Галицький фаховий коледж імені В’ячеслава Чорновола
відділення комп’ютерних технологій
циклова комісія інформатики та комп’ютерних дисциплін

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач відділенням
комп’ютерних технологій
Наталія СТЕФУРАК /_____/
підпис
«___» _____ 2024 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи
освітньо-професійного ступеня «фаховий молодший бакалавр»
зі спеціальності 122 «Комп’ютерні науки»

на тему: «Мобільний додаток «Особистий тренер» під операційну систему Android»

Студент групи КН-41 Микола ДОЛИК _____
(підпис)

Керівник роботи Оксана СИРОТЮК _____
(підпис)

Консультанти:

з техніко-економічного обґрунтування Любов МЕЛЕНЧУК _____
(підпис)

нормоконтролер Надія ГАВРИШКІВ _____
(підпис)

Тернопіль – 2024

Галицький фаховий коледж імені В’ячеслава Чорновола

відділення комп’ютерних технологій

циклова комісія інформатики та комп’ютерних дисциплін

ЗАТВЕРДЖУЮ

Завідувач відділенням

комп’ютерних технологій

Наталія СТЕФУРАК /_____/

підпис

«___» _____ 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу

на здобуття освітньо-професійного ступеня «фаховий молодший бакалавр»

студенту Долик Микола Степанович

(прізвище, ім’я та по-батькові студента)

1. Тема роботи: Мобільний додаток «Особистий тренер» під операційну систему Android

затверджено наказом по коледжу від “27” листопада 2023 р., № 234а-н

2. Термін здачі студентом завершеного проєкту “___” _____ 2024 р.
3. Вихідні дані до роботи: результати аналізів мобільних додатків для тренувань .
4. Перелік питань, які повинні бути розроблені в роботі:

а) основна частина : аналіз предметної області та постановка завдань, проєктування системи, розробка мобільного додатку.

б) техніко-економічне обґрунтування: аналіз ринку збуту, розрахунок витрат на проєктування, обгрунтування необхідності розробки

5. Перелік графічного матеріалу: контекстна діаграма додатку , декомпозиція контекстної діаграми.
6. Консультанти проєкту: Меленчук Л.І.

Розділ	Консультанти	Підпис, дата	
		Завдання видано	Завдання прийнято
з техніко-економічного обґрунтування	Меленчук Л.І. (вчена ступінь, звання П.І.Б. консультанта)		

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

№ п/п	Найменування етапу	Терміни	
		початку	завершення
1.	Вибір теми кваліфікаційної роботи.	21.11.2023	01.12.2023
2.	Аналіз програмних рішень.	01.04.2024	07.04.2024
3.	Опрацювання теоретичних матеріалів, написання розділу роботи.	08.04.2024	09.04.2024
4.	Формалізація вимог. Аналіз технології реалізації. Написання розділу роботи.	09.04.2024	16.04.2024
5.	Проєктування та реалізація графічного інтерфейсу , реалізація основних функцій додатку.	20.04.2024	10.05.2024
6.	Тестування програмного засобу.	10.05.2024	16.05.2024
7.	Створення відповідного розділу роботи.	16.05.2024	22.05.2024
8.	Обгрунтування вартості роботи.	05.06.2024	10.06.2024
9.	Оформлення пояснювальної записки.	14.06.2024	14.06.2024
10.	Попередній захист кваліфікаційної роботи.	17.06.2024	17.06.2024
11.	Підготовка до захисту та виправлення помилок.	17.06.2024	24.06.2024
12.	Захист кваліфікаційної роботи.	26.06.2024	26.06.2024

5. Дата видачі “___” _____ 2023 р. Керівник _____/

Завдання прийняв до виконання _____/

Реферат

Мобільний додаток «Особистий тренер» під операційну систему Android. Кваліфікаційна робота. Долик Микола Степанович. Галицький фаховий коледж імені В'ячеслава Чорновола, відділення комп'ютерних технологій. Спеціальність 122 «Комп'ютерні науки», 2024. Сторінок – 83, рисунків – 19, додатків – 1.

Об'єкт дослідження – мобільні додатки для тренувань та засоби їхньої реалізації під операційну систему Android.

Метою роботи є створення мобільного додатку «Особистий тренер» під операційну систему Android з використанням Java, Kotlin, XML та SQLite.

Мобільний додаток повинен забезпечити введення та зберігання інформації про тренувальні програми, вправи, користувачів, а також забезпечити можливість відстеження прогресу тренувань в реальному режимі часу. Крім того, необхідно спроєктувати та розробити адміністративну частину для забезпечення можливості управління контентом та повідомленнями від користувачів.

Для реалізації поставленої задачі було використано велику кількість різноманітних технологій та інструментів мобільної розробки, таких як Java, Android Studio та Figma.

Результатом розробки стала завершена робота, яка готова до використання.

Ключові слова: МОБІЛЬНИЙ ДОДАТОК, ОСОБИСТИЙ ТРЕНЕР, ANDROID, JAVA, KOTLIN, XML, SQLITE, ANDROID STUDIO, FIGMA, ТРЕНУВАННЯ.

Abstract

Mobile application "Personal Trainer" for the Android operating system.

Qualification work. Dolyk Mykola Stepanovych. Galician Vocational College named after Vyacheslav Chornovil, Department of Computer Technology. Specialty 122 "Computer Science", 2024. Pages - 56, figures - 32, appendices - 1.

The object of research is mobile applications for training and means of their implementation for the Android operating system.

The aim of the work is to create a mobile application "Personal Trainer" for the Android operating system using Java, Kotlin, XML and SQLite.

The mobile application should provide input and storage of information about training programs, exercises, users, as well as provide the ability to track training progress in real time. In addition, it is necessary to design and develop an administrative part to provide the ability to manage content and messages from users.

To accomplish this task, a large number of different mobile development technologies and tools were used, such as Java, Android Studio, and Figma.

The result of the development was a completed work that is ready for use.

Keywords: MOBILE APPLICATION, PERSONAL TRAINER, ANDROID, JAVA, KOTLIN, XML, SQLITE, ANDROID STUDIO, FIGMA, TRAINING.

Мобільний додаток
«Особистий тренер» під
операційну систему Android

					КР. КН 24.545.5.000 ПЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Долик М.С.				Літ.	Арк.	Аркушів
Перев.		Сиротюк О.Б.						
Рецензет.		Кульчинська Н.З.				ГФКімВЧ.ВКТ.ЦК ІтаКД		
Н. Контр.		Гавришків Н.Г.				Гр. КН-41		
Зав. від.		Стефурак Н.А.						

СКОРОЧЕННЯ І УМОВНІ ПОЗНАКИ

ЄСВ - Єдиний соціальний внесок

ПДВ - Податок на додану вартість

DAO - Data Access Object (об'єкт доступу до даних)

SQLite - Полегшена система управління реляційними базами даних

Android SDK - Android Software Development Kit (набір засобів розробки для Android)

XML - eXtensible Markup Language (розширювана мова розмітки)

API - Application Programming Interface (інтерфейс програмування додатків)

IDE - Integrated Development Environment (інтегроване середовище розробки)

UX - User Experience (досвід користувача)

ВСТУП

У сучасному світі, де здоровий спосіб життя стає все більш популярним, мобільні додатки відіграють важливу роль у допомозі людям досягти своїх фітнес-цілей. "Особистий тренер" - це проект мобільного додатку для операційної системи Android, розроблений з метою надання користувачам зручного та ефективного способу керування своїми тренуваннями. Метою проекту є створення персоналізованого досвіду тренувань для користувачів, допомагаючи їм відстежувати свій прогрес, отримувати рекомендації щодо вправ та планувати майбутні тренування.

Додаток орієнтуватиметься на різні рівні підготовки користувачів, від початківців до досвідчених атлетів, і пропонуватиме широкий вибір вправ та програм тренувань для різних цілей, таких як збільшення м'язової маси, схуднення або загальне покращення фізичної форми. Проект спрямований на створення інтуїтивно зрозумілого та привабливого інтерфейсу користувача, який забезпечить зручний доступ до всіх функцій додатку. Розробка додатку "Особистий тренер" є складним завданням, що вимагає ретельного планування, глибокого розуміння принципів проектування користувацького інтерфейсу, ефективного керування даними та надійної архітектури програмного забезпечення. У роботі буде детально розглянуто процес розробки додатку, від аналізу вимог до реалізації функціональності та тестування.

У вступній частині буде надано короткий огляд поточного стану ринку мобільних фітнес-додатків, обґрунтування необхідності розробки "Особистого тренера" та основні переваги, які він пропонує користувачам. Також будуть визначені цілі та завдання дипломної роботи.

Далі у роботі буде представлено детальний опис процесу розробки додатку, включаючи аналіз вимог, проектування архітектури, реалізацію функціональності, інтеграцію з базою даних та розробку інтерфейсу користувача.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ

1.1 Опис предметної області

У сучасному світі зростає популярність здорового способу життя та активної фізичної активності. Мобільні додатки, спрямовані на підтримку цього стилю життя, стають все більш важливими для користувачів. Для задоволення попиту буде створено мобільний застосунок, призначений для підтримки спортивних заходів та здорового способу життя. Цей додаток надасть зручну та ефективну платформу для користувачів, щоб вони могли отримувати необхідні інструменти та ресурси для досягнення своїх спортивних цілей. Він дозволяє користувачам стежити за своїми тренуваннями, вести облік фізичної активності, отримувати поради щодо харчування та мотиваційні повідомлення. Також, додаток створює спільноту користувачів, де можна обмінюватися досвідом та підтримувати один одного на шляху до здорового способу життя. Завдяки цьому додатку користувачі можуть легко і ефективно керувати своїми тренуваннями та здоров'ям, отримуючи необхідні інструменти та ресурси прямо на своїй мобільній платформі.

1.2 Аналіз наявних рішень

Перш ніж розпочати проектування програмного виробу, варто проаналізувати вже існуючі, або подібні програмні рішення. Аналіз існуючих програмних рішень у сфері мобільних додатків для спорту є важливим етапом у визначенні необхідних функцій та особливостей майбутнього програмного засобу. У цьому розділі буде проводитись огляд та порівняльний аналіз трьох найпопулярніших та одних з кращих існуючих інструментів:

- Персональний тренер BodBot AI.
- Тренування – Muscle Booster.

– Fitness & Bodybuilding.

Кожен з цих інструментів має свої позитивні сторони, а також недоліки або обмеження, які варто розглянути для якісної реалізації, щоб уникнути створення продукту з великою кількістю недоліків заздалегідь.

1.2.1 Персональний тренер BodBot AI

BodBot – це ваш власний цифровий персональний тренер, який надає індивідуальні тренування, пристосовані до ваших цілей, обладнання, фізичних можливостей, бажаного рівня складності та багато іншого. Ось деякі особливості BodBot:

– Адаптація з кожним підходом: Немає стандартних планів. Кожне тренування, вправа, підхід і повторення пристосовані до ваших цілей, вашого тіла та вашого прогресу.

– Візуалізація даних. Отримуйте глибокі інсайти щодо своїх цілей за допомогою потужних інструментів відстеження та візуалізації даних. Дізнайтеся, як BodBot допомагає покращити вашу загальну фізичну підготовку та як ваш спосіб життя впливає на результати.

– Інтелектуальне планування тренувань. Незалежно від графіку користувача, BodBot може вписати тренування в будь-яку комбінацію. Він легко впорається з різними обмеженнями часу.

– Контроль над суглобами. Працюйте над своєю фізичною формою, уникайте навантаження на певні суглоби та тренуйтеся, навіть якщо у вас є травми або обмеження.

Завдяки інтелектуальним рекомендаціям BodBot працює з вами, щоб покращити продуктивність та ефективність ваших тренувань. Це сучасна фізіологія та кінезіологія, реалізована в рекомендаціях з розумного тренування. Ви можете знайти додаток BodBot на Google Play або App Store

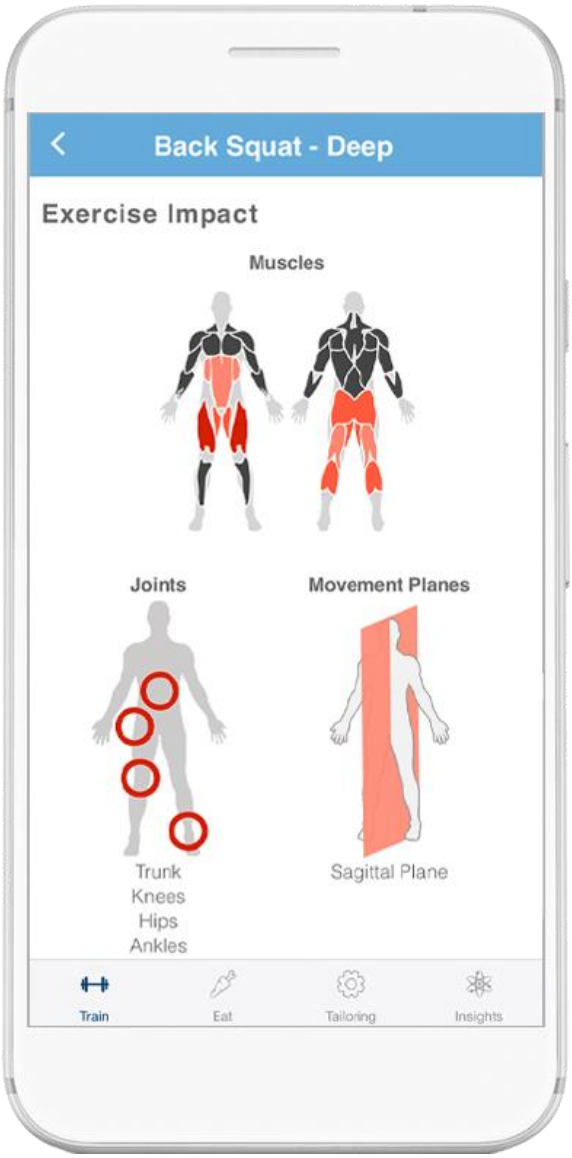

Рисунок 1.1 – Частина інтерфейсу BodBot

1.2.2 Muscle Booster

Muscle Booster – це додаток для тренувань, який пропонує індивідуальні програми для набору м’язової маси або схуднення. Ось деякі ключові особливості Muscle Booster:

– Гімнастичні тренування. Можна вибрати тренування в спортзалі або вдома. Додаток створить для вас індивідуальний план, враховуючи ваші цілі та обладнання.

