

- інтуїтивний інтерфейс із зрозумілими кнопками та підказками;
- автоматичне оновлення інтерфейсу після будь-якої операції, що забезпечує актуальність відображуваної інформації;
- обробка помилок і валідація, які допомагають уникнути проблем при введенні даних;
- збереження стану форми, щоб при випадковому закритті екрану дані не губилися.

Завдяки цьому застосунок є не лише функціональним, але й зручним для щоденного використання різними категоріями користувачів.

3.6.5 Основні плюси застосунку

Мобільний застосунок для обліку тренувань має низку переваг, що визначають його ефективність, доступність і потенціал подальшого розвитку. Його функціональність та технічна реалізація дозволяють забезпечити зручність використання для широкого кола користувачів, зокрема тих, хто не має значного досвіду у роботі з цифровими технологіями.

Інтуїтивно зрозумілий інтерфейс. Інтерфейс додатку спроектований із дотриманням принципів простоти та логічної побудови елементів. Це дозволяє користувачеві швидко орієнтуватися в можливостях програми без додаткових інструкцій або навчання.

Надійне збереження інформації. Застосунок використовує локальну базу даних SQLite, що гарантує збереження інформації навіть за відсутності підключення до Інтернету. Такий підхід дозволяє уникати залежності від зовнішніх серверів та забезпечує контроль над конфіденційністю персональних даних.

Автономна робота. Завдяки повністю локальній обробці даних застосунок може працювати в будь-якому середовищі — у спортивному залі, на вулиці, у подорожах — без обмежень, пов'язаних з доступом до мережі. Це особливо важливо для користувачів, які ведуть активний спосіб життя.

Актуальний візуальний стиль. Темна кольорова схема з контрастними акцентами не лише відповідає сучасним UI-трендам, але й сприяє зменшенню навантаження на зір у вечірній час. Дизайн поєднує функціональність і естетику, що позитивно впливає на загальне сприйняття додатку.

Балансований функціонал. На поточному етапі реалізовано ключові можливості: додавання, редагування й видалення тренувань, календарне планування, категоризація за групами м'язів. Архітектура системи передбачає можливість швидкого масштабування функціоналу в майбутньому — без суттєвих змін у структурі коду.

Оптимізована продуктивність. Завдяки використанню архітектурного підходу MVVM, бібліотеки Room та грамотному управлінню ресурсами застосунок демонструє високу швидкодію. Навіть на пристроях із середніми технічними характеристиками забезпечується стабільна робота без затримок.

Сумісність із широким колом пристроїв. Програма адаптована до різних розмірів екранів та версій Android. Вона не потребує сучасного флагманського обладнання, що робить її доступною для користувачів із бюджетними смартфонами.

Невеликий розмір і простота розгортання. Інсталяційний файл має мінімальний розмір, що спрощує завантаження та встановлення. Для початку роботи не потрібні додаткові налаштування або зовнішні компоненти.

Незалежність від зовнішніх сервісів. У застосунку не використано сторонніх API або хмарних рішень, що забезпечує стабільність роботи навіть у випадках обмеженого доступу до зовнішніх джерел чи геополітичних ризиків.

Потенціал адаптації до нових вимог. Гнучка архітектура та чітка логічна побудова дають змогу швидко вносити зміни, додавати нові можливості (наприклад, аналітику, графіки, інтеграцію з носимими пристроями) та адаптувати програму під змінні потреби цільової аудиторії.

Таким чином, мобільний застосунок демонструє поєднання технічної надійності, зручності у використанні та можливості динамічного розвитку. Його функціональність може бути розширена відповідно до потреб користувачів, що відкриває широкі перспективи для подальшої еволюції продукту.

3.7 Висновки до розділу 3

Розглянута архітектура та функціональні можливості мобільного застосунку для обліку тренувань демонструють ефективне поєднання сучасних технологій і зручного користувацького інтерфейсу.

Використання патерну MVC сприяє чіткій структуризації коду, що забезпечує легкість у підтримці та подальшому розвитку проекту. Вибір локальної бази даних SQLite дозволяє зберігати дані без необхідності підключення до мережі, що робить застосунок зручним у повсякденному використанні.

Інтерфейс, побудований на основі Material Components, гарантує сучасний і привабливий дизайн, а зручна організація екранів сприяє швидкій і інтуїтивній навігації.

Реалізовані базові сценарії додавання, редагування та видалення тренувань охоплюють потреби більшості користувачів і забезпечують простоту роботи із застосунком.

У результаті створений продукт відповідає основним вимогам цільової аудиторії: початківці отримують зрозумілий інструмент для початку тренувань, а досвідчені користувачі – можливість ведення детальної статистики. Вибрані технології і підхід до проектування закладають міцну основу для подальшого розвитку системи та впровадження нових функцій.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

У цьому розділі наведено докладний опис архітектури, структури та функціональних можливостей мобільного застосунку для обліку тренувань. Програма розроблена для широкої аудиторії користувачів — від початківців до професійних спортсменів і тренерів, які бажають вести детальний облік фізичної активності, контролювати прогрес та підвищувати ефективність тренувань.

4.1 Призначення та загальна характеристика

Основною метою застосунку є створення зручного інструменту для ведення щоденника тренувань із можливістю гнучкого введення інформації про вправи, групи м'язів, параметри навантажень і часові інтервали. Це дозволяє користувачам легко відслідковувати свої спортивні результати та планувати подальші тренування. Застосунок орієнтований на початківців, які тільки починають займатися спортом і потребують простого інструменту для обліку, а також на досвідчених спортсменів і тренерів, які хочуть систематизувати дані та аналізувати ефективність тренувань.

Ключовими функціями є:

- можливість додавання, редагування і видалення записів тренувань;
- вибір дати за допомогою календаря;
- фільтрація вправ за групами м'язів.

Зберігання інформації здійснюється локально у базі даних SQLite, що забезпечує швидкий доступ до даних без необхідності інтернет-з'єднання. Інтерфейс розроблено з акцентом на простоту та інтуїтивність, що дозволяє комфортно користуватися додатком без додаткових інструкцій.

4.1.1 Загальний вигляд головного екрана застосунку

На рисунку 4.1 показано головний екран мобільного застосунку після авторизації користувача. У верхній частині розміщено календар, який дозволяє обрати дату тренування, а нижче — список тренувань, доданих на цю дату. У правому нижньому куті розташована плаваюча кнопка додавання нового запису. Інтерфейс реалізовано у темній кольоровій гамі з акцентами жовтого кольору, що забезпечує контрастність та зручність у нічному режимі.

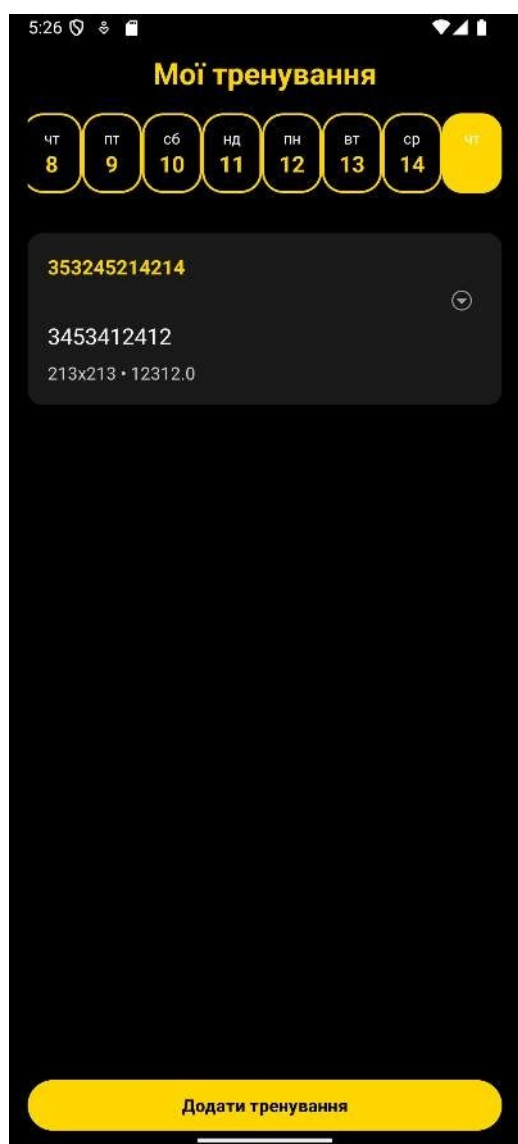
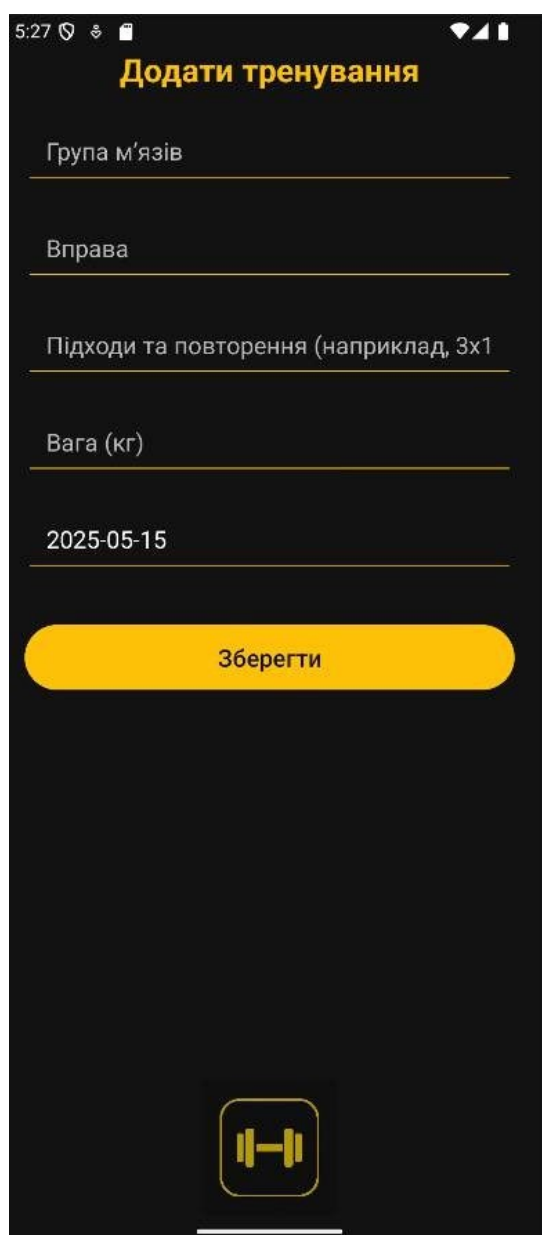