– Набір м’язової маси. Muscle Booster містить рутини для всіх груп м’язів та рівнів фітнесу, від початківців до досвідчених спортсменів.

– Анімаційні відео. Додаток надає відео-інструкції для кожного вправи.

Muscle Booster пропонує платну підписку, але ви можете завантажити його безкоштовно та використовувати обмежені функції.

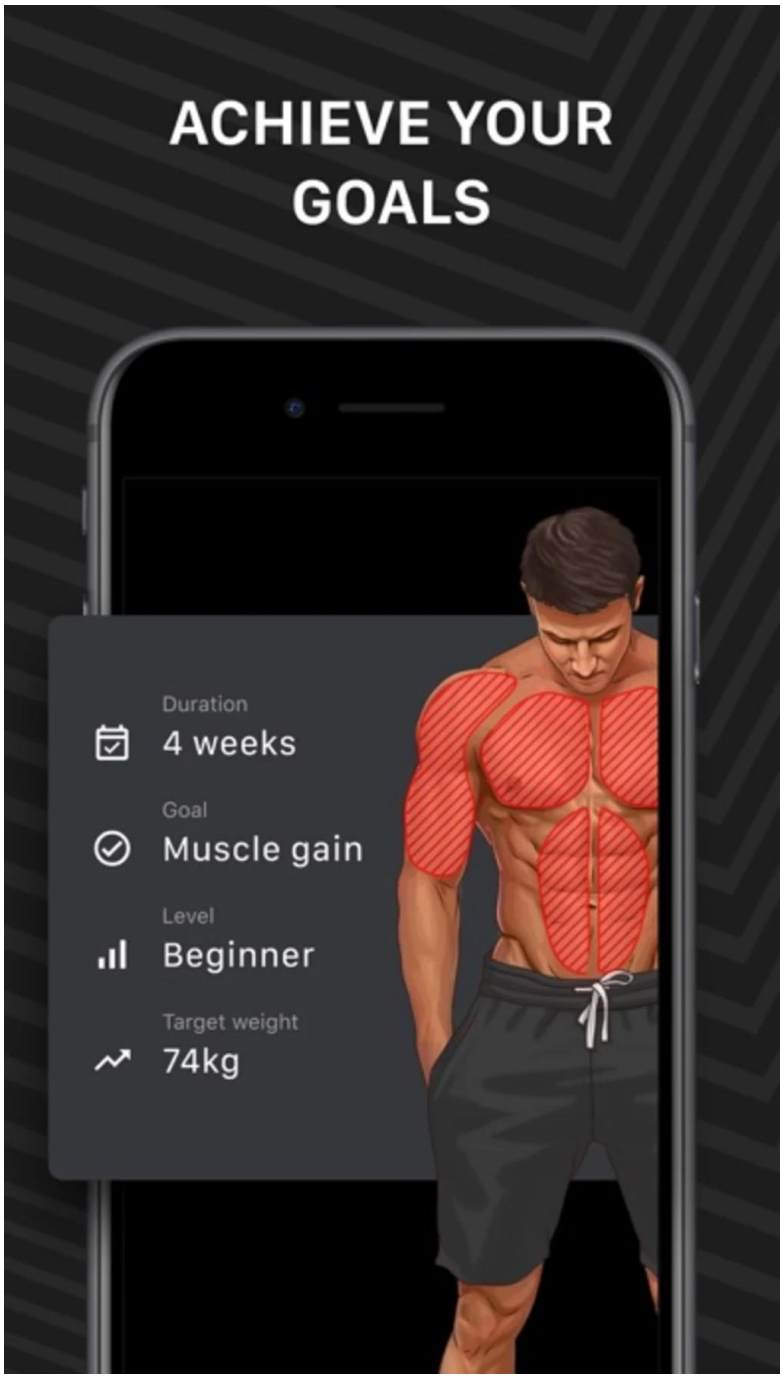

Рисунок 1.2 – Додаток Muscle Booster

1.2.3 Fitness & Bodybuilding

Fitness & Bodybuilding - це інноваційний і потужний фітнес-додаток, який надає попередньо встановлені плани тренувань для бодібілдингу, силових тренувань, м'язового тонусу, загальної фізичної підготовки та пауерліфтингу.

Цей додаток допоможе вам розвивати силу кора, знижувати надлишкову вагу та покращувати вашу фізичну витривалість. Його дружній інтерфейс дозволяє кожному використовувати вправи для тренування найважливіших м'язових груп. Ви зможете відстежувати свої досягнення та аналізувати результати тренувань. Це, як мати найкращого персонального тренера в кишені, доступного 24/7/365.

Основні можливості додатка Fitness & Bodybuilding:

– Ефективні вправи для кожної м'язової групи.
– Детальний опис кожної вправи.
– Високоякісне фото та відеонавчання.
– Зображення задіяних м'язів.
– Попередньо встановлені тренування, специфічні для ваших цілей.
– Можливість додавати власні тренування зі своїми власними вправами та фотографіями.
– Журнал для ваших тренувань, де ви можете вказувати підходи, повторення та вагу.
– Вбудований таймер та календар.
– Історія всіх ваших даних з інтерактивними графіками продуктивності.
– Швидка підтримка та часті оновлення.
– Тренувальні плани, створені провідними персональними тренерами.

За допомогою Fitness & Bodybuilding ви зможете вести активний спосіб життя, розвивати здорові звички та відчувати себе чудово. Додаток допоможе зробити ваші тренування в спортзалі або вдома цікавими та ефективними, поєднуючи різноманітні вправи та використовуючи наявне обладнання.

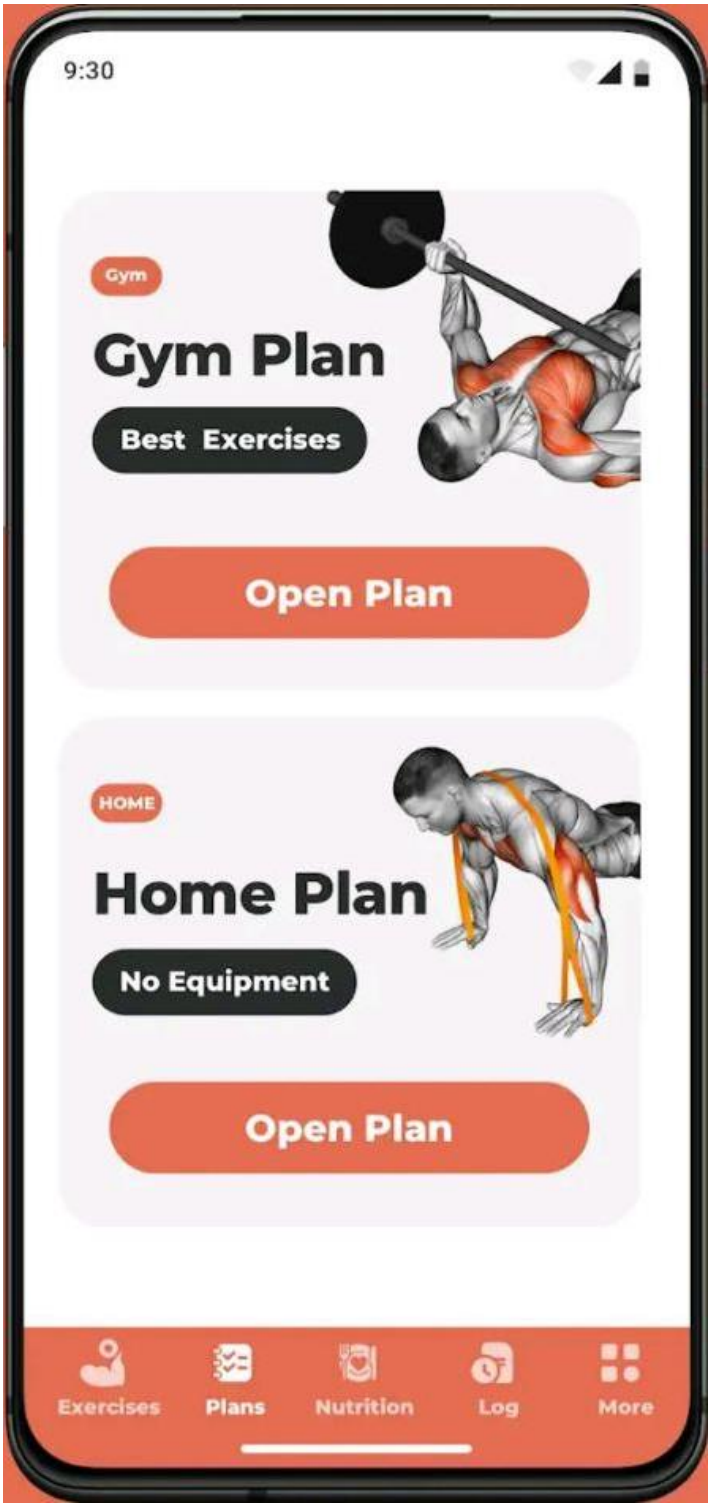

Рисунок 1.3 – Частина інтерфейсу Fitness & Bodybuilding

1.2.4 Порівняльний аналіз схожих продуктів

Для того, щоб провести всебічне порівняння існуючих додатків, необхідно розглянути низку ключових аспектів, що мають вплив на користувацький досвід та функціональність програмних рішень. Серед основних критеріїв оцінки варто виділити: зручність взаємодії з інтерфейсом, можливості автоматизації процесів, наявність фільтрів для відбору даних, спектр підтримуваних протоколів, простоту використання для кінцевого споживача, вимоги до обчислювальних ресурсів системи, доступність спеціалізованих інструментів та сумісність з різними платформами. Розглянемо детальніше кожен з цих аспектів:

– Інтерфейс користувача. Оцінюється зручність, інтуїтивність та естетичність графічного інтерфейсу, а також наявність альтернативних способів взаємодії (наприклад, голосове керування).

– Автоматизація. Визначає здатність програми автоматизувати певні завдання та процеси, зменшуючи необхідність ручного втручання користувача. Фільтрація даних: Наявність можливості застосовувати фільтри для відбору релевантної інформації за певними критеріями.

– Підтримка протоколів. Оцінюється спектр мережевих та інших протоколів, які програма здатна розпізнавати та обробляти.

– Зручність використання. Визначає простоту освоєння програми та її інтуїтивність для кінцевого користувача, незалежно від його рівня технічної підготовки.

– Споживання ресурсів. Оцінюється ефективність використання обчислювальних ресурсів системи, таких як оперативна пам'ять, процесорний час та дисковий простір.

– Спеціалізовані функції. Наявність унікальних інструментів та можливостей, що відрізняють програму від інших рішень на ринку.

– Кросплатформеність. Здатність програми працювати на різних операційних системах та апаратних платформах без втрати функціональності.

Провівши детальний аналіз за цими критеріями, можна об'єктивно оцінити сильні та слабкі сторони кожного програмного рішення, що дозволить визначити напрямки для вдосконалення та диференціації нового продукту на ринку(таб. 1.1).

Таблиця 1.1 – Порівняльна характеристика додатків

Критерії	BodBot AI	Muscle Booster	Fitness & Bodybuilding
Інтерфейс	Графічний інтуїтивний інтерфейс	Простий графічний інтерфейс	Графічний інтерфейс з анімаціями
Персоналізація	Високий рівень персоналізації тренувань	Персоналізовані плани залежно від цілей	Попередньо встановлені плани тренувань
Відстеження прогресу	Детальне відстеження з візуалізацією даних	Базове відстеження ваги та вправ	Деталізований журнал тренувань з графіками
Харчування	Не передбачено	Не передбачено	Не передбачено
Спільнота	Не передбачено	Не передбачено	Не передбачено
Мотивація	Система досягнень	Не передбачено	Не передбачено
Персональний тренер	ТАК, з елементами ШІ	НІ	НІ
Кросплатформеність	Android, iOS	Android, iOS	Android

Ця порівняльна таблиця дає можливість отримати об'єктивний огляд їхніх переваг і обмежень.

Щодо інтерфейсу, BodBot AI відрізняється інтуїтивним графічним інтерфейсом, тоді як Muscle Booster та Fitness & Bodybuilding опираються на простіші графічні інтерфейси.

Здатність до персоналізації виявилася високою у BodBot AI завдяки адаптивним тренуванням, в той час як Muscle Booster та Fitness & Bodybuilding пропонують більш обмежену персоналізацію.

Щодо відстеження прогресу, BodBot AI вирізняється детальним відстеженням з візуалізацією даних, тоді як у Muscle Booster воно базове, а Fitness & Bodybuilding має докладний журнал з графіками.

У підтримці харчування та спільноти жоден з додатків не пропонує відповідних функцій.

Зручність використання для користувача у BodBot AI висока завдяки інтуїтивному інтерфейсу, хоча у Muscle Booster та Fitness & Bodybuilding вона нижча через простіші інтерфейси.

Споживання ресурсів для всіх трьох додатків, ймовірно, приблизно однакове.

Наявність спеціалізованих функцій, таких як система досягнень, персональний тренер з ШІ та анімовані інструкції, є перевагою BodBot AI та Fitness & Bodybuilding відповідно.

Кросплатформеність забезпечується у BodBot AI та Muscle Booster для Android та iOS, тоді як Fitness & Bodybuilding підтримує лише Android.

Таким чином, порівняння виявляє сильні та слабкі сторони кожного додатку, що допомагає визначити можливості для подальшого вдосконалення та диференціації майбутнього програмного рішення.

1.3 Аналіз вимог до програмного засобу та постановка завдання

На основі аналізу існуючих рішень були визначені функціональні та нефункціональні вимоги до майбутнього мобільного додатку для фітнесу та ведення здорового способу життя.

Функціональні вимоги:

– Реєстрація та авторизація користувачів.
– Створення та налаштування облікових записів користувачів.
– Можливість створення власних програм тренувань.
– Відстеження прогресу, ваги, вимірів тіла та активності.
– Система досягнень, нагород та мотивації.

Нефункціональні вимоги:

– Зручний та інтуїтивно зрозумілий користувацький інтерфейс.
– Висока продуктивність та оптимізація споживання ресурсів.
– Сумісність з різними операційними системами.
– Можливість оновлення та розширення функціоналу.
– Підтримка та безпека користувачів.

Основними завданнями для розробки є реалізація додатку для полегшення тренувань, відстеження прогресу, харчування та мотивації користувачів, а також створення зручного інтерфейсу та модулів візуалізації даних. Важливим аспектом є тестування, валідація та забезпечення безпеки додатку.

РОЗДІЛ 2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Формування цілей і основних завдань мобільного додатку

Розробка мобільного додатку "Особистий тренер" для операційної системи Android охоплює низку етапів, що визначають його успіх у кінцевому результаті.

Першим етапом є аналіз вимог та планування. На цьому етапі необхідно визначити основні цілі додатку, його цільову аудиторію та необхідний функціонал. Правильне формулювання вимог на початковому етапі дозволить уникнути змін у процесі розробки, які можуть призвести до додаткових витрат часу та ресурсів.

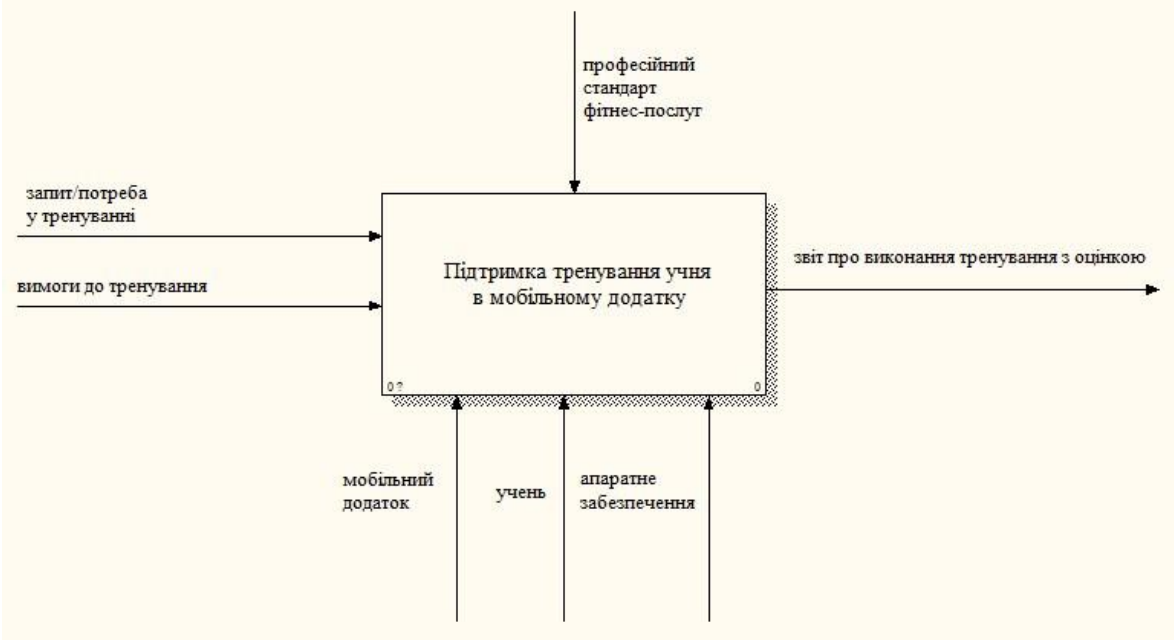

Рисунок 2.1 – Контекстна діаграма додатку

На рисунку 2.1 зображено процес надання тренувань за допомогою мобільного додатку. На вхід подаються запит на тренування та вимоги до нього. Згідно професійних стандартів фітнес-послуг формується програма тренувань з урахуванням мобільного додатку, апаратного забезпечення та потреб самого

користувача. Після виконання тренувань формується звіт з оцінкою для аналізу та коригування подальшого процесу.

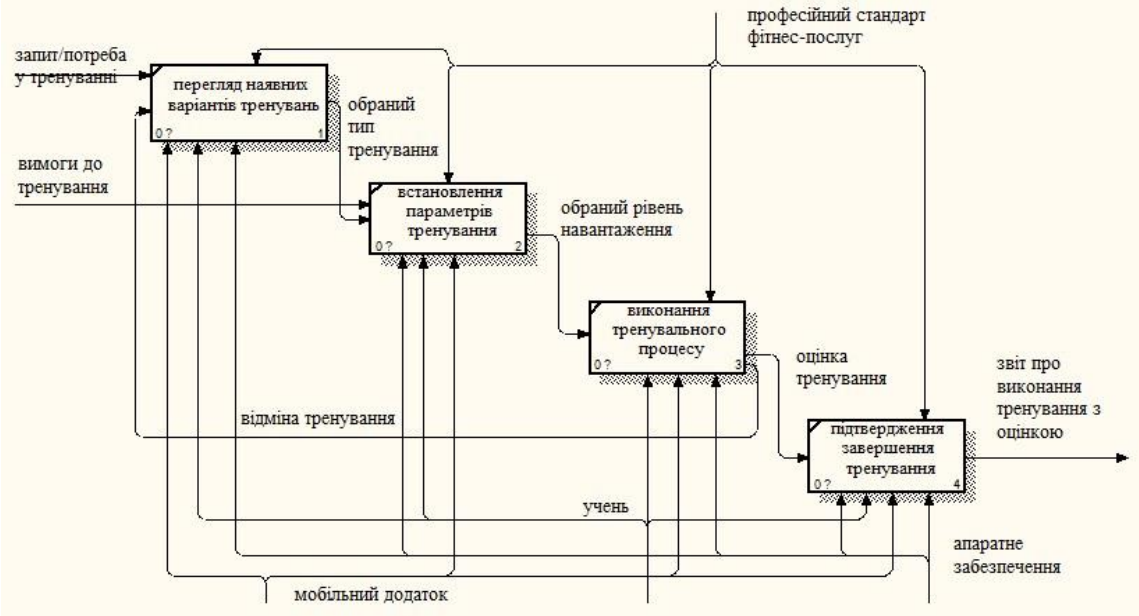

Рисунок 2.2 – Декомпозиція контекстної діаграми

Наступним є етап проєктування, який визначає структуру, дизайн та технології, що будуть використовуватися для реалізації додатку.

Після цього настає етап розробки, що включає написання коду, реалізацію функціональності, створення користувацького інтерфейсу, розробку серверної частини для обробки запитів користувачів. Цей крок вимагає ретельної уваги до деталей та високого рівня відповідальності, щоб забезпечити відповідність розробленого додатку всім вимогам та належне функціонування.

Передостаннім етапом є тестування та оптимізація додатку для виявлення та виправлення помилок, перевірки відповідності вимогам та усунення можливих недоліків до його публікації в магазині додатків.

Завершальним етапом є розміщення додатку в Google Play Store, що зробить його доступним для користувачів Android.

2.2 Функціональна структура мобільного додатку

Функціональна організація додатку визначає основні аспекти розробки, взаємозв'язки між різними компонентами та можливості для подальшого розширення.

Незареєстрований користувач має можливість ознайомитись з функціоналом додатку та зареєструватись. Після реєстрації користувачеві відкриваються наступні можливості:

– Налаштування облікового запису: редагування особистих даних, уподобань, цілей.

– Програми тренувань: доступ до бібліотеки готових програм, створення власних.

– Відстеження прогресу та активності: контроль ваги, вимірів тіла, фізичної активності.

– Система мотивації та досягнень: нагороди, виклики, змагання.

2.3 Проєктування інтерфейсу мобільного додатку

Інтерфейс користувача (UI) відіграє важливу роль у забезпеченні зручної та ефективної взаємодії з мобільним додатком. Для додатку "Особистий тренер" пропонується інтуїтивний та сучасний дизайн з використанням матеріального стилю Android. У фреймворку Figma для розробки користувацького інтерфейсу створюється макет майбутнього додатку. Спочатку визначаються основні розділи та функціональні елементи, такі як головне меню, особистий кабінет користувача, розділ з програмами тренувань, база вправ із відео-інструкціями тощо. Для кожного елементу продумується розміщення, стиль оформлення, кольорова гама відповідно до фірмового стилю додатку. Визначаються зв'язки та переходи між різними розділами програми. Кожен екран детально прописується з розміщенням всіх потрібних елементів керування, навігаційних кнопок, віджетів. Особлива увага приділяється

зручності та інтуїтивності інтерфейсу для користувача. Після створення всіх необхідних екранів відпрацьовуються взаємозв'язки між ними та прописуються правила переходів, обробки подій натискань, введення даних тощо. У підсумку отримується повноцінний інтерактивний макет майбутнього додатку, готовий для передачі розробникам для подальшого програмного втілення. Орієнтовні макети основних екранів додатку зображені на рисунках 2.3.2-2.3.4.

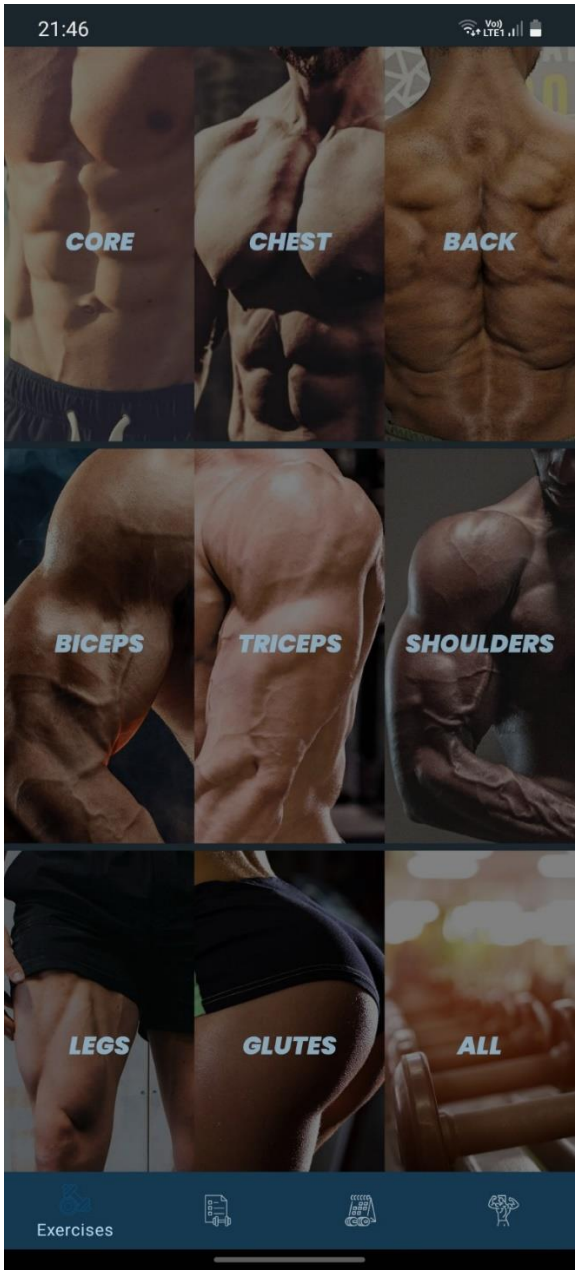

Рисунок 2.3 –Домашня сторінка додатку

Цей рисунок містить детальні візуальні підказки щодо різних м'язових груп, що свідчить про наявність у додатку функціоналу, орієнтованого на вправи та тренування окремих частин тіла. Це може бути корисним для користувачів, які прагнуть цілеспрямовано розвивати конкретні м'язи.