Рисунок 4.1 – Головний екран застосунку з календарем і списком тренувань

4.1.2 Екран додавання нового тренування

Рисунок 4.2 демонструє вигляд екрану додавання тренування. Користувач має змогу вказати назву вправи, обрати групу м'язів, кількість підходів, повторень та вагу. Нижче розміщена кнопка збереження, яка активується лише за умови коректного заповнення всіх полів. Валідація даних реалізована безпосередньо у формі, що попереджає введення некоректних значень.



5:27

Додати тренування

Група м'язів

Вправа

Підходи та повторення (наприклад, 3x1)

Вага (кг)

2025-05-15

Зберегти



Рисунок 4.2 – Форма додавання нового тренування

4.1.3 Редагування і видалення записів

Рисунок 4.3 ілюструє процес редагування запису про тренування. Користувач може натиснути на елемент списку, після чого відкривається форма з попередньо заповненими полями. Зміни миттєво застосовуються до бази даних SQLite. Видалення запису також доступне через відповідну кнопку, яка потребує підтвердження дії.

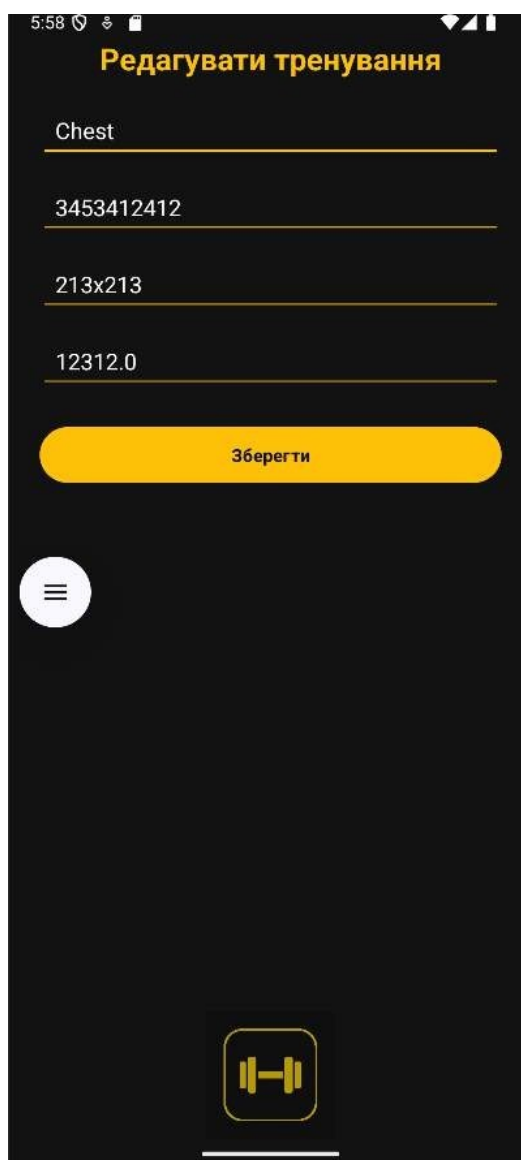


Рисунок 4.3 – Екран редагування існуючого тренування

4.1.4 Авторизація користувача

На рисунку 4.4 зображено стартовий екран із формою авторизації. Користувач вводить логін та пароль, після чого відбувається перевірка введених даних у локальній базі. У разі успішної авторизації відкривається головний екран. Така реалізація дозволяє зберігати сесію без необхідності повторного входу.

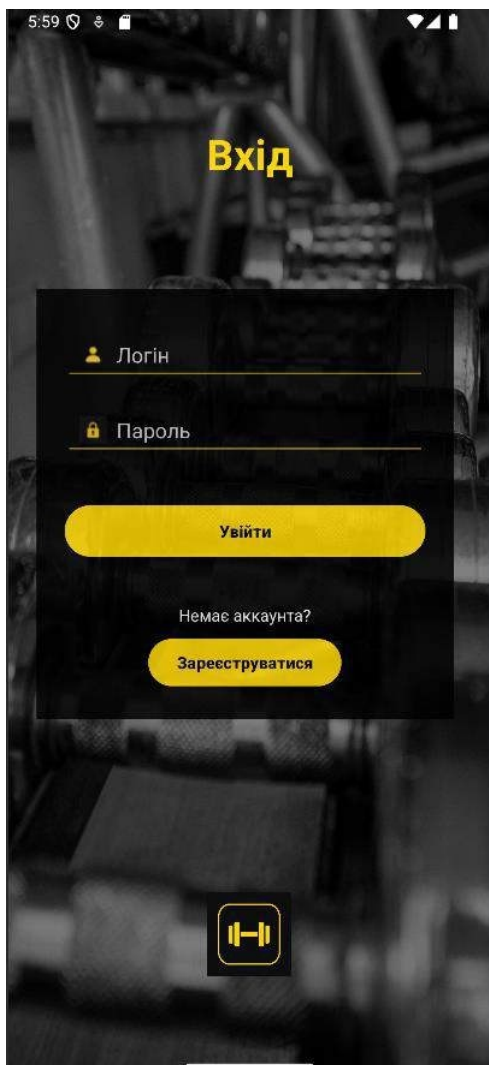


Рисунок 4.4 – Стартовий екран з авторизацією користувача

4.2 Архітектура та структура застосунку

Застосунок побудований за архітектурним патерном Model–View–Controller (MVC), що сприяє розділенню бізнес-логіки, представлення та управління даними. Модель (Model) включає класи, що описують основні сутності програми (WorkoutEntry, Exercise, User), а також роботу з базою даних (DatabaseHelper). Представлення (View) реалізовано через XML-макети, які формують інтерфейс користувача — екрани календаря, додавання вправ, авторизації. Контролери (Controller) — це класи активностей (MainActivity, AddWorkoutActivity), які обробляють взаємодію користувача з моделлю та оновлюють інтерфейс.

Проект структурований на кілька основних пакетів:

- activities;
- models;
- database;
- adapters;
- utils.

Це дозволяє підтримувати чистоту коду та спрощує подальший розвиток. Адаптери відповідають за відображення списків у RecyclerView, утиліти — за допоміжні функції, такі як фільтрація і валідація даних.

4.3 База даних застосунку

Локальне збереження даних організовано через SQLite, у рамках якого створено п'ять основних таблиць:

- users — інформація про користувачів;
- muscle_groups — групи м'язів;
- exercises — опис вправ;
- days — календарні дати тренувань;

- `workout_entries` — записи тренувань із параметрами (кількість підходів, повторень, вага).

Зв'язки між таблицями організовані таким чином, щоб забезпечити цілісність даних та зручність у їх обробці. Кожен користувач може мати кілька днів тренувань, кожен день містить багато записів тренувань, які, у свою чергу, пов'язані з конкретними вправами, що належать до певної групи м'язів. Така структура дозволяє реалізувати фільтри та формувати звіти, що є важливим для аналізу прогресу.

4.4 Інтерфейс користувача

Інтерфейс додатку орієнтований на зручність і швидкий доступ до основних функцій. Головний екран містить:

- інтерактивний календар для вибору дати;
- список тренувань, доданих на цей день;
- кнопку додавання нового тренування, розташовану в нижній частині екрану.

Екран додавання вправи передбачає ввід таких параметрів, як назва вправи, вибір групи м'язів, кількість підходів і повторень, а також вага. Дані перед збереженням проходять валідацію на коректність і повноту, що запобігає помилкам і підвищує надійність роботи застосунку.

Візуальне оформлення виконане у темних тонах із чорним фоном та жовтими акцентами. Використані контрастні шрифти і крупні кнопки роблять інтерфейс читабельним і зручним навіть у умовах недостатнього освітлення. Анімації появи елементів сприяють приємному сприйняттю і покращують користувацький досвід.

4.5 Функціональні модулі

Авторизація користувачів реалізована із збереженням облікових записів локально. Після введення логіна та пароля відбувається перевірка коректності даних, і у разі успіху користувач потрапляє на головний екран застосунку. Такий підхід забезпечує базовий рівень безпеки та персоналізації.

Додавання та редагування тренувань реалізовано із можливістю змінювати параметри вправ, включаючи кількість підходів, вагу і групу м'язів.

Всі зміни миттєво відображаються у списках, що робить взаємодію плавною та інтуїтивною. Також передбачена можливість видалення непотрібних записів, що допомагає підтримувати актуальність даних.