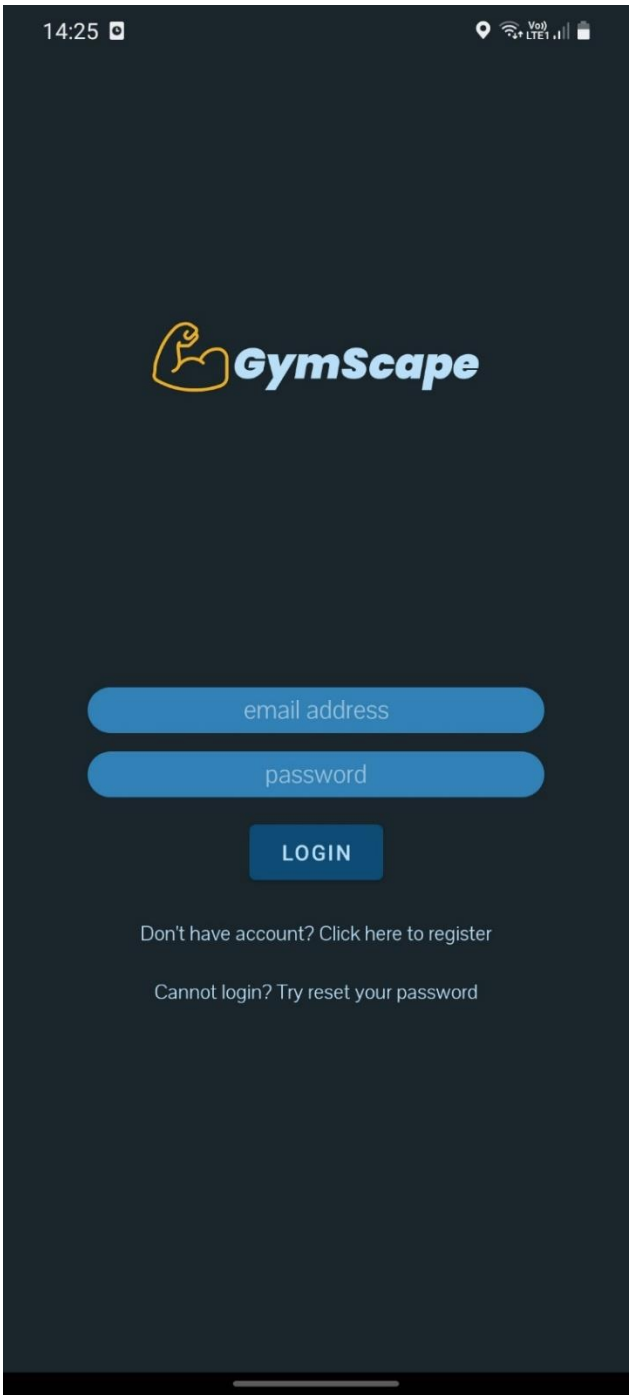

Рисунок 2.4 – Сторінка аутентифікації

Цей рисунок демонструє стандартний екран реєстрації в додаток, з полями для електронної пошти та пароля, а також опціями для реєстрації та відновлення паролю. Це типовий інтерфейс для аутентифікації користувачів.

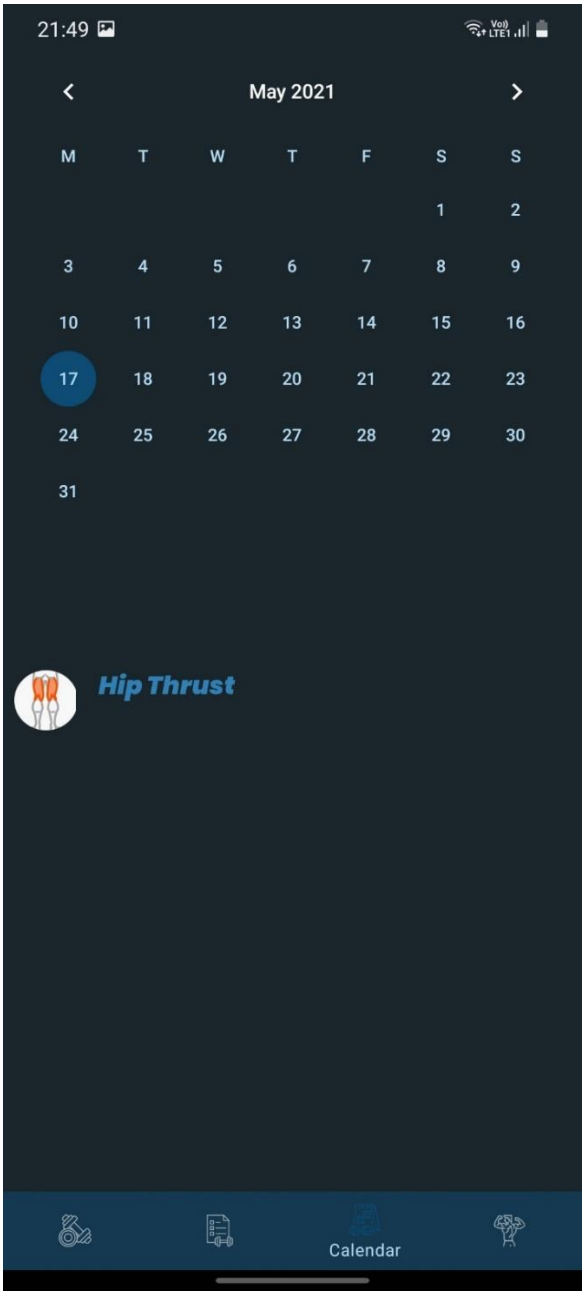

Рисунок 2.5 – Календар

Цей рисунок демонструє календар, який, дозволяє планувати та відстежувати тренування.

Інтерфейс додатку передбачає використання різних кольорів, простих іконок та елементів для забезпечення привабливого та сучасного вигляду.

Навігація між різними розділами буде інтуїтивною та зручною за допомогою нижньої панелі навігації. Загалом, інтерфейс виглядає сучасним та інтуїтивно зрозумілим, з акцентом на організацію тренувань, візуалізацію анатомії та допомогу користувачам у досягненні їхніх фітнес-цілей.

2.4 Проєктування бази даних мобільного додатку

База даних відіграє ключову роль у зберіганні та організації даних, необхідних для належного функціонування мобільного додатку "Особистий тренер". Для розробки буде використана система керування базами даних SQLite, яка є легковаговою та кросплатформеною, що ідеально підходить для мобільних додатків.

Концептуальна модель бази даних включає такі основні сутності:

– Користувач зберігає інформацію про зареєстрованих користувачів, їхні особисті дані, уподобання та налаштування.

– Тренування містить інформацію про програми тренувань, вправи, інтенсивність, тривалість тощо.

– Вправа зберігає деталі про різні види вправ, задіяні групи м'язів, інструкції з виконання.

– Прогрес записує дані про прогрес користувача, такі як вага, виміри тіла, пройдені тренування.

– Активність фіксує інформацію про фізичну активність користувача, крім запланованих тренувань.

– Досягнення відстежує досягнення користувачів, такі як виконані виклики, завдання тощо.

Така структура бази даних дозволить ефективно зберігати та отримувати доступ до даних, необхідних для основних функцій додатку, таких як керування тренуваннями, відстеження прогресу, харчування, досягнень та соціальної взаємодії.

Нижче наведено аналіз інформаційних об'єктів та їх атрибутів сутності, а також аналіз та опис зв'язків між сутностями(таб 2.1, таб 2.2).

Таблиця 2.1 - Аналіз інформаційних об'єктів та їх атрибутів сутності

Інформаційний об'єкт	Атрибут сутності	Ідентифікатор	Ключ
User	id	id	P-key
	name	name	-
	email	email	-
	password	password	-
	height	height	-
	weight	weight	-
	goals	goals	-
	preferences	preferences	-
Workout	id	id	P-key
	name	name	-
	description	description	-
	duration	duration	-
	difficulty	difficulty	-
	equipment	equipment	-
	user_id	user_id	F-key

Продовження таблиці 2.1

Exercise	id	id	P-key
	name	name	-
	description	description	-
	muscle_groups	muscle_groups	-
	instructions	instructions	-
Progress	id	id	P-key
	user_id	user_id	F-key
	weight	weight	-
	body_measurements	body_measurements	-
	date	date	-
Activity	id	id	P-key
	user_id	user_id	F-key
	type	type	-
	duration	duration	-
	date	date	-
Achievement	id	id	P-key
	user_id	user_id	F-key
	name	name	-
	description	description	-
	date_achieved	date_achieved	-

Таблиця 2.2 - Аналіз та опис зв'язків між сутностями

Зв'язки між сутностями	Назва зв'язку	Тип зв'язку	Зміст зв'язку
User - Workout	"Має"	1:M	Один користувач може мати багато тренувань, але кожне тренування належить лише одному користувачу
User - Progress	"Має"	1:M	Один користувач може мати багато записів прогресу, але кожен запис прогресу належить лише одному користувачу
User - Activity	"Має"	1:M	Один користувач може мати багато записів , але запис кожен належить одному користувачу
Workout - Exercise	"Містить"	M:N	Одне тренування може містити багато вправ, і одна вправа є частиною багатьох тренувань

Ця структура бази даних дозволить ефективно зберігати та отримувати доступ до даних, необхідних для основних функцій додатку, таких як керування тренуваннями, відстеження прогресу, активності, досягнень .

РОЗДІЛ 3 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

3.1 Розробка інтерфейсу додатку

Однією з найважливіших частин розробки мобільного додатку "Особистий тренер" є створення зручного та інтуїтивно зрозумілого користувацького інтерфейсу. З метою забезпечення максимального комфорту користувачів при роботі з додатком, зокрема для покращення ефективності тренувальних вправ, інтерфейс повинен бути чітко й ергономічно спроектованим.

Процес розробки користувацького інтерфейсу для додатку "Особистий тренер" складався з кількох етапів. На початкових стадіях проектування інтерфейсу була обрана концепція, що відповідає специфіці фітнес-вправ. З огляду на особливості тренувань з обтяженнями, пов'язаних з підвищеним фізичним навантаженням, дизайн додатку повинен бути простим та зрозумілим, а весь функціонал повинен бути досяжним за мінімум кроків.

Для розробки інтерфейсу були застосовані підходи та принципи, властиві фітнес-додаткам. Простота та природність рухів були ключовими вимогами до розробленого інтерфейсу. Інструментарій взаємодії з додатком був спроектований таким чином, щоб забезпечити зручність переходу між функціями, керування процесами та подолання незручностей, пов'язаних з фізичними навантаженнями під час тренувань.

При проектуванні користувацького інтерфейсу мобільного додатку "Особистий тренер" автор активно використовував Figma - потужний та багатофункціональний інструмент для розробки дизайну різноманітних додатків. Завдяки Figma була успішно реалізована можливість створення зручного інтерфейсу для відстежування фізичного прогресу та покращення фізичних показників під час тренувань.

Figma дозволила ефективно розпланувати розміщення та взаємодію елементів інтерфейсу, зокрема кнопок керування, вікон для відображення

інформації тощо. Цей інструмент був використаний для візуалізації алгоритму взаємодії користувачів з додатком. Зокрема, автор мав можливість відповідним

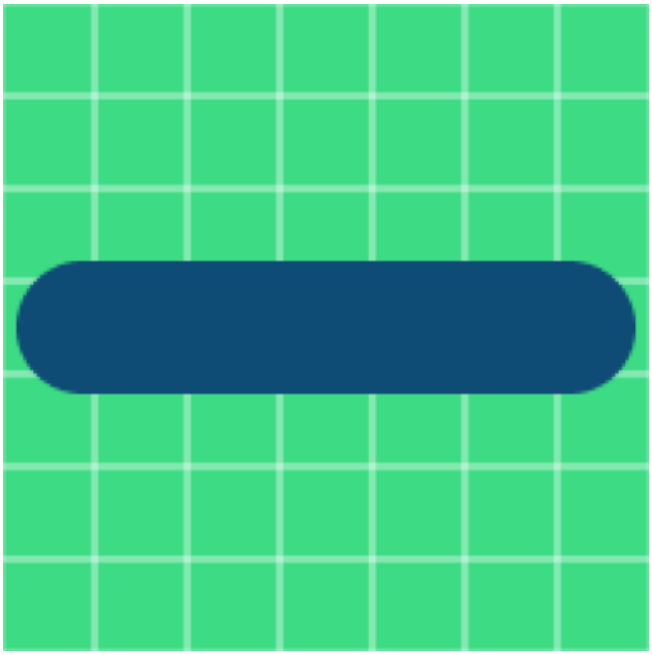

Рисунок 3.1 - Кнопка

чином налаштувати інтерактивність кнопок (рис. 3.1) та інших елементів керування для забезпечення зручності використання продукту на пристроях з різними характеристиками. Загалом, Figma надала широкі можливості для створення ефективного та інтуїтивно зрозумілого інтерфейсу мобільного додатку для керування досвідом користувачів. Далі наведені зображення інтерфейсу додатку.

Рисунок 4.2 – Логотип додатку

Це зображення (рис. 4.3) демонструє інтерфейс додатку Figma, який використовується для проектування та прототипування дизайну користувацького інтерфейсу.

На лівій панелі є список сторінок та елементів, які включають екрани логіну, додавання вправ, огляд вправ, налаштування та головний екран. У центральній частині показано макети різних екранів додатку, таких як профіль користувача, список вправ, деталі вправ та екран додавання вправ до тренування.

Макети мають темний колір у стилі Material Design з акцентами блакитного кольору. Вони містять різні елементи інтерфейсу, такі як кнопки, списки, поля вводу, зображення та піктограми.

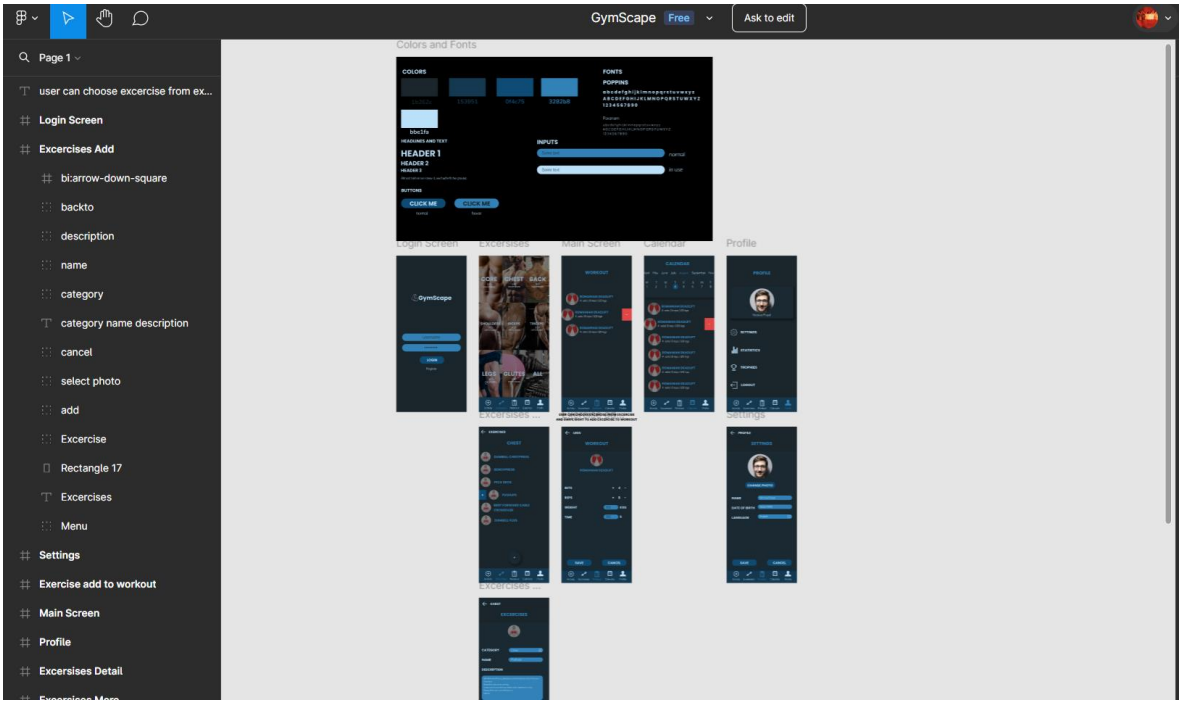

Рисунок 4.3 – Розробка інтерфейсу сторінок

3.2 Розробка функціоналу додатку

Далі код, який представляє клас Java `Exercise`, який є моделлю даних для сутності "вправа" у додатку GymScape. Клас розроблений для використання з Android Room, який є бібліотекою для керування локальною базою даних SQLite у додатках Android.

Цей код є класом Java, що представляє сутність "Exercise" (вправа) для програми або додатку, пов'язаного з тренуваннями в тренажерному залі. Розглянемо його по частинах:

Імпорти: `javaCopyimport androidx.room.Entity; import androidx.room.Ignore; import androidx.room.PrimaryKey; import java.io.Serializable;`

Ці імпорти вказують, що клас використовує анотації (@Entity, @Ignore, @PrimaryKey) з бібліотеки Room, яка є абстракцією над SQLite для Android, а також реалізує інтерфейс Serializable для можливості серіалізації об'єктів цього класу. Анотація @Entity:

```
javaCopy@Entity(tableName = "exercise_table")
```

Ця анотація вказує, що клас Exercise відображається на таблицю SQLite з назвою "exercise_table". Поля класу:

```
javaCopy@PrimaryKey(autoGenerate = true) private int id; private String name; private int category; private String description; private String picture; private boolean isDatabase;
```

Тут визначено поля (властивості) класу Exercise: id (первинний ключ, автоматично генерується) name (назва вправи) category (категорія вправи, цілочисельне значення) description (опис вправи) picture (посилання на зображення вправи) isDatabase (булеве значення, що вказує, чи вправа зберігається в базі даних) Конструктори:

Методи доступу (getters і setters): javaCopy// Getters public int getId() { ... } public String getName() { ... } // ... і так далі // Setters public void setId(int id) { ... } public void setName(String name) { ... } // ... і так далі Ці методи забезпечують доступ до полів класу для читання (getters) та запису (setters). Таким чином, цей

код визначає клас Exercise, що представляє сутність вправи в додатку, пов'язаному з тренуваннями в тренажерному залі. Він містить відповідні поля для зберігання інформації про вправи, конструктори для їх ініціалізації та методи доступу для роботи з даними.

Цей клас є частиною моделі даних додатку GymScape і буде використовуватися для зберігання та керування даними про вправи в локальній базі даних Room.

Уривок коду для редагування та оновлення інформації про тренування (Workout).

```
private EditWorkoutViewModel viewModel; Workout workout;
ImageView exerciseImageCategory; TextView exerciseName; EditText
editTextDate; TextView setsCountTextView; EditText
weightTextNumber; Button updateWorkoutButton; Button
deleteWorkoutButton; Button cancelWorkoutButton; Button
increaseSets; Button decreaseSets; Calendar calendar;
```

У цій частині коду оголошуються та ініціалізуються наступні змінні: viewModel: Екземпляр класу EditWorkoutViewModel, який є ViewModel для цієї діяльності. ViewModel використовується для відокремлення логіки від інтерфейсу користувача. workout: Об'єкт класу Workout, який представляє тренування, що редагується. exerciseImageCategory, exerciseName, editTextDate, setsCountTextView, weightTextNumber, updateWorkoutButton, deleteWorkoutButton, cancelWorkoutButton, increaseSets, decreaseSets: Ці змінні посилаються на елементи інтерфейсу користувача (ImageView, TextView, EditText, Button) для відображення та редагування інформації про тренування. calendar: Об'єкт класу Calendar, який допомагає працювати з датами.

```
onCreate javaCopy@Override protected void onCreate(Bundle
savedInstanceState) { super.onCreate(savedInstanceState);
setContentView(R.layout.activity_edit_workout); viewModel = new
ViewModelProvider(this,
ViewModelProvider.AndroidViewModelFactory.getInstance(getApplicat
```

```
ion())).get(EditWorkoutViewModel.class); Intent intent =
getIntent(); workout = (Workout)
intent.getSerializableExtra(UsedEnums.WORKOUT.toString());
exerciseImageCategory = findViewById(R.id.exerciseImageCategory);
exerciseName = findViewById(R.id.exerciseName); editTextDate =
findViewById(R.id.editTextDate); setsCountTextView =
findViewById(R.id.setsCountTextView); weightTextNumber =
findViewById(R.id.weightTextNumber); updateWorkoutButton =
findViewById(R.id.updateWorkoutButton); deleteWorkoutButton =
findViewById(R.id.deleteWorkoutButton); cancelWorkoutButton =
findViewById(R.id.cancelWorkoutButton); increaseSets =
findViewById(R.id.increaseSetsButton); decreaseSets =
findViewById(R.id.decreaseSetsButton); }
```

Ця частина коду є методом onCreate, який викликається під час створення діяльності.

ViewModel відповідає за керування даними, пов'язаними з редагуванням тренування, такими як кількість підходів та дата. Він також забезпечує методи для взаємодії з базою даних через репозиторій для видалення та оновлення записів тренувань. Activity може спостерігати за змінами даних у ViewModel за допомогою LiveData і відповідно оновлювати інтерфейс користувача.

```
package com.example.gymscape.ui.editworkout;

import android.app.Application;

import androidx.annotation.NonNull;

import androidx.lifecycle.AndroidViewModel;

import androidx.lifecycle.LiveData;

import androidx.lifecycle.MutableLiveData;

import com.example.gymscape.Model.Workout;

import
com.example.gymscape.architecture.shared.ExerciseRepository;

import java.util.Date;
```

```
public class EditWorkoutViewModel extends AndroidViewModel {
    ExerciseRepository repository;
    MutableLiveData<Integer> setsCount;
    MutableLiveData<Date> date;
    public EditWorkoutViewModel(Application application) {
        super(application);
        repository =
ExerciseRepository.getInstance(application);
        setsCount = new MutableLiveData<>();
        date = new MutableLiveData<>();
    }
    public LiveData<Integer> getSetsCount() {
        return setsCount;
    }
    public void setSetsCount(int setsCount) {
        this.setsCount.setValue(setsCount);
    }
    public LiveData<Date> getDate() {
        return date;
    }
    public void setDate(Date date) {
        this.date.setValue(date);
    }
    public void delete(Workout workout)
    {
```

```
            repository.deleteWorkout(workout);
        }
        public void update(Workout workout)
        {
            repository.updateWorkout(workout);
        }
    }
```

delete(Workout): Метод, який викликає метод deleteWorkout у репозиторії для видалення запису тренування з бази даних.

update(Workout): Метод, який викликає метод updateWorkout у репозиторії для оновлення запису тренування в базі даних.

Цей код представляє клас `ExercisesFragment`, який є фрагментом у додатку GymScape. Фрагмент - це модульний компонент інтерфейсу користувача, який можна повторно використовувати в різних Activity.

```
package com.example.gymscape.ui.main.exercises;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;
import com.example.gymscape.R;
public class ExercisesFragment extends Fragment {
    private ExercisesViewModel exercisesViewModel;
```

```
        public View onCreateView(@NonNull LayoutInflater
inflater,
                ViewGroup container, Bundle savedInstanceState)
{
            exercisesViewModel =
                    new
ViewModelProvider(this).get(ExercisesViewModel.class);
            View root =
inflater.inflate(R.layout.fragment_exercises, container, false);
            /*final TextView textView =
root.findViewById(R.id.text_exercises);
exercisesViewModel.getText().observe(getViewLifecycleOwner(), new
Observer<String>() {
                @Override
                public void onChanged(@Nullable String s) {
                    textView.setText(s);
                }
            });*/
            return root;
        }
    }
```

Хоча в коді немає багато функціональності, він демонструє базову структуру фрагменту та його зв'язок з ViewModel. У додатку цей фрагмент використаний для відображення списку вправ, деталей окремих вправ або інших функцій, пов'язаних з вправами.

3.3 Розробка бази даних

Клас ExerciseDatabase є основою для роботи з базою даних Room у додатку GymScape. Він визначає сутності (таблиці) бази даних та забезпечує доступ до відповідних DAO для взаємодії з цими сутностями. Екземпляр ExerciseDatabase можна отримати в інших частинах додатку за допомогою методу getInstance(Context), який гарантує, що буде повернутий єдиний екземпляр бази даних.

Використовуючи DAO, отримані з ExerciseDatabase, репозиторій або інші компоненти додатку можуть виконувати операції CRUD (створення, читання, оновлення та видалення) з таблицями Exercise та Workout в базі даних Room.

```
package com.example.gymscape.architecture.database;

import android.content.Context;

import androidx.room.Database;

import androidx.room.Room;

import androidx.room.RoomDatabase;

import com.example.gymscape.Model.Exercise;

import com.example.gymscape.Model.Workout;

@Database(entities = {Exercise.class, Workout.class},
version = 8)

public abstract class ExerciseDatabase extends RoomDatabase
{

    private static ExerciseDatabase instance;

    public abstract ExerciseDAO exerciseDAO();

    public abstract WorkoutDAO workoutDAO();
```

```
        public static synchronized ExerciseDatabase
getInstance(Context context)
        {
            if(instance == null)
            {
                instance = Room.databaseBuilder(context,
ExerciseDatabase.class, "exercise_database")
                    .fallbackToDestructiveMigration().build();
            }
            return instance;
        }
    }
```

3.4 Тестування додатку

Тестування - це важлива частина процесу розробки програмного забезпечення. Воно допомагає виявляти та виправляти помилки, забезпечує відповідність вимогам, підвищує якість коду та покращує загальну надійність та продуктивність програми. Існує кілька типів тестування, таких як модульне тестування, інтеграційне тестування, системне тестування, приймальне тестування та вимоги до тестування. Тестування може проводитися вручну або автоматизовано за допомогою різних інструментів та фреймворків. Воно є критично важливим для забезпечення високої якості програмного забезпечення та задоволення потреб користувачів.