4.6 Валідація та безпека

Для забезпечення коректності введених даних застосовується валідація, яка:

- перевіряє правильність формату email;
- обмежує кількість підходів у межах від 1 до 20;
- вимагає додатне значення ваги;
- перевіряє заповненість обов'язкових полів.

Також безпека даних гарантується за рахунок локального зберігання без передачі інформації на сервер. Авторизація передбачає базову хешовану перевірку, що у подальшому може бути розширена інтеграцією з Firebase для підвищення рівня безпеки та зручності.

4.7 Використані інструменти та технології

Для розробки застосунку використано:

- Android Studio — основне середовище розробки;
- Java — мова програмування;

- SQLite — локальна база даних;
- XML-макети з Material Components — для інтерфейсу;
- GitHub — для контролю версій.

4.8 Перспективи розвитку

У майбутньому планується:

- інтеграція із хмарними сервісами (Firebase) для синхронізації даних користувачів;
- створення персоналізованих профілів та збереження прогресу у хмарі;
- додавання візуалізації статистичних даних: графіки прогресу, аналітика навантажень за тиждень;
- фільтри за періодами; впровадження сповіщень і нагадувань про тренування для підтримки мотивації.

4.9 Висновки до розділу 4

У цьому розділі наведено повний опис архітектури, структури, функціональності та інтерфейсу розробленого мобільного застосунку для обліку тренувань. Застосунок поєднує в собі зручність, гнучкість та ефективність, що робить його корисним інструментом для різних категорій користувачів. Обрана архітектура та технології дозволяють легко масштабувати додаток та впроваджувати нові можливості, забезпечуючи довготривалу підтримку та розвиток продукту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Fitness App Market Statistics 2022–2030 [Electronic resource] // Allied Market Research. – Access mode: <https://www.alliedmarketresearch.com/fitness-app-market>
2. Fitness app usage in the United States and worldwide [Electronic resource] // Statista. – Access mode: <https://www.statista.com>
3. MyFitnessPal. Official website [Electronic resource]. – Access mode: <https://www.myfitnesspal.com>
4. Strava. Run and Cycling Tracking App [Electronic resource]. – Access mode: <https://www.strava.com>
5. User reviews of fitness apps [Electronic resource] // Google Play. – Access mode: <https://play.google.com>
6. SQLite in Android with Room [Electronic resource] // Android Developers. – Access mode: <https://developer.android.com/training/data-storage/room>
7. Guidelines for Android fitness apps [Electronic resource] // Material Design. – Access mode: <https://m3.material.io>
8. Build apps for Android [Electronic resource] // Android Developers. – Access mode: <https://developer.android.com>
9. About SQLite [Electronic resource] // SQLite Documentation. – Access mode: <https://www.sqlite.org/about.html>
10. Material Design Components for Android [Electronic resource] // Google. – Access mode: <https://m3.material.io/components>
11. Kotlin vs Java: What's the Difference? [Electronic resource] // JetBrains. – Access mode: <https://kotlinlang.org/docs/comparison-to-java.html>
12. Mobile Operating Systems' Market Share Worldwide [Electronic resource] // Statista. – Access mode: <https://www.statista.com/statistics/272307/market-share-held-by-mobile-operating-systems-worldwide>

13. Android SQLite Database Tutorial [Electronic resource] // Vogella GmbH. – Access mode: <https://www.vogella.com/tutorials/AndroidSQLite/article.html>
14. Kotlin vs Java for Android Development [Electronic resource] // GeeksforGeeks. – Access mode: <https://www.geeksforgeeks.org/kotlin-vs-java-for-android-development/>
15. Material Components for Android GitHub Repository [Electronic resource] // GitHub. – Access mode: <https://github.com/material-components/material-components-android>
16. Platform Architecture [Electronic resource] // Android Open Source Project. – Access mode: <https://source.android.com/docs/core/architecture>
17. Usage statistics of Java for websites [Electronic resource] // W3Techs. – Access mode: <https://w3techs.com/technologies/details/pl-java>
18. Android Emulator documentation [Electronic resource] // Android Developers. – Access mode: <https://developer.android.com/studio/run/emulator>
19. Guide to App Architecture: MVVM [Electronic resource] // Android Developers. – Access mode: <https://developer.android.com/topic/architecture>
20. UI Testing with Espresso [Electronic resource] // Android Developers. – Access mode: <https://developer.android.com/training/testing/espresso>
21. Agile Software Development [Electronic resource] // Martin Fowler. – Access mode: <https://martinfowler.com/agile.html>
22. Silberschatz, A. Database System Concepts / A. Silberschatz, H. Korth, S. Sudarshan. – 7th ed. – New York: McGraw-Hill, 2019. – 1376 p.
23. Banerjee, A. Mastering Android Development with Kotlin / A. Banerjee. – Birmingham: Packt Publishing, 2020. – 766 p.
24. Андрушко, А. В. Розробка мобільних додатків на платформі Android / А. В. Андрушко. – Львів: Львівська політехніка, 2021. – 220 с.
25. Бондар, С. І. Основи програмування мобільних пристроїв на Java / С. І. Бондар, Ю. О. Шевченко. – Київ: НАУ, 2020. – 186 с.

26. Фрайтаг, Б. Мобільна розробка: сучасний підхід до UX-дизайну / Б. Фрайтаг. – Львів: Видавництво Старого Лева, 2021. – 248 с.
27. Захарченко, В. П. Безпека програмного забезпечення: підручник / В. П. Захарченко. – Київ: КНЕУ, 2019. – 230 с.
28. Бас, С. В. Архітектура програмного забезпечення / С. В. Бас. – Київ: КНУ, 2021. – 312 с.
29. Welling, L. PHP and MySQL Web Development / L. Welling, L. Thomson. – 5th ed. – Boston: Pearson Education, 2017. – 832 p.
30. Owens, M. The Definitive Guide to SQLite / M. Owens. – New York: Apress, 2010. – 304 p.
31. Best practices for input validation in Android apps [Electronic resource] // Stack Overflow. – Access mode: <https://stackoverflow.com/questions/1836125/how-to-validate-edittext-in-android>

ДОДАТОК А
Технічне завдання

Вступ

Назва програми: Мобільний застосунок для обліку тренувань(GymNote)

У сучасному світі зростає інтерес до активного способу життя, фізичного розвитку та підтримання здоров'я. Все більше людей регулярно займаються спортом, відвідують тренажерні зали, бігають або тренуються вдома. Для досягнення значущих результатів важливо не лише дотримуватися тренувального режиму, але й вести облік занять, аналізувати прогрес, визначати ефективність тренувань та планувати наступні кроки.

У зв'язку з цим виникає потреба в цифровому інструменті, який дозволив би систематизувати інформацію про тренування та зробити процес самоконтролю більш ефективним і зручним. Мобільний застосунок для обліку тренувань розроблено саме з цією метою — він автоматизує щоденний контроль за тренуваннями, дає змогу зручно створювати записи, переглядати історію занять та орієнтуватися у структурі виконаних вправ.

Застосунок реалізовано як нативний мобільний додаток для Android-пристроїв, що забезпечує стабільну роботу на різних моделях смартфонів. Функціональність програми включає реєстрацію користувача, додавання, редагування та видалення записів тренувань, перегляд історії занять, відображення тренувань у форматі календаря та фільтрацію за групами м'язів. Додаток має сучасний інтерфейс, оформлений у темній чорно-жовтій гамі, яка поєднує естетику та зручність.

Застосунок побудовано з використанням мови програмування Java, бази даних SQLite, архітектурного шаблону MVVM та Material UI-компонентів. Така архітектура забезпечує зрозумілу структуру коду, розділення логіки програми від інтерфейсу, полегшуючи подальше масштабування і супровід застосунку.

A.1 Підстава для розробки

Розробка мобільного застосунку для обліку тренувань зумовлена зростаючою потребою користувачів у цифрових інструментах для самостійного контролю фізичних навантажень, моніторингу результатів і планування тренувального процесу. У сучасних умовах дедалі більше людей займається спортом індивідуально — вдома, на вулиці або у тренажерному залі — без постійного нагляду тренера. Для таких користувачів особливо важливо мати простий і зручний спосіб фіксації тренувань, перегляду історії занять та аналізу змін у власній фізичній формі.

На ринку наявні аналоги здебільшого мають або надто складний функціонал, або вимагають платної підписки, або не дозволяють налаштувати тренування під індивідуальні потреби користувача. Більшість таких програм перевантажені додатковими функціями, що не є критичними для повсякденного використання, а інтерфейс може бути незрозумілим для початківців.

Враховуючи це, виникла необхідність створити мобільний додаток, який би:

- забезпечував зручний інтерфейс користувача;
- дозволяв швидко додавати, редагувати та видаляти тренування;
- надавав функцію перегляду тренувань у вигляді календаря;
- підтримував фільтрацію за групами м'язів;
- не вимагав постійного підключення до Інтернету, зберігаючи всі дані локально.

Основна підстава для виконання роботи сформульована у відповідності з Наказом № 209 від 28 квітня 2025 року, виданим у Національному університеті «Запорізька політехніка», де для виконання даної дипломної кваліфікаційної роботи бакалавра визначено «Мобільний застосунок для трекінгу спортивних тренувань» як тему роботи.