Модульне тестування зосереджується на перевірці окремих модулів або компонентів програми. Метою модульного тестування є ізолювання та

тестування кожного модуля на предмет правильності його поведінки та виявлення будь-яких дефектів до їх поширення на інші частини системи.

Інтеграційне тестування перевіряє взаємодію та правильну інтеграцію різних модулів та компонентів системи. Його мета - виявити проблеми, пов'язані з інтерфейсами та взаємодією між модулями, які не були виявлені під час модульного тестування.

Системне тестування оцінює повну інтегровану систему для перевірки відповідності вимогам, правильної інтеграції компонентів і забезпечення загальної функціональності системи.

Вимоги до тестування визначають критерії та умови, за яких система вважається успішно протестованою. Вони охоплюють різні аспекти, такі як функціональність, продуктивність, безпека, доступність та сумісність.

Протестуємо функціонал додатку:

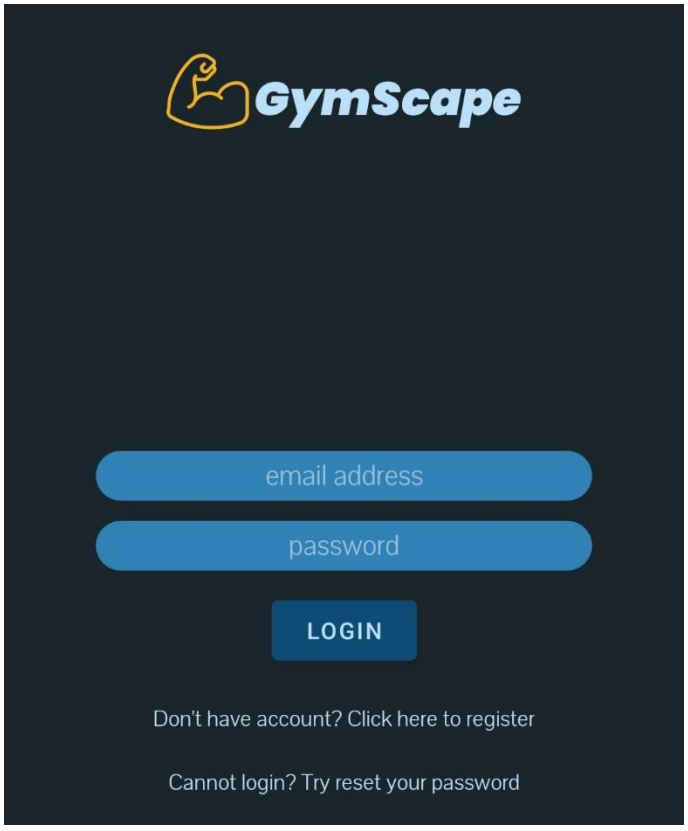

Рисунок 3.1 – Форма реєстрації

У полях вказуємо електронну пошту та пароль. На електронну пошту, яку вказано, приходить повідомлення з проханням верифікуватись (рис.3.2)

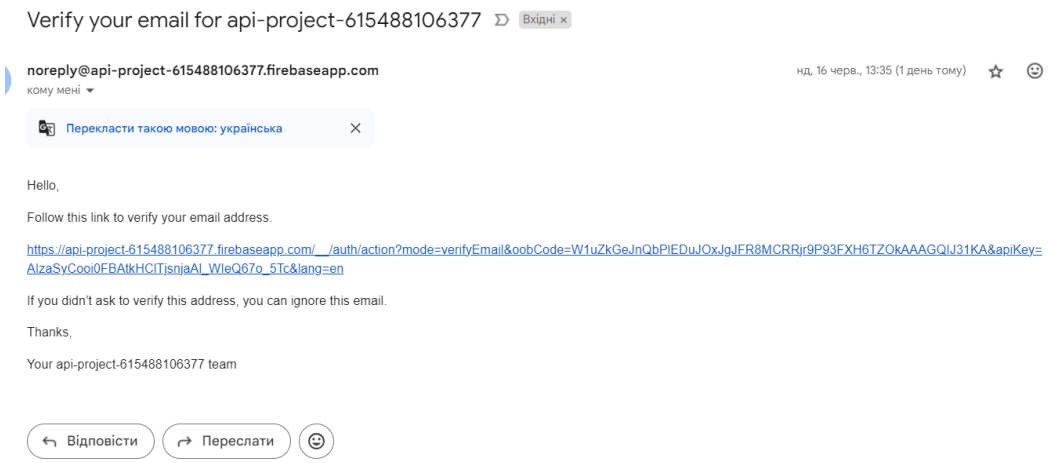

Рисунок 3.2 – Повідомлення про верифікацію

Після проходження за посиланням повідомляється що пошту успішно верифіковано та заходить в додаток на сторінку Exercises (рис 3.3)

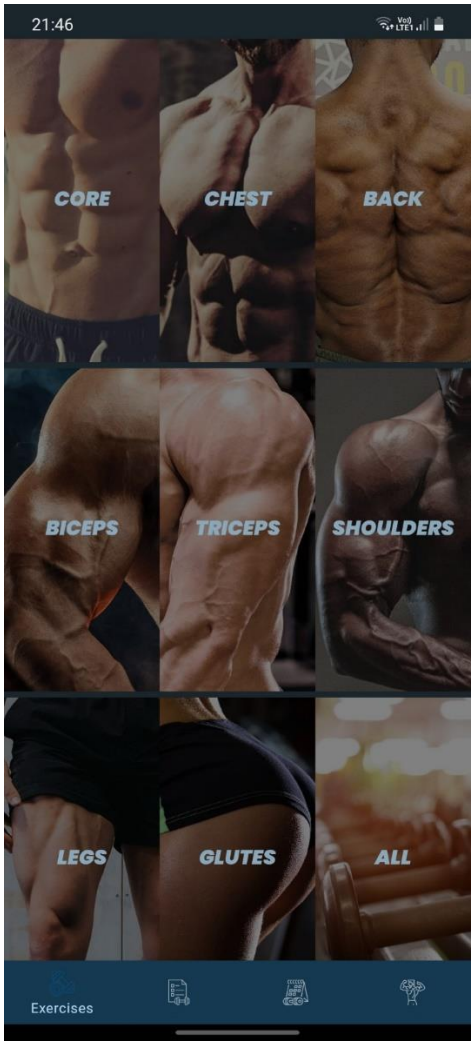

Рисунок 3.3- Сторінка Exercises

Далі перевірка на додавання та віднімання вправ у тренування (рис.3.4). Вказуємо саму вправу , дату на яку її заплановано, кількість підходів та вагу з якою буде проведена вправа,

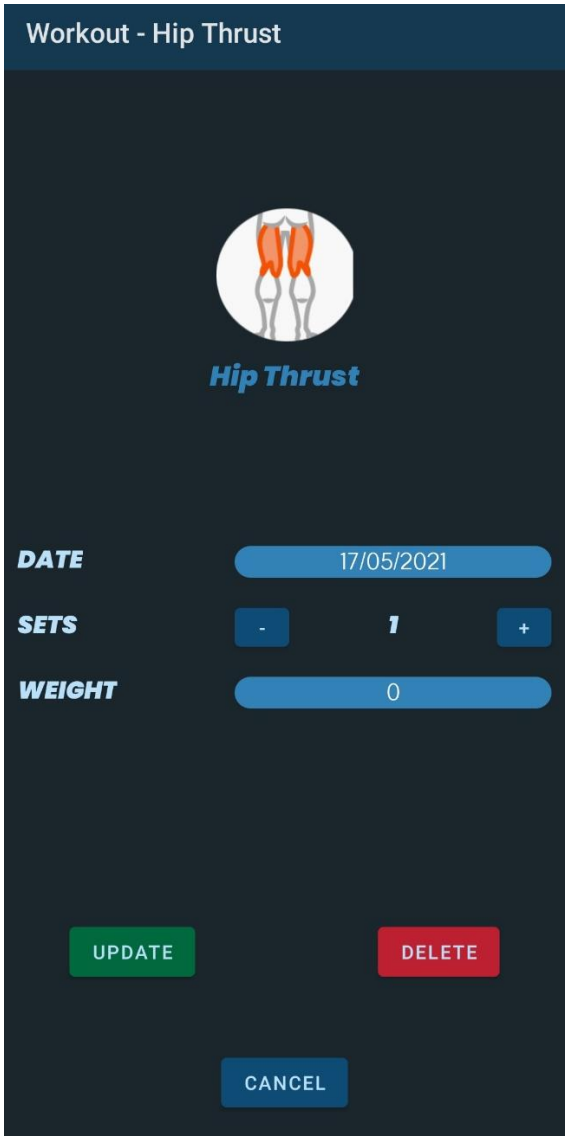

Рисунок 3.4 – Форма додавання вправи

Натиснувши кнопку Update перевіримо чи збереглась вправа на сторінці Calendar (рис.3.5). Також за допомогою цієї форми можна і видаляти вправи.

Рисунок 3.5 – Сторніка Calendar

Також протестуємо видалення вже запланованої вправи (рис.3.6). Обираємо ту вправу яку хочемо видалити та на формі тиснемо кнопку Delete.

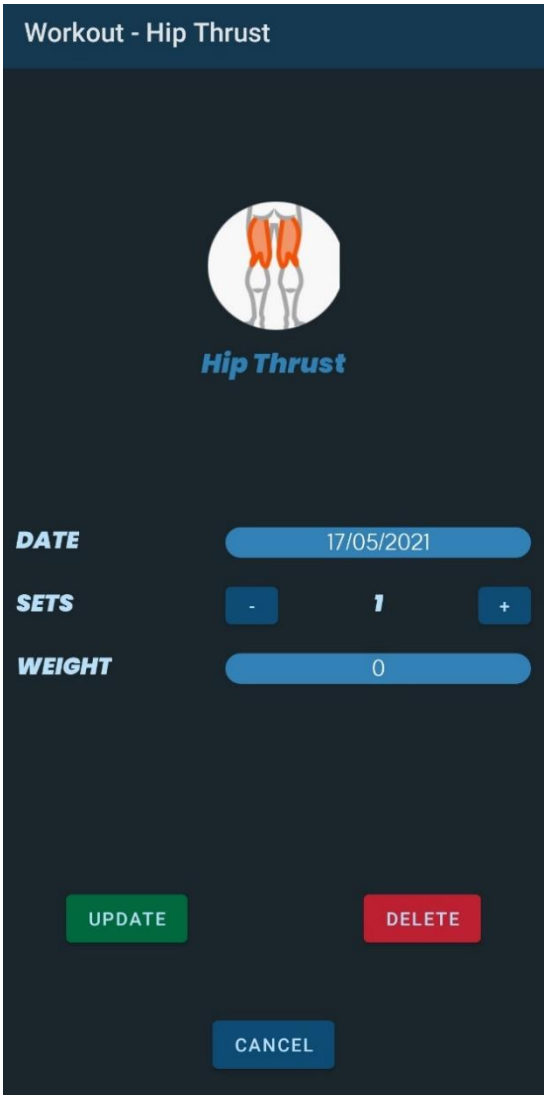

Рисунок 3.6 – Видалення вправи

Після проведених дій також перевіряємо чи зникла вправа з запланованих.

Перевіримо чи наявні вправи є в додатку з дотриманням усіх правил.

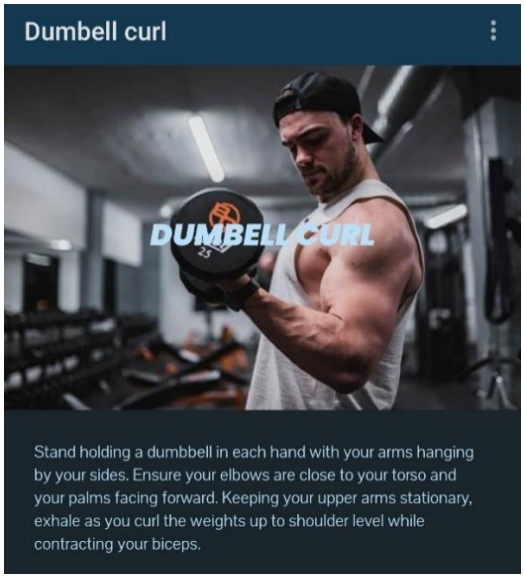

Рисунок 3.7 – Характеристика вправи

Як видно на рисунку 3.7, опис виконання вправи, назва та фото присутні.

Останнім протестуємо додавання авторської вправи кристувачем (рис.3.8).На формі вказуємо назву, опис та фото вправи та натискаємо кнопку ADD.

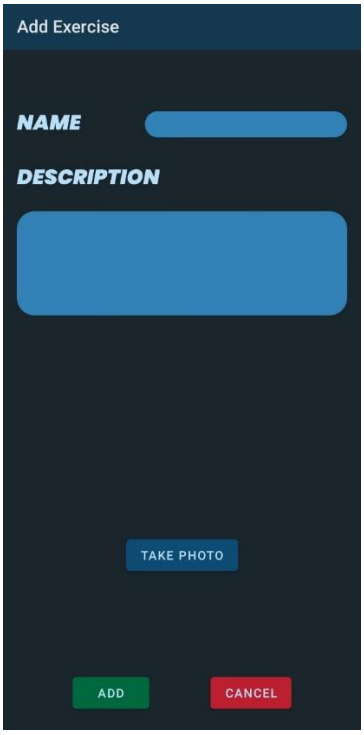

Рисунок 3.8 – Форма додавання авторської вправи

Перевіряємо чи з’явилась вправа у списку.