A.2 Вимоги до мобільного застосунку для обліку тренувань

A.2.1 Функціональні вимоги

Мобільний застосунок повинен забезпечувати користувачеві можливість ведення персонального обліку тренувальної активності через інтуїтивний інтерфейс та гнучкий набір функцій. Основні можливості програми включають:

- реєстрація та вхід: створення нового облікового запису з вказанням логіна та пароля, а також авторизація за збереженими даними;
- створення тренування: додавання нової сесії з вибором дати, вправ, кількості підходів, повторень, ваги, а також коментарів або нотаток;
- редагування і видалення: можливість коригувати або видаляти вже створені записи;
- календар тренувань: візуальне представлення активності користувача в календарному форматі з можливістю переходу по днях;
- категоризація вправ: групування вправ за цільовими м'язовими групами для швидкого пошуку та навігації;
- історія тренувань: перегляд попередніх сесій у вигляді списку або календаря;
- темна тема інтерфейсу: реалізація темного режиму для зручного використання в умовах недостатнього освітлення.

A.2.2 Вимоги до надійності

Програма має гарантувати стабільну роботу за будь-яких сценаріїв взаємодії з користувачем. З цією метою необхідно дотримуватись таких умов:

- перевірка правильності введених даних (запобігання збереженню порожніх або некоректних полів);
- обробка нештатних ситуацій без аварійного завершення програми;

- збереження цілісності локальної бази даних навіть у разі раптового завершення сесії;
- стабільна робота з великим обсягом даних (десятки або сотні тренувань);
- відсутність втрати даних при повторному запуску програми або оновленні системи.

A.2.3 Платформні та експлуатаційні умови

Мобільний застосунок створено під операційну систему Android та орієнтований на сучасні смартфони. Мінімальні вимоги:

- операційна система: Android 8.0 або вище;
- оперативна пам'ять: не менше 2 ГБ;
- вільне місце для встановлення: від 100 МБ;
- дозволи: доступ до локального сховища (для збереження бази даних), інтернет-підключення не є обов'язковим.

A.2.4 Технологічні вимоги

Для розробки та запуску застосунку використовуються наступні технології:

- мова програмування: Java;
- база даних: SQLite;
- архітектура: MVVM (Model-View-ViewModel);
- бібліотеки: Material Components для оформлення інтерфейсу;
- IDE: Android Studio (версія 2022.1 або вище);
- додаткові вимоги: використання XML-розмітки для UI, підтримка адаптивного дизайну.

A.2.5 Сумісність та портативність

Застосунок спроектовано таким чином, щоб бути максимально сумісним із широким спектром пристроїв та не залежати від зовнішніх середовищ чи сервісів. Основні характеристики сумісності такі:

- застосунок не вимагає встановлення сторонніх бібліотек чи компонентів;
- уся логіка та інтерфейс реалізовані нативними засобами Android SDK;
- програма функціонує повністю автономно без підключення до серверів;
- дані зберігаються у внутрішньому сховищі, що не потребує доступу до хмарних сервісів.

A.2.6 Розповсюдження та зберігання

Для зручності користувачів та забезпечення надійного збереження застосунок поширюється у вигляді автономного інсталяційного пакету. Порядок розповсюдження та способи збереження такі:

- застосунок розповсюджується у вигляді APK-файлу з інструкцією встановлення;
- усі компоненти (бібліотеки, ресурси, шрифти, зображення) включено до складу пакету;
- архів з проєктом та арк-файлом може бути збережений на USB-носії або в хмарному сховищі для захисту від втрати.

A.3 Потенційні ризики при експлуатації та заходи їхнього усунення

У процесі експлуатації мобільного застосунку можуть виникати різноманітні ризики, пов'язані як з технічними обмеженнями мобільного

середовища, так і з людським фактором — неправильними діями користувача. З метою забезпечення стабільної та безпечної роботи застосунку важливо заздалегідь передбачити потенційні проблеми та способи їхнього усунення або мінімізації. У таблиці нижче наведено основні типові ризики, їхні можливі наслідки для користувача та передбачені технічні рішення (табл. А.1).

Таблиця А.1 — Таблиця ризиків

Потенційний ризик	Можливі наслідки	Запропоноване рішення
Видалення застосунку без резервного копіювання	Повна втрата даних користувача	Можливість локального експорту/імпорту бази
Некоректне введення даних	Помилки у збереженні або обробці інформації	Валідація вводу на всіх етапах
Сумісність із майбутніми версіями Android	Порушення роботи застосунку	Регулярна перевірка та оновлення залежностей
Випадкове видалення запису	Втрата важливої інформації про тренування	Додаткове підтвердження при видаленні
Перевантаження бази через велику кількість записів	Зниження продуктивності	Оптимізація запитів, обмеження на обсяг даних

А.4 Вимоги до продуктивності

Основні вимоги до продуктивності застосунку:

- час запуску — не більше 2 секунд;
- реакція на дії користувача — до 200 мс;
- плавне відображення анімацій при зміні екранів;
- використання Room або SQLite без витоків пам'яті;
- забезпечення стабільної роботи при ≥ 500 записах у базі.

Для досягнення цих показників застосовуються:

- lazy-завантаження елементів списку (RecyclerView);
- ViewModel з LiveData для відв'язування UI від логіки;
- профілювання за допомогою Android Profiler та Logcat.

A.5. Вимоги до безпеки

Хоча застосунок не містить конфіденційної інформації, передбачені базові заходи захисту:

- заборона доступу до бази даних іншими застосунками (встановлений `MODE_PRIVATE`);
- уникнення SQL-ін'єкцій (використання `SQLiteStatement` або `Room DAO`);
- можливість обмеження доступу до програми за допомогою PIN-коду (опціонально);
- шифрування резервної копії (в майбутніх версіях).

ДОДАТОК Б
Текст програми

Б.1 DBHelper.java

```

public boolean updateWorkout(int workoutId, int userId, String
muscleGroup, String exercise, String setsReps, String weight,
String date) {
    SQLiteDatabase db = this.getWritableDatabase();

    String[] parts = setsReps.split("x");
    int sets = 0, reps = 0;
    try {
        if (parts.length == 2) {
            sets = Integer.parseInt(parts[0].trim());
            reps = Integer.parseInt(parts[1].trim());
        } else {
            return false;
        }
    } catch (NumberFormatException e) {
        e.printStackTrace();
        return false;
    }

    try {
        db.beginTransaction();

        int muscleGroupId = getOrCreateMuscleGroup(db,
muscleGroup);
        int exerciseId = getOrCreateExercise(db, exercise,
muscleGroupId);
        int dayId = getOrCreateDay(db, userId,
normalizeDate(date));

        ContentValues cv = new ContentValues();
        cv.put("exercise_id", exerciseId);
        cv.put("sets", sets);
        cv.put("reps", reps);
    }
}

```

```

        try {
            double w = Double.parseDouble(weight);
            cv.put("weight", w);
        } catch (NumberFormatException e) {
            cv.putNull("weight");
        }

        cv.put("day_id", dayId);

        int rows = db.update("workout_entries", cv, "id = ?",
new String[]{String.valueOf(workoutId)});

        if (rows > 0) {
            db.setTransactionSuccessful();
            return true;
        } else {
            return false;
        }
    } finally {
        db.endTransaction();
    }
}

```

```

public List<Workout> getWorkoutsForDate(int userId, String
inputDate) {
    List<Workout> workouts = new ArrayList<>();
    SQLiteDatabase db = this.getReadableDatabase();

    String date = normalizeDate(inputDate);
    Log.d("DBHelper", "Querying workouts for date: " + date);

    String query = "SELECT we.id, wg.name AS muscle_group,
e.name AS exercise, we.sets, we.reps, we.weight, d.date " +
        "FROM workout_entries we " +

```

```

        "JOIN exercises e ON we.exercise_id = e.id " +
        "JOIN muscle_groups wg ON e.muscle_group_id = wg.id
" +

        "JOIN days d ON we.day_id = d.id " +
        "WHERE d.date LIKE ? AND d.user_id = ?";

    Cursor cursor = db.rawQuery(query, new String[]{date + "%",
String.valueOf(userId)});

    while (cursor.moveToNext()) {
        int id = cursor.getInt(0);
        String muscleGroup = cursor.getString(1);
        String exercise = cursor.getString(2);
        int sets = cursor.getInt(3);
        int reps = cursor.getInt(4);
        double weight = cursor.getDouble(5);
        String workoutDate = cursor.getString(6);

        workouts.add(new Workout(id, muscleGroup, exercise, sets
+ "x" + reps, String.valueOf(weight), workoutDate));
    }

    cursor.close();
    return workouts;
}

public void insertWorkout(int userId, String muscleGroupName,
String exerciseName, int sets, int reps, double weight, String
rawDate) {
    SQLiteDatabase db = this.getWritableDatabase();
    String date = formatDate(rawDate);

    int muscleGroupId = getOrCreateMuscleGroup(db,
muscleGroupName);
    int exerciseId = getOrCreateExercise(db, exerciseName,
muscleGroupId);
    int dayId = getOrCreateDay(db, userId, date);

```

```

        ContentValues values = new ContentValues();
        values.put("day_id", dayId);
        values.put("exercise_id", exerciseId);
        values.put("sets", sets);
        values.put("reps", reps);
        values.put("weight", weight);
        db.insert("workout_entries", null, values);
    }

    public boolean deleteWorkoutById(int workoutId) {
        SQLiteDatabase db = this.getWritableDatabase();
        int deletedRows = db.delete("workout_entries", "id = ?",
new String[]{String.valueOf(workoutId)});
        return deletedRows > 0;
    }

    private String normalizeDate(String inputDate) {
        try {
            if (inputDate.contains(" ")) {
                inputDate = inputDate.split(" ")[0];
            }
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd", Locale.getDefault());
            Date date = sdf.parse(inputDate);
            return sdf.format(date);
        } catch (Exception e) {
            Log.e("DBHelper", "Error normalizing date: " +
inputDate, e);
            return inputDate;
        }
    }

    private String formatDate(String inputDate) {
        try {
            if (inputDate.contains(" ")) {