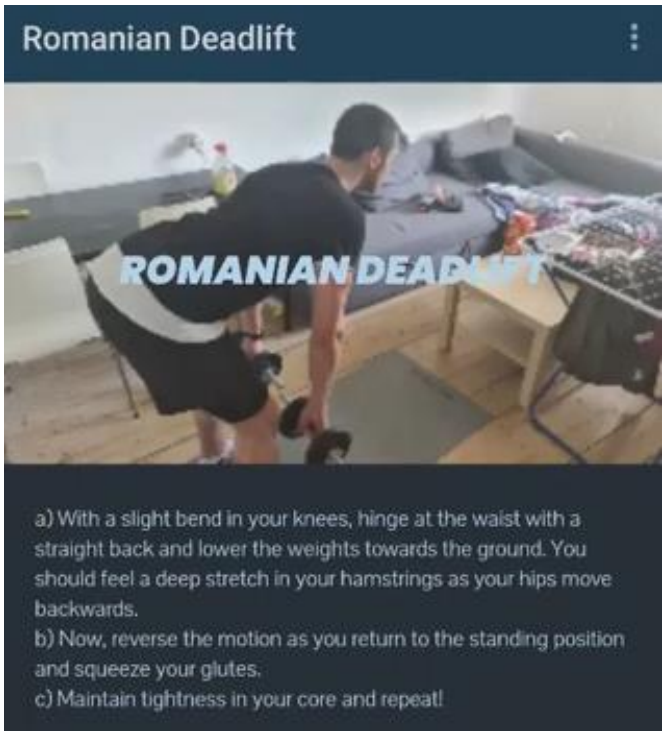

Рисунок 3.9 – Створена користувачем вправа

Під час тестування були перевірені різні функціональні можливості додатку, включаючи реєстрацію користувачів, верифікацію електронної пошти, додавання та видалення вправ, перегляд розкладу тренувань, доступність опису та фото вправ, а також створення авторських вправ користувачами.

Усі перевірені функції працювали належним чином, відповідно до очікувань та вимог. Користувачі успішно проходили реєстрацію та верифікацію, могли додавати та видаляти вправи з розкладу, переглядати деталі вправ та створювати власні авторські вправи.

Загалом, результати тестування показали, що додаток функціонує коректно та відповідає встановленим вимогам.

РОЗДІЛ 4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ

4.1 Аналіз ринку збуту продукту

Мобільний додаток "Особистий тренер" для операційної системи Android призначений для людей, які бажають вести здоровий спосіб життя, займатися фітнесом та контролювати своє харчування. Продукт є новим на ринку.

Потенційні замовники (користувачі): люди різного віку та статі, які прагнуть покращити своє фізичне здоров'я, схуднути або набрати м'язову масу.

Додаток буде реалізовуватись на ринку мобільних додатків через Google Play Store.

Очікується високий попит на додаток, оскільки все більше людей стають свідомими щодо здорового способу життя та фізичної активності.

Перевага буде надана цифровому маркетингу (реклама в соціальних мережах, пошукова оптимізація тощо) та функції "Поділитися з другом" в самому додатку.

Додаток матиме безкоштовну базову версію та платну преміум-версію з розширеними функціями.

Основні конкуренти: MyFitnessPal, Nike Training Club, Fitbod, Freeletics та інші додатки для фітнесу й здорового харчування.

Продукція конкурентів переважно має схожий функціонал (відстеження харчування, тренувальні програми, підрахунок калорій), але "Особистий тренер" буде вирізнятися персоналізованим підходом, інтуїтивним інтерфейсом та конкурентними цінами.

4.2 Розрахунок витрат на проектування

Розрахунок витрат на розробку мобільного додатку "Особистий тренер":

– Зарплата розробників: 450000 грн. Ця сума включає оплату праці команди висококваліфікованих розробників, які будуть залучені до створення додатку "Особистий тренер". До складу команди входитимуть:

– Провідний розробник Android (1 особа) - 150000 грн за 6 місяців.

– Розробники Android (2 особи) - по 90000 грн за 6 місяців на кожного.

– Веб-розробник (1 особа) - 60000 грн за 6 місяців.

– Дизайнер UI/UX (1 особа) - 45000 грн за 6 місяців.

– Аналітик вимог (1 особа) - 30000 грн за 6 місяців.

– Тестувальник (1 особа) - 45000 грн за 6 місяців.

Ця сума включає оплату праці команди висококваліфікованих розробників мобільних додатків, які будуть залучені до створення додатку "Особистий тренер". До команди входитимуть фахівці з різних галузей: мобільні розробники (Android), веб-розробники (для серверної частини), дизайнери користувацького інтерфейсу, аналітики вимог та тестувальники. Передбачається, що розробка триватиме близько 6 місяців, з урахуванням етапів проектування, кодування, тестування та налагодження.

Єдиний соціальний внесок: 99000 грн (22% від зарплати).

Відповідно до чинного законодавства, роботодавець зобов'язаний сплачувати єдиний соціальний внесок за кожного найманого працівника. Ця сума покриває відрахування на пенсійне страхування, страхування на випадок безробіття, нещасного випадку та інші соціальні внески.

Витрати на обладнання та програмне забезпечення: 90000 грн.

Для забезпечення ефективної роботи команди розробників необхідно придбати потужні комп'ютери, ноутбуки, периферійні пристрої та програмне забезпечення. Це включає операційні системи, середовища розробки,

інструменти для тестування, системи контролю версій, інструменти для проектування інтерфейсів та інше спеціалізоване програмне забезпечення.

Витрати на тестування та розгортання: 67500 грн (15% від зарплати).

Тестування є критично важливим етапом розробки, який забезпечує високу якість і надійність додатку. Витрати на тестування охоплюють оплату праці тестувальників, придбання інструментів автоматизованого тестування, а також витрати на розгортання додатку в магазинах додатків, таких як Google Play Store.

Інші прямі витрати: 112500 грн (25% від зарплати).

Ця стаття включає різноманітні додаткові витрати, пов'язані з розробкою, такі як витрати на комунікації (інтернет, телефонний зв'язок), канцелярські товари, оренду офісних приміщень, навчання та підвищення кваліфікації персоналу, а також інші непередбачені витрати.

Усього прямих витрат: 819000 грн.

Накладні витрати: 286650 грн.

Накладні витрати охоплюють адміністративні та офісні витрати, необхідні для забезпечення розробки, такі як оплата праці адміністративного персоналу, оренда офісних приміщень, комунальні послуги, обслуговування офісної техніки та іншого обладнання.

Планові накопичення: 276412 грн.

Ця сума є резервом на непередбачені витрати, які можуть виникнути протягом розробки. Це може включати непередбачені затримки, зміни у вимогах, додаткові витрати на ліцензії чи обладнання тощо. Наявність такого резерву дозволяє забезпечити гнучкість та адаптивність проекту до можливих змін.

Кошторисна вартість проекту: 1382062 грн.

Ця сума є загальною вартістю розробки додатку "Особистий тренер" і включає всі вищезазначені витрати.

Податок на додану вартість: 276412 грн (20% від кошторисної вартості).

Відповідно до чинного податкового законодавства, необхідно сплатити податок на додану вартість у розмірі 20% від кошторисної вартості проекту.

Загальна договірна ціна розробки: 1658474 грн.

Ця сума є остаточною ціною, яку необхідно заплатити за розробку мобільного додатку "Особистий тренер". Вона складається з кошторисної вартості проекту та податку на додану вартість.

Таким чином, для успішної реалізації проекту розробки мобільного додатку "Особистий тренер" необхідно інвестувати 1658474 грн. Ця сума є обґрунтованою та враховує всі необхідні витрати на персонал, обладнання, програмне забезпечення, тестування, розгортання, накладні витрати, резерви та податки. Детальний розрахунок забезпечує прозорість та дозволяє ретельно спланувати бюджет проекту.

4.3 Обґрунтування необхідності розробки

Мобільний додаток "Особистий тренер" задовольнить потреби користувачів у веденні здорового способу життя, правильному харчуванні та фізичних тренуваннях в зручному та доступному форматі.

Він вплине на поліпшення фізичного стану та самопочуття користувачів, що позитивно позначиться на їх продуктивності та якості життя.

Основні напрямки отримання ефекту:

– Економія коштів на персональних тренерах та дієтологах (додаток пропонує схожий функціонал за нижчу ціну)

– Заощадження часу на планування тренувань та підрахунок калорій (все автоматизовано)

– Поліпшення здоров'я та самопочуття користувачів (зниження ризику хвороб, підвищення працездатності)

– Зручність використання додатку в будь-якому місці та часі

Загалом, розробка мобільного додатку "Особистий тренер" є економічно обґрунтованою завдяки високому попиту на здоровий спосіб життя, конкурентним перевагам продукту та очікуваному ефекту для користувачів.

Додатково варто зазначити, що впровадження додатку дозволить залучити нових клієнтів та розширити цільову аудиторію, адже він пропонує зручний та доступний спосіб підтримувати здоровий спосіб життя без значних фінансових витрат. Завдяки наявності преміум-версії з розширеними функціями, додаток також забезпечить стабільний потік доходів у довгостроковій перспективі. Крім того, інтеграція додатку з соціальними мережами та функція "Поділитися з другом" сприятимуть вірусному поширенню та підвищенню популярності серед користувачів, що, в свою чергу, збільшить прибутковість проекту.

ВИСНОВКИ

Результатом роботи є функціональний мобільний додаток «Особистий тренер» під операційну систему Android.

У роботі було описано кожен етап розробки мобільного додатку "Особистий тренер", який призначений для ведення здорового способу життя, контролю фізичних навантажень та харчування. В результаті виконання проекту було створено повнофункціональний мобільний додаток, який повною мірою задовольняє всі вимоги, поставлені на початковому етапі розробки.

Під час створення додатку були використані сучасні технології та інструменти мобільної розробки, а також дотримані основні принципи та практики розробки програмного забезпечення. Цей процес надав безцінний практичний досвід, який, безсумнівно, стане у нагоді при реалізації майбутніх проектів. Крім того, виконання цієї роботи дозволило поглибити та закріпити знання, отримані під час вивчення дисциплін, пов'язаних з розробкою мобільних додатків, зокрема Android-програмування, розробки користувацьких інтерфейсів тощо.

Загалом, розробка мобільного додатку "Особистий тренер" стала вагомим кроком у професійному зростанні, оскільки надала цінний практичний досвід та глибше розуміння процесу створення мобільних застосунків. Завдяки виконанню цього проекту було отримано можливість застосувати теоретичні знання на практиці та випробувати себе у розробці повноцінного продукту, готового до впровадження та використання на ринку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мельник Р. А. Програмування мобільних пристроїв та змішана реальність : підручник. Львів : Видавництво Львівської політехніки, 2018. 300 с. (дата звернення: 05.06.2024)

2. Голощапов А. Л. Google Android. Програмування для мобільних пристроїв. Київ : БХВ-Київ, 2012. 448 с. (дата звернення: 18.06.2024)

3. Android Developers. Build your first Android app. URL: https://developer.android.com/training/basics/firstapp (дата звернення: 09.06.2024)

4. Козак О. Л., Пасічник В. В., Басюк Т. М. Системи мобільних технологій : навч. посіб. Львів : Новий Світ-2000, 2018. 122 с. (дата звернення: 27.06.2024)

8 Android Developers. Створення фітнес-додатків. Веб-сайт URL: https://developer.android.com/guide/health-and-fitness (дата звернення: 19.06.2024)

9 Курс "Розробка мобільних додатків для Android". Prometheus. Веб-сайт URL: https://courses.prometheus.org.ua/courses/course-v1:LITS+114+2020_T3/about (дата звернення: 11.06.2024)

10 Шевчук Д. М. Розробка мобільних додатків: від ідеї до реалізації : підручник. Київ : Видавництво "Ліра-К", 2023. 280 с. (дата звернення: 15.06.2024)

11 Бойко О. В., Мельник В. М. Android-програмування: основи та практика : навч. посіб. Львів : Видавництво Львівської політехніки, 2022. 350 с. (дата звернення: 20.06.2024)

12 DOU. Мобільна розробка. URL: https://dou.ua/lenta/tags/mobile-development/ (дата звернення: 10.06.2024)

13 Ковальчук А. М., Петренко С. В. Сучасні технології мобільного програмування : монографія. Харків : Фоліо, 2023. 200 с. (дата звернення: 25.06.2024)