```

```

        inputDate = inputDate.split(" ")[0];
    }
    SimpleDateFormat parser = new
SimpleDateFormat("yyyy-MM-dd", Locale.getDefault());
    Date date = parser.parse(inputDate);
    return parser.format(date);
} catch (Exception e) {
    return inputDate;
}
}

private int getOrCreateDay(SQLiteDatabase db, int userId,
String date) {
    Cursor cursor = db.rawQuery("SELECT id FROM days WHERE
user_id = ? AND date = ?", new String[]{String.valueOf(userId),
date});
    if (cursor.moveToFirst()) {
        int id = cursor.getInt(0);
        cursor.close();
        return id;
    }
    cursor.close();

    ContentValues cv = new ContentValues();
    cv.put("user_id", userId);
    cv.put("date", date);
    return (int) db.insert("days", null, cv);
}

private int getOrCreateMuscleGroup(SQLiteDatabase db, String
name) {
    Cursor cursor = db.rawQuery("SELECT id FROM
muscle_groups WHERE name = ?", new String[]{name});
    if (cursor.moveToFirst()) {
        int id = cursor.getInt(0);
        cursor.close();
    }
}

```

```

        return id;
    }
    cursor.close();

    ContentValues cv = new ContentValues();
    cv.put("name", name);
    return (int) db.insert("muscle_groups", null, cv);
}

private int getOrCreateExercise(SQLiteDatabase db, String
name, int muscleGroupId) {
    Cursor cursor = db.rawQuery("SELECT id FROM exercises
WHERE name = ? AND muscle_group_id = ?", new String[]{name,
String.valueOf(muscleGroupId)});
    if (cursor.moveToFirst()) {
        int id = cursor.getInt(0);
        cursor.close();
        return id;
    }
    cursor.close();

    ContentValues cv = new ContentValues();
    cv.put("name", name);
    cv.put("muscle_group_id", muscleGroupId);
    return (int) db.insert("exercises", null, cv);
}
}

```

B.2 CalendarAdapter.java

```

public void onBindViewHolder(@NonNull CalendarViewHolder holder,
int position) {
    Calendar date = (Calendar) baseDate.clone();
    date.add(Calendar.DAY_OF_YEAR, position - CENTER_POSITION);
}

```

```

// Обнуляем время
date.set(Calendar.HOUR_OF_DAY, 0);
date.set(Calendar.MINUTE, 0);
date.set(Calendar.SECOND, 0);
date.set(Calendar.MILLISECOND, 0);

String dayName = new SimpleDateFormat("E", new
Locale("uk")).format(date.getTime());
int dayOfMonth = date.get(Calendar.DAY_OF_MONTH);
int monthIndex = date.get(Calendar.MONTH);

String dayNumber;
if (dayOfMonth == 1) {
    String[] months = {"Січ", "Лют", "Бер", "Кві", "Тра",
"Чер",
        "Лип", "Сер", "Вер", "Жов", "Лис", "Гру"};
    String monthName = months[monthIndex];
    dayNumber = dayOfMonth + "\n" + monthName;
} else {
    dayNumber = String.valueOf(dayOfMonth);
}

holder.dayName.setText(dayName);
holder.dayNumber.setText(dayNumber);
holder.itemView.setSelected(position == selectedPosition);

holder.itemView.setOnClickListener(v -> {
    int adapterPosition = holder.getAdapterPosition();
    if (adapterPosition == RecyclerView.NO_POSITION) return;

    selectDateAtPosition(adapterPosition);
});
}

@Override

```

```

public int getItemCount() {
    return ITEM_COUNT;
}

// Новый метод — выбор даты программно
public void selectDateAtPosition(int position) {
    if (position < 0 || position >= ITEM_COUNT) return;

    int prev = selectedPosition;
    selectedPosition = position;
    notifyItemChanged(prev);
    notifyItemChanged(selectedPosition);

    Calendar selectedDate = (Calendar) baseDate.clone();
    selectedDate.add(Calendar.DAY_OF_YEAR, selectedPosition -
CENTER_POSITION);

    // Обнуляем время при выборе даты
    selectedDate.set(Calendar.HOURL_OF_DAY, 0);
    selectedDate.set(Calendar.MINUTE, 0);
    selectedDate.set(Calendar.SECOND, 0);
    selectedDate.set(Calendar.MILLISECOND, 0);

    Log.d("CalendarAdapter", "Selected date: " +
        new SimpleDateFormat("yyyy-MM-dd",
Locale.getDefault()).format(selectedDate.getTime()));

    if (listener != null) {
        listener.onDateSelected(selectedDate);
    }
}

static class CalendarViewHolder extends RecyclerView.ViewHolder
{
    TextView dayName;
    TextView dayNumber;
}

```

```

CalendarViewHolder(@NonNull View itemView) {
    super(itemView);
    dayName = itemView.findViewById(R.id.tvDayName);
    dayNumber = itemView.findViewById(R.id.tvDayNumber);
}
}

```

Б.3 AddWorkoutActivity.java

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_add_workout);

    dbHelper = new DBHelper(this);

    etMuscleGroup = findViewById(R.id.etMuscleGroup);
    etExercise = findViewById(R.id.etExercise);
    etSetsReps = findViewById(R.id.etSetsReps);
    etWeight = findViewById(R.id.etWeight);
    etDate = findViewById(R.id.etDate);
    btnSave = findViewById(R.id.btnSave);

    String dateFromIntent =
    getIntent().getStringExtra("selectedDate");
    if (dateFromIntent != null && !dateFromIntent.isEmpty())
    {
        etDate.setText(dateFromIntent);
        Log.d("AddWorkout", "Дата из интента: " +
dateFromIntent);
    } else {
        // если дата не передана, устанавливаем текущую
        String today = new SimpleDateFormat("yyyy-MM-dd",
Locale.getDefault())
            .format(Calendar.getInstance().getTime());
        etDate.setText(today);
    }

    SharedPreferences prefs =
    getSharedPreferences("MyPrefs", MODE_PRIVATE);
    userId = prefs.getInt("user_id", -1);
    if (userId == -1) {
        Toast.makeText(this, "\n" + "Помилка: користувача не
знайдено", Toast.LENGTH_SHORT).show();
        finish();
        return;
    }
}

```

```

    }

    etDate.setOnClickListener(v -> {
        final Calendar calendar = Calendar.getInstance();
        int year = calendar.get(Calendar.YEAR);
        int month = calendar.get(Calendar.MONTH);
        int day = calendar.get(Calendar.DAY_OF_MONTH);

        DatePickerDialog datePickerDialog = new
        DatePickerDialog(this, (view, y, m, d) -> {
            String selectedDate =
            String.format(Locale.getDefault(), "%04d-%02d-%02d", y, m + 1,
            d);

            etDate.setText(selectedDate);
        }, year, month, day);
        datePickerDialog.show();
    });

    btnSave.setOnClickListener(v -> {
        String muscleGroup =
        etMuscleGroup.getText().toString().trim();
        String exercise =
        etExercise.getText().toString().trim();
        String setsReps =
        etSetsReps.getText().toString().trim();
        String weightStr =
        etWeight.getText().toString().trim();
        String date = etDate.getText().toString().trim();

        if (muscleGroup.isEmpty() || exercise.isEmpty() ||
        setsReps.isEmpty() || date.isEmpty()) {
            Toast.makeText(this, "\n" + "Будь ласка,
            заповніть усі обов'язкові поля!", Toast.LENGTH_SHORT).show();
            return;
        }

        String[] parts = setsReps.toLowerCase().split("x");
        if (parts.length != 2) {
            Toast.makeText(this, "Неправильний формат для
            підходів та повторів. Використовуйте формат '3x12'",
            Toast.LENGTH_SHORT).show();
            return;
        }

        int sets, reps;
        try {
            sets = Integer.parseInt(parts[0].trim());
            reps = Integer.parseInt(parts[1].trim());
        } catch (NumberFormatException e) {
            Toast.makeText(this, "Неправильний формат чисел
            у підходах та повторях", Toast.LENGTH_SHORT).show();
            return;
        }
    }

```

```

        double weight = 0;
        try {
            weight = weightStr.isEmpty() ? 0 :
Double.parseDouble(weightStr);
        } catch (NumberFormatException e) {
            Toast.makeText(this, "\n" + "Введіть коректну
вартість", Toast.LENGTH_SHORT).show();
            return;
        }

        try {
            dbHelper.insertWorkout(userId, muscleGroup,
exercise, sets, reps, weight, date);

            Log.d("AddWorkout", "Додана тренівка: " +
date + ", " + muscleGroup + ", " + exercise);

            Toast.makeText(this, "\n" + "Додано
тренування", Toast.LENGTH_SHORT).show();

            Intent resultIntent = new Intent();
            resultIntent.putExtra("date", date);
            setResult(Activity.RESULT_OK, resultIntent);
            finish();
        } catch (Exception e) {
            Toast.makeText(this, "\n" + "Помилка при
збереженні:" + e.getMessage(), Toast.LENGTH_SHORT).show();
            Log.e("AddWorkout", "Помилка при додаванні
тренування", e);
        }
    });
}
}

```

Б.4 DashboardActivity.java

```

private void initUI() {
    calendarRecyclerView =
findViewById(R.id.calendarRecyclerView);
    calendarRecyclerView.setLayoutManager(new
LinearLayoutManager(this, LinearLayoutManager.HORIZONTAL,
false));

    workoutList = findViewById(R.id.workoutList);
    workoutList.setLayoutManager(new LinearLayoutManager(this));

    addWorkoutButton = findViewById(R.id.addWorkoutButton);
    addWorkoutButton.setOnClickListener(v -> {
        Intent intent = new Intent(this,
AddWorkoutActivity.class);
    });
}

```

```

        intent.putExtra("selectedDate", selectedDateString);
        startActivityForResult(intent,
ADD_WORKOUT_REQUEST_CODE);
    });
}

private void setupCalendar() {
    Calendar baseDate = Calendar.getInstance();
    baseDate.set(Calendar.YEAR, 2025);
    baseDate.set(Calendar.MONTH, Calendar.MAY);
    baseDate.set(Calendar.DAY_OF_MONTH, 15);
    baseDate.set(Calendar.HOUR_OF_DAY, 0);
    baseDate.set(Calendar.MINUTE, 0);
    baseDate.set(Calendar.SECOND, 0);
    baseDate.set(Calendar.MILLISECOND, 0);

    CalendarAdapter adapter = new CalendarAdapter(this, date ->
{
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-
dd", Locale.getDefault());
    selectedDateString = sdf.format(date.getTime());
    Log.d("Dashboard", "Выбрана дата: " +
selectedDateString);
    loadWorkoutsFromDB(selectedDateString);
}, baseDate);

    calendarRecyclerView.setAdapter(adapter);
    calendarRecyclerView.post(() -> {

calendarRecyclerView.scrollToPosition(adapter.CENTER_POSITION);
        adapter.selectDateAtPosition(adapter.CENTER_POSITION);
    });
}

private void loadWorkoutsFromDB(String date) {
    List<Workout> loadedWorkouts =
dbHelper.getWorkoutsForDate(userId, date);
    workouts.clear();
    workouts.addAll(loadedWorkouts);

    if (workoutAdapter == null) {
        workoutAdapter = new WorkoutAdapter(workouts, new
WorkoutAdapter.OnWorkoutActionListener() {
            @Override
            public void onEdit(Workout workout, int position) {
                launchEditWorkout(workout);
            }

            @Override
            public void onDelete(Workout workout, int position)
{
                if (dbHelper.deleteWorkoutById(workout.getId()))
{

```

```

        workoutAdapter.removeItem(position);
        Log.d("Dashboard", "\n" + "Видалено
тренування ID: " + workout.getId());
    } else {
        Log.e("Dashboard", "Помилка при видаленні
тренування з БД");
    }
}

});
workoutList.setAdapter(workoutAdapter);
} else {
    workoutAdapter.updateData(loaderWorkouts);
}

Log.d("Dashboard", "\n" + "Завантажено " +
loaderWorkouts.size() + " \n" + "тренувань на дату " + date);
}

private void launchEditWorkout(Workout workout) {
    Intent intent = new Intent(this, EditWorkoutActivity.class);
    intent.putExtra("workoutId", workout.getId());
    intent.putExtra("userId", userId); // <- Додаємо сюди
userId
    intent.putExtra("muscleGroup", workout.getMuscleGroup());
    intent.putExtra("exercise", workout.getExercise());
    intent.putExtra("setsReps", workout.getSetsReps());
    intent.putExtra("weight", workout.getWeight());
    intent.putExtra("date", workout.getDate());
    startActivityForResult(intent, EDIT_WORKOUT_REQUEST_CODE);
}

```

Б.5 EditWorkoutActivity.java

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_edit_workout);

    etMuscleGroup = findViewById(R.id.etMuscleGroup);
    etExercise = findViewById(R.id.etExercise);
    etSetsReps = findViewById(R.id.etSetsReps);
    etWeight = findViewById(R.id.etWeight);
    btnSave = findViewById(R.id.btnSave);

    Intent intent = getIntent();
    workoutId = intent.getIntExtra("workoutId", -1);
}

```

```

int userId = intent.getIntExtra("userId", -1);
date = intent.getStringExtra("date");

if (workoutId == -1 || userId == -1) {
    Toast.makeText(this, "Ошибка: не найден ID тренировки
или пользователя", Toast.LENGTH_SHORT).show();
    finish();
    return;
}

etMuscleGroup.setText(intent.getStringExtra("muscleGroup"));
etExercise.setText(intent.getStringExtra("exercise"));
etSetsReps.setText(intent.getStringExtra("setsReps"));
etWeight.setText(intent.getStringExtra("weight"));

btnSave.setOnClickListener(v -> {
    String muscleGroup =
etMuscleGroup.getText().toString().trim();
    String exercise =
etExercise.getText().toString().trim();
    String setsReps =
etSetsReps.getText().toString().trim();
    String weight = etWeight.getText().toString().trim();

    if (muscleGroup.isEmpty() || exercise.isEmpty() ||
setsReps.isEmpty() || weight.isEmpty()) {
        Toast.makeText(this, "Пожалуйста, заполните все
поля", Toast.LENGTH_SHORT).show();
        return;
    }

    DBHelper dbHelper = new DBHelper(this);
    // Обновляем тренировку с передачей userId и date
    boolean success = dbHelper.updateWorkout(workoutId,
userId, muscleGroup, exercise, setsReps, weight, date);

```

```

        if (success) {
            Intent resultIntent = new Intent();
            resultIntent.putExtra("date", date);
            setResult(RESULT_OK, resultIntent);
            finish();
        } else {
            Toast.makeText(this, "Ошибка при сохранении",
Toast.LENGTH_SHORT).show();
        }
    });
}

```

B.6 WorkoutAdapter.java

```

public void onBindViewHolder(@NonNull WorkoutViewHolder holder,
int position) {
    Workout workout = workouts.get(position);
    holder.muscleGroup.setText(workout.getMuscleGroup());
    holder.exercise.setText(workout.getExercise());
    holder.setsReps.setText(workout.getSetsReps());
    holder.weight.setText(workout.getWeight());

    if (!animatedPositions.contains(position)) {

holder.itemView.startAnimation(AnimationUtils.loadAnimation(hold
er.itemView.getContext(), android.R.anim.slide_in_left));
        animatedPositions.add(position);
    }

    holder.moreBtn.setOnClickListener(v -> {
        PopupMenu popup = new PopupMenu(v.getContext(), v);
        popup.getMenuInflater().inflate(R.menu.workout_menu,
popup.getMenu());
        popup.setOnMenuItemClickListener(item -> {
            int itemId = item.getItemId();
            if (itemId == R.id.menu_edit) {
                listener.onEdit(workout,
holder.getAdapterPosition());
                return true;
            } else if (itemId == R.id.menu_delete) {
                listener.onDelete(workout,
holder.getAdapterPosition());
                return true;
            }
            return false;
        }
    });
}

```

```

        });
        popup.show();
    });
}

@Override
public int getItemCount() {
    return workouts.size();
}

public void removeItem(int position) {
    workouts.remove(position);
    notifyItemRemoved(position);
}

public static class WorkoutViewHolder extends
RecyclerView.ViewHolder {
    TextView muscleGroup, exercise, setsReps, weight;
    ImageView moreBtn;

    public WorkoutViewHolder(@NonNull View itemView) {
        super(itemView);
        muscleGroup =
itemView.findViewById(R.id.tvMuscleGroup);
        exercise = itemView.findViewById(R.id.tvExercise);
        setsReps = itemView.findViewById(R.id.tvSetsReps);
        weight = itemView.findViewById(R.id.tvWeight);
        moreBtn = itemView.findViewById(R.id.btnMore);
    }
}
}

```

ДОДАТОК В
Опис програми

В.1 Загальна характеристика застосунок

Розроблений мобільний застосунок за темою «Розробка мобільного застосунок для обліку тренувань» призначений для забезпечення зручного обліку, планування й аналізу тренувального процесу користувачів. Основна мета програми — дати користувачам можливість фіксувати свої фізичні активності, відслідковувати динаміку, переглядати історію тренувань та взаємодіяти з інтерфейсом на основі календаря.

Програма реалізована як Android-застосунок, написаний мовою Java з використанням архітектурного патерну MVVM, бази даних SQLite, UI-компонентів з бібліотеки Material Components та адаптований під темну чорно-жовту візуальну гаму, що відповідає стилістиці спортивного середовища.

Застосунок орієнтований на аматорів і професійних спортсменів, які бажають організовано вести облік фізичних навантажень, а також мати швидкий доступ до інформації про свої тренування без необхідності підключення до інтернету. Програма забезпечує локальне збереження даних, зручний інтерфейс додавання, редагування й перегляду записів, а також інтегрує інтуїтивну систему навігації.

В.2 Функціональне призначення

Мобільний застосунок призначений для ведення персонального щоденника тренувань з можливістю додавання, редагування, перегляду та видалення інформації про вправи, дати та м'язові групи. Основна мета — створити інтуїтивно зрозумілий і ефективний інструмент для моніторингу тренувального процесу без підключення до інтернету.