ДОДАТКИ

Додаток А:

```
package com.example.gymscape.architecture.database; import
androidx.lifecycle.LiveData; import androidx.room.Dao; import
androidx.room.Delete; import androidx.room.Insert; import
androidx.room.Query; import com.example.gymscape.Model.Exercise;
import java.util.List; @Dao public interface ExerciseDAO {
@Insert void insert(Exercise exercise); @Delete void
delete(Exercise exercise); @Query("SELECT * FROM exercise_table
ORDER BY category DESC") LiveData<List<Exercise>>
getAllExercises(); @Query("DELETE FROM exercise_table") void
deleteAll(); }

package com.example.gymscape.architecture.database; import
android.content.Context; import androidx.room.Database; import
androidx.room.Room; import androidx.room.RoomDatabase; import
com.example.gymscape.Model.Exercise; import
com.example.gymscape.Model.Workout; @Database(entities =
{Exercise.class, Workout.class}, version = 8) public abstract
class ExerciseDatabase extends RoomDatabase { private static
ExerciseDatabase instance; public abstract ExerciseDAO
exerciseDAO(); public abstract WorkoutDAO workoutDAO(); public
static synchronized ExerciseDatabase getInstance(Context context)
{ if(instance == null) { instance = Room.databaseBuilder(context,
ExerciseDatabase.class, "exercise_database")
.fallbackToDestructiveMigration().build(); } return instance; } }

package com.example.gymscape.architecture.database; import
androidx.lifecycle.LiveData; import androidx.room.Dao; import
androidx.room.Delete; import androidx.room.Insert; import
androidx.room.Query; import androidx.room.Update; import
com.example.gymscape.Model.Workout; import java.util.List; @Dao
public interface WorkoutDAO { @Insert void insert(Workout
workout); @Delete void delete(Workout workout); @Update void
update(Workout workout); // I dont know how to define it in the
repository :'( (but I tried) @Query("SELECT * FROM workout_table
WHERE date= :date") LiveData<List<Workout>> getWorkoutByDate(int
date); @Query("SELECT * FROM workout_table")
LiveData<List<Workout>> getAllWorkouts(); @Query("DELETE FROM
workout_table") void deleteAll(); }

package com.example.gymscape.architecture.shared; import
android.app.Application; import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData; import
com.example.gymscape.Model.Exercise; import
com.example.gymscape.Model.Workout; import
com.example.gymscape.architecture.database.ExerciseDAO; import
com.example.gymscape.architecture.database.ExerciseDatabase;
import com.example.gymscape.architecture.database.WorkoutDAO;
```

```
import com.example.gymscape.architecture.webservices.ExerciseApi;
import
com.example.gymscape.architecture.webservices.ExerciseResponse;
import
com.example.gymscape.architecture.webservices.ServiceGenerator;
import java.util.ArrayList; import java.util.List; import
java.util.concurrent.ExecutorService; import
java.util.concurrent.Executors; import retrofit2.Call; import
retrofit2.Callback; import retrofit2.Response; import
retrofit2.internal.EverythingIsNonNull; public class
ExerciseRepository { /** SHARED **/ private static
ExerciseRepository instance; /** WEBSERVICES **/ private final
MutableLiveData<List<Exercise>> exercisesData; /** DATABASE **/
private final ExerciseDAO exerciseDAO; private final
LiveData<List<Exercise>> allExercisesDao; private final
ExecutorService executorService; private final WorkoutDAO
workoutDAO; private final MutableLiveData<List<Workout>>
workoutByDate; private final LiveData<List<Workout>> allWorkouts;
private ExerciseRepository(Application app) { /** WEBSERVICES **/
exercisesData = new MutableLiveData<>(); /** DATABASE **/
ExerciseDatabase database = ExerciseDatabase.getInstance(app);
exerciseDAO = database.exerciseDAO(); allExercisesDao =
exerciseDAO.getAllExercises(); executorService =
Executors.newFixedThreadPool(2); workoutDAO =
database.workoutDAO(); workoutByDate = new MutableLiveData<>();
allWorkouts = workoutDAO.getAllWorkouts(); } /** SHARED **/
public static synchronized  ExerciseRepository
getInstance(Application app) { if(instance == null) { instance =
new ExerciseRepository(app); } return instance; } /** WEBSERVICES
**/ public LiveData<List<Exercise>> getExercisesData() { return
exercisesData; } public void setAllExercisesData() { ExerciseApi
exerciseApi = ServiceGenerator.getExerciseApi();
Call<List<ExerciseResponse>> call = exerciseApi.getExercises();
call.enqueue(new Callback<List<ExerciseResponse>>() {
@EverythingIsNonNull @Override public void
onResponse(Call<List<ExerciseResponse>> call,
Response<List<ExerciseResponse>> response) {
ArrayList<ExerciseResponse> exerciseResponses = new
ArrayList<>(response.body()); ArrayList<Exercise> exercises = new
ArrayList<>(); for (ExerciseResponse exerciseResponse :
exerciseResponses) {
exercises.add(exerciseResponse.getExercise()); }
exercisesData.setValue(exercises); } @Override public void
onFailure(Call<List<ExerciseResponse>> call, Throwable t) { } });
} /** DATABASE **/ public LiveData<List<Exercise>>
getAllExercisesDao() { return allExercisesDao; } public void
deleteAllExercises() {
executorService.execute(exerciseDAO::deleteAll); } public void
deleteAllWorkouts() {
executorService.execute(workoutDAO::deleteAll); } public void
insert (Exercise exercise) { executorService.execute(() ->
exerciseDAO.insert(exercise)); } public void delete (Exercise
```

```
exercise) { executorService.execute(() ->
exerciseDAO.delete(exercise)); } public void insertWorkout
(Workout workout) { executorService.execute(() ->
workoutDAO.insert(workout)); } public LiveData<List<Workout>>
getWorkoutByDate(int date) { executorService.execute(() ->
workoutDAO.getWorkoutByDate(date));
workoutByDate.setValue(workoutDAO.getWorkoutByDate(date).getValue
()); return workoutByDate; } public LiveData<List<Workout>>
getAllWorkouts() { return allWorkouts; } public void
deleteWorkout(Workout workout) { executorService.execute(() ->
workoutDAO.delete(workout)); } public void updateWorkout(Workout
workout) { executorService.execute(() ->
workoutDAO.update(workout)); } }
```

```
package com.example.gymscape.architecture.webservices;  import
java.util.List;  import retrofit2.Call; import
retrofit2.http.GET;  public interface ExerciseApi {
@GET("8dea0cee-983c-4cde-8963-be67b6247516")
Call<List<ExerciseResponse>> getExercises(); }
```

```
package com.example.gymscape.architecture.webservices;  import
com.example.gymscape.Model.Exercise;  public class
ExerciseResponse {     private int id;     private String name;
private int cat;     private String description;     private
String picture;      public Exercise getExercise()     {
return new Exercise(id, name, cat, description, picture, false);
} }
```

```
package com.example.gymscape.architecture.webservices;  import
retrofit2.Retrofit; import
retrofit2.converter.gson.GsonConverterFactory;  public class
ServiceGenerator {     private static ExerciseApi exerciseApi;
public static ExerciseApi getExerciseApi()     {
if(exerciseApi == null)         {             exerciseApi = new
Retrofit.Builder()
.baseUrl("https://run.mocky.io/v3/")
.addConverterFactory(GsonConverterFactory.create())
.build()                    .create(ExerciseApi.class);
}         return exerciseApi;     } }
```

```
package com.example.gymscape.Model;  import androidx.room.Entity;
import androidx.raoom.Ignore; import androidx.room.PrimaryKey;
import java.io.Serializable;  @Entity(tableName =
"exercise_table") public class Exercise implements Serializable {
@PrimaryKey(autoGenerate = true)     private int id;     private
String name;     private int category;     private String
description;     private String picture;     private boolean
isDatabase;      @Ignore     public Exercise(int id, String name,
int category, String description, String picture, boolean
isDatabase) {         this.id = id;         this.name = name;
this.category = category;         this.description = description;
this.picture = picture;         this.isDatabase = isDatabase;
```

```
}      public Exercise(String name, int category, String
description, String picture, boolean isDatabase) {
this.name = name;         this.category = category;
this.description = description;         this.picture = picture;
this.isDatabase = isDatabase;     }      public int getId() {
return id;     }      public String getName() {         return
name;     }      public int getCategory() {         return
category;     }      public String getDescription() {
return description;     }      public String getPicture() {
return picture;     }      public void setId(int id) {
this.id = id;     }      public void setName(String name) {
this.name = name;     }      public void setCategory(int
category) {         this.category = category;     }      public
void setDescription(String description) {
this.description = description;     }      public void
setPicture(String picture) {         this.picture = picture;
}      public boolean isDatabase() {         return isDatabase;
}      public void setIsDatabase(boolean isDatabase) {
this.isDatabase = isDatabase;     } }

package com.example.gymscape.Model;  import
androidx.annotation.Nullable; import androidx.room.Entity; import
androidx.room.Ignore; import androidx.room.PrimaryKey;  import
java.io.Serializable;  @Entity(tableName = "workout_table")
public class Workout implements Serializable {
@PrimaryKey(autoGenerate = true)     private int id;     private
String exerciseName;     private int category;     private int
date;     private int sets;     private int weight;      @Ignore
public Workout(int id, String exerciseName, int category, int
date, int sets, int weight) {         this.id = id;
this.exerciseName = exerciseName;         this.category =
category;         this.date = date;         this.sets = sets;
this.weight = weight;     }      @Ignore     public Workout()
{      }      public Workout(String exerciseName, int category,
int date, int sets, int weight) {         this.exerciseName =
exerciseName;         this.category = category;         this.date
= date;         this.sets = sets;         this.weight = weight;
}      public int getId() {         return id;     }      public
void setId(int id) {         this.id = id;     }      public
String getExerciseName() {         return exerciseName;     }
public void setExerciseName(String exerciseName) {
this.exerciseName = exerciseName;     }      public int
getCategory() {         return category;     }      public void
setCategory(int category) {         this.category = category;
}      public int getDate() {         return date;     }
public void setDate(int date) {         this.date = date;     }
public int getSets() {         return sets;     }      public
void setSets(int sets) {         this.sets = sets;     }
public int getWeight() {         return weight;     }      public
void setWeight(int weight) {         this.weight = weight;     }
@Override     public String toString() {         return
"Workout{" +                 ", exerciseName='" + exerciseName +
```

```
'\'' +                  ", category=" + category +
", date=" + date +                  ", sets=" + sets +
", weight=" + weight +                  '}';     } }

package com.example.gymscape.ui.editworkout;  import
androidx.appcompat.app.AppCompatActivity; import
androidx.core.content.ContextCompat; import
androidx.lifecycle.Observer; import
androidx.lifecycle.ViewModelProvider;  import
android.app.DatePickerDialog; import android.content.Intent;
import android.graphics.drawable.ColorDrawable; import
android.os.Bundle; import android.text.format.DateFormat; import
android.util.Log; import android.widget.Button; import
android.widget.DatePicker; import android.widget.EditText; import
android.widget.ImageView; import android.widget.TextView; import
android.widget.Toast;  import com.example.gymscape.Model.Workout;
import com.example.gymscape.R; import
com.example.gymscape.SharedFunctions; import
com.example.gymscape.ui.MainActivity; import
com.example.gymscape.ui.UsedEnums;  import
java.text.ParseException; import java.text.SimpleDateFormat;
import java.time.ZoneId; import java.util.Calendar; import
java.util.Date; import java.util.Locale;  public class
EditWorkoutActivity extends AppCompatActivity {      private
EditWorkoutViewModel viewModel;      Workout workout;
ImageView exerciseImageCategory;     TextView exerciseName;
EditText editTextDate;     TextView setsCountTextView;
EditText weightTextNumber;     Button updateWorkoutButton;
Button deleteWorkoutButton;     Button cancelWorkoutButton;
Button increaseSets;     Button decreaseSets;      Calendar
calendar;      @Override     protected void onCreate(Bundle
savedInstanceState) {         super.onCreate(savedInstanceState);
setContentView(R.layout.activity_edit_workout);
viewModel = new ViewModelProvider(this,
ViewModelProvider.AndroidViewModelFactory.getInstance(getApplicat
ion())).get(EditWorkoutViewModel.class);          Intent intent =
getIntent();         workout = (Workout)
intent.getSerializableExtra(UsedEnums.WORKOUT.toString());
exerciseImageCategory = findViewById(R.id.exerciseImageCategory);
exerciseName = findViewById(R.id.exerciseName);
editTextDate = findViewById(R.id.editTextDate);
setsCountTextView = findViewById(R.id.setsCountTextView);
weightTextNumber = findViewById(R.id.weightTextNumber);
updateWorkoutButton = findViewById(R.id.updateWorkoutButton);
deleteWorkoutButton = findViewById(R.id.deleteWorkoutButton);
cancelWorkoutButton = findViewById(R.id.cancelWorkoutButton);
increaseSets = findViewById(R.id.increaseSetsButton);
decreaseSets = findViewById(R.id.decreaseSetsButton);
calendar = Calendar.getInstance();
this.getSupportActionBar().setTitle("Workout - " +
workout.getExerciseName());
this.getSupportActionBar().setBackgroundDrawable(new
```

```
ColorDrawable(ContextCompat.getColor(this, R.color.darkBlue)));
exerciseImageCategory.setImageResource(SharedFunctions.getIcon(wo
rkout.getCategory()));
exerciseName.setText(workout.getExerciseName());
weightTextNumber.setText("" + workout