До основних функціональних можливостей програми належать:

- додавання нових тренувань з вибором дати, вправ і цільової м'язової групи;

- використання календаря для перегляду тренувань у хронологічному порядку;
- перегляд списку вправ за кожну дату;
- можливість редагування та видалення записів;
- адаптований під темну тему інтерфейс із сучасними елементами UI;
- локальне збереження даних у SQLite, що дозволяє працювати без інтернет-з'єднання.

Таким чином, GymNote забезпечує мінімалістичний і ефективний підхід до фіксації фізичної активності та ведення особистого фітнес-журналу.

В.3 Логічна структура та архітектура

Архітектура програми побудована на основі патерну MVVM (Model-View-ViewModel), проте для взаємодії з базою даних використовується чистий SQLite, без використання бібліотеки Room. Це дозволяє більш гнучко контролювати SQL-запити, однак потребує ручного управління транзакціями, створенням таблиць та обробкою курсорів.

Основні компоненти системи:

- Model: класи-сутності WorkoutEntry, Exercise, MuscleGroup, Day, які відображають структуру даних та таблиць у базі;
- View: активності та layout-файли, які забезпечують відображення інтерфейсу (додавання/редагування тренувань, календар, список вправ);
- ViewModel: посередники між логікою додатку та інтерфейсом, що реалізують бізнес-логіку та обробку даних;
- DatabaseHelper: клас для керування SQLite-базою, створення таблиць, виконання SQL-запитів (INSERT, SELECT, UPDATE, DELETE), обробки курсорів;

- UI компоненти: елементи інтерфейсу на основі Material Components — кнопки, картки, поля введення, списки, діалоги підтвердження, а також анімації для плавного відображення інформації.

Структура бази даних реалізується через SQL-запити у класі DatabaseHelper, який успадковує SQLiteOpenHelper. Уся інформація зберігається локально у базі даних SQLite, що забезпечує автономність роботи додатку. Дані структуровані у таблицях users, days, workout_entries, exercises, muscle_groups, що дозволяє реалізувати логіку зв'язків між датами, вправами та групами м'язів.

В.4 Використані технології та середовище розробки

Апаратне забезпечення (для тестування і розробки):

- процесор: AMD Ryzen 5 5600H / Intel Core i5 (аналогічний);
- оперативна пам'ять: 8–16 ГБ DDR4;
- накопичувач: SSD $\geq$ 256 ГБ;
- екран: $\geq$ 6.0", роздільна здатність FullHD (для мобільного тестування);
- ОС: Android 9.0+ (для цільових пристроїв), Windows 10/11 (для розробки).

Програмне забезпечення:

- Android Studio — середовище розробки;
- Java — основна мова програмування;
- XML — опис інтерфейсів;
- SQLite — вбудована база даних для Android (без Room);
- SQLiteOpenHelper — базовий клас для роботи з БД;
- Gradle — система автоматичної збірки;
- Material Components for Android — UI-бібліотека;
- Git — система контролю версій; AVD / реальний пристрій — для тестування роботи програми.

Додаток Г
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Розробка мобільного додатку для обліку тренувань

Виконав: студент групи КНТ-121 , Андрій МАНЖУРА
Керівник: професор , Михайло ПОЛЯКОВ

Рисунок Г.1 – Слайд 1

Об'єкт, предмет і мета роботи

Об'єкт дослідження –
процес автоматизованого
обліку фізичних тренувань
користувача.

Предмет дослідження –
мобільний застосунок для
систематизації, аналізу та
збереження даних про
тренування.

Метою роботи є створення зручного у
використанні мобільного додатку для Android-
платформи, що дозволяє користувачам
реєструвати та переглядати фізичні вправи,
враховуючи групи м'язів, кількість підходів,
повторень, вагу та дату виконання.

Рисунок Г.2 – Слайд 2

Порівняння аналогів застосунку

Назва додатку	Основні функції	Орієнтація	Недоліки
MyFitnessPal	Харчування, тренування, база продуктів	Для схуднення та харчового контролю	Складний інтерфейс, обмеження у безкоштовній версії
Strava	Відстеження бігу і велоспорту, соціальна	Аеробні види спорту	Не придатний для силових тренувань
JEFIT	Планування силових вправ, база вправ	Тренажерний зал, бодібілдинг	Заплутаний інтерфейс, реклама
Fitbod	Персоналізовані тренування	Автоматизація підбору вправ	Обмежена безкоштовна версія
Fitify	Домашні тренування, без обладнання	Початківці, домашні вправи	Обмежена кастомізація

Рисунок Г.3 – Слайд 3

Вимоги до застосунку

Функціональні вимоги

- Реєстрація нового користувача з введенням імені, email та паролю;
- Авторизація користувача за логіном і паролем;
- Можливість створення, перегляду, редагування та видалення тренувань;
- Збереження параметрів кожної вправи (назва, група м'язів, кількість підходів, повторень, вага);
- Візуалізація календаря з позначеними днями тренувань; Підтримка шаблонів тренувань;
- Фільтрація та сортування тренувань за різними критеріями (дата, група м'язів тощо);
- Ведення базової статистики та аналітики тренувального процесу;

Інтерфейсні вимоги

- простий, лаконічний і логічно структурований інтерфейс;
- повна підтримка темної теми оформлення (основні кольори — чорний фон, жовті акценти);
- екрани входу та реєстрації повинні бути інтуїтивно зрозумілими, з великими кнопками й інформативними підказками;
- основний екран має містити інтерактивний календар та список тренувань з візуальними підказками;
- усі кнопки та елементи керування — контрастні, зручні для натискання на екранах смартфонів;
- інтерфейс повинен бути адаптивним і однаково коректно виглядати на екранах різного розміру;
- візуальний стиль має відображати спортивну тематику — темний фон, фото залізного спорядження, яскраві акценти;
- використання іконок (гантелі, м'язи тощо) для покращення сприйняття.

Нефункціональні вимоги

- Стабільна робота на Android-пристроях версії 7.0 (Nougat) і вище;
- Повноцінна офлайн-робота без необхідності постійного підключення до Інтернету;
- Мінімальне використання ресурсів пристрою (RAM, CPU, battery);
- Швидкий час запуску та висока чутливість до взаємодії.

Рисунок Г.4 – Слайд 4

Порівняння мов програмування та обґрунтування вибору

Критерій	Java	Kotlin	Dart(Flutter)	C# (Xamarin)
Офіційна підтримка Google	Так	Так	Так(через Flutter)	Ні(від Microsoft)
Читабельність коду	Відносно складна	Компактна та сучасна	Зручна	Складна(через шаблони Xamarin)
Документація та спільнота	Дуже велика	Швидко зростає	Активна	Менша, ніж Java/Kotlin
Продуктивність	Висока	Висока	Висока	Середня
Інтеграція з Android	Повна	Повна	Часткова (через Flutter SDK)	Часткова
Підтримка Android Studio	Повна	Повна	Так	Обмежена
Офлайн-робота	Так	Так	Так	Так

Рисунок Г.5 – Слайд 5

Обґрунтування вибору середовища розробки

Критерій	Android Studio (використано)	Альтернатива: Visual Studio Code	Обґрунтування вибору
Підтримка Android SDK	Повна інтеграція	Потрібна ручна конфігурація	Android Studio створено спеціально для Android-розробки
Інтеграція з емуляторами	Вбудовані емулятори, AVD Manager	Зовнішні плагіни	З Android Studio налаштування простіші та стабільніші
Інструменти UI-дизайну	Візуальний редактор, Material Components	Обмежені плагіни	Прискорює розробку інтерфейсу користувача
Підтримка Java/Kotlin	Повна (офіційно підтримується Google)	Через зовнішні компілятори	Забезпечує стабільність та глибоку інтеграцію
Налагодження (debugging)	Глибоке, інтегроване	Обмежене	Зручний перегляд логів, зупинки, call stack
Профілювання продуктивності	Вбудовані інструменти	Відсутні або сторонні рішення	Дозволяє оптимізувати продуктивність застосунку
Автоматична збірка (Gradle)	Повна підтримка	Лімітовано через CLI	Зручне керування залежностями
Підтримка архітектури MVVM	Через шаблони, бібліотеки Jetpack	Потрібна ручна інтеграція	Прискорює реалізацію архітектури
Готовність до релізу	Має засоби підпису APK, оптимізації	Потрібно налаштовувати вручну	Android Studio полегшує фінальну підготовку до релізу

Рисунок Г.6 – Слайд 6

База даних мобільного застосунку

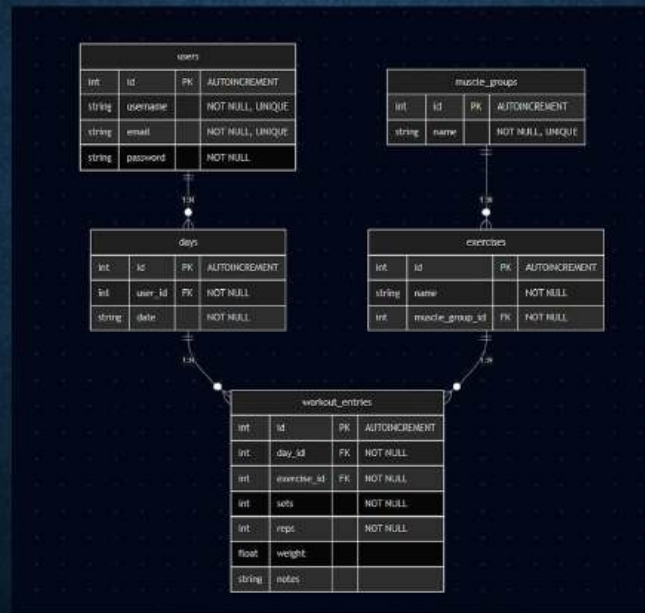


Рисунок Г.7 – Слайд 7

Зв'язок класів мобільного застосунку



Рисунок Г.8 – Слайд 8

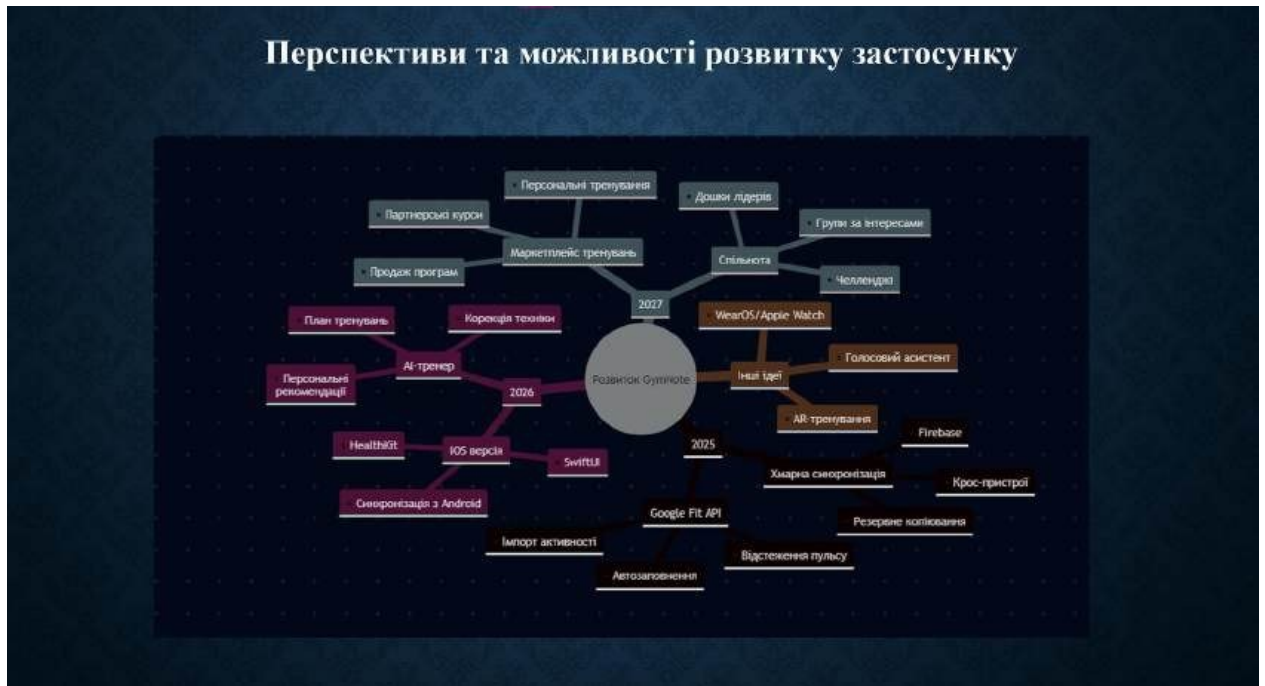


Рисунок Г.9 – Слайд 9

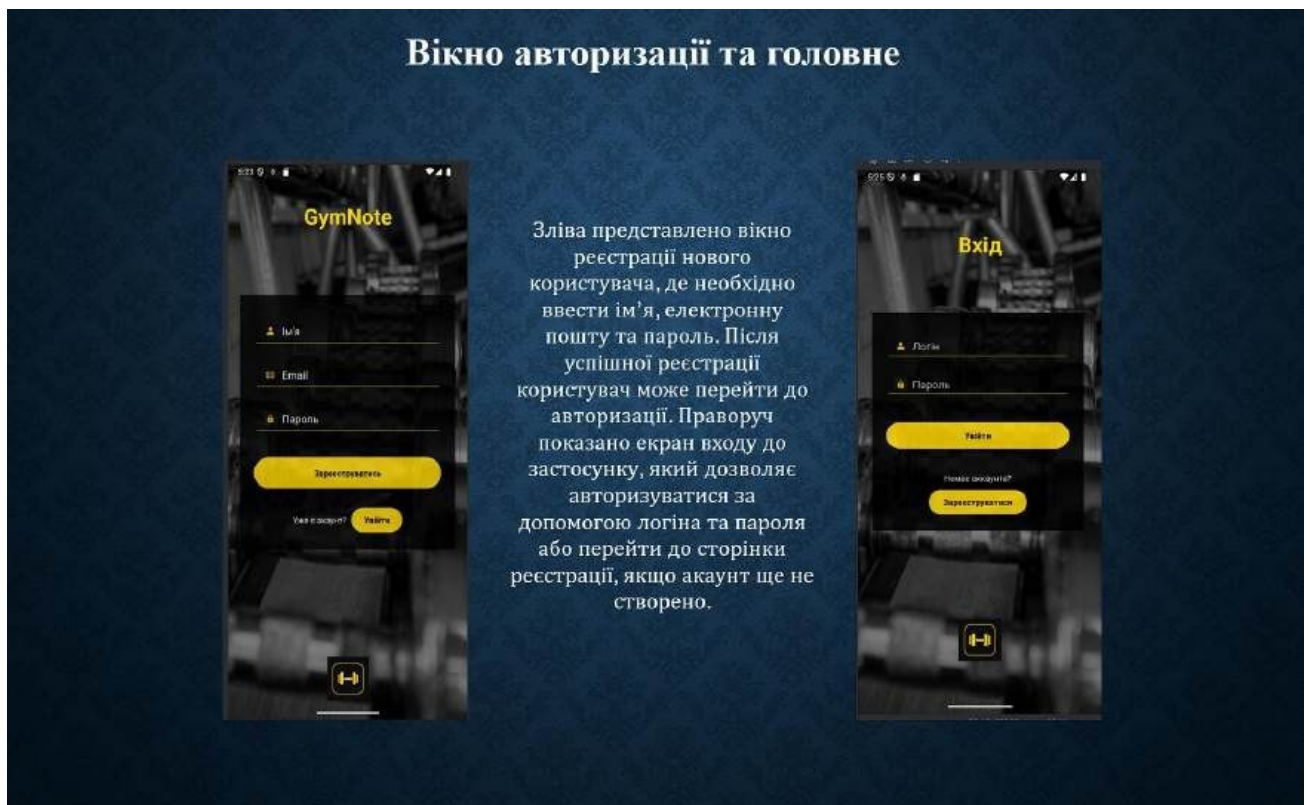


Рисунок Г.10 – Слайд 10

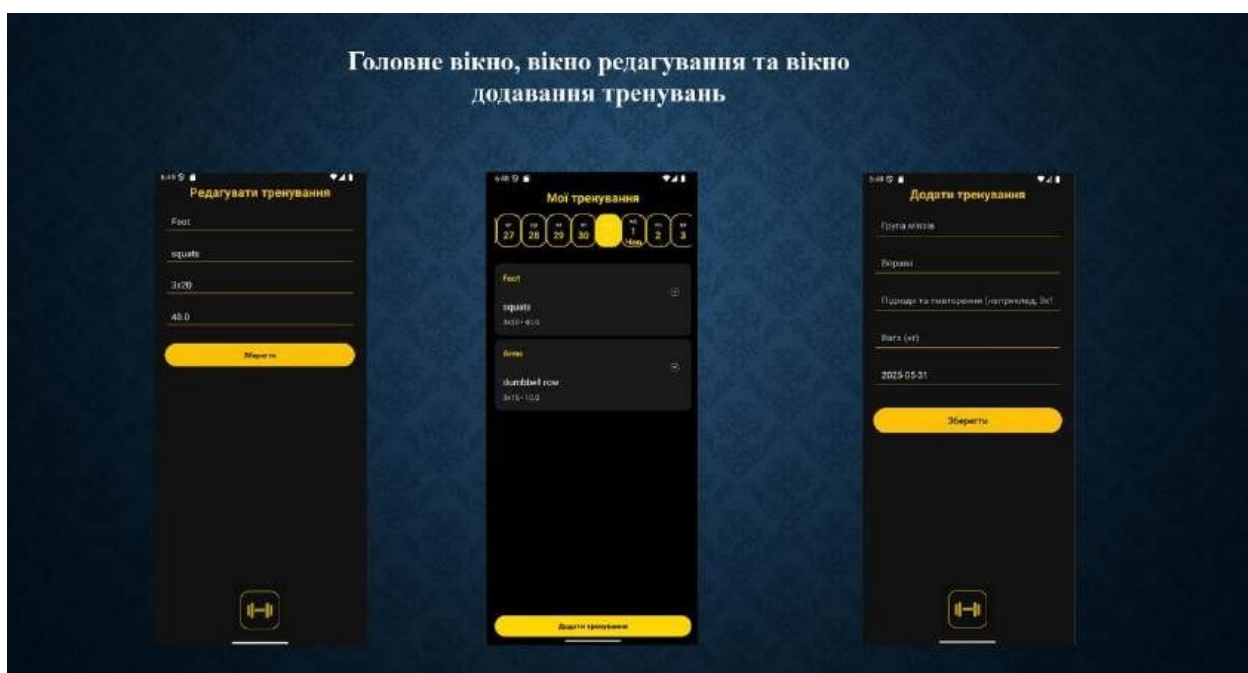


Рисунок Г.11 – Слайд 11



Рисунок Г.12 – Слайд 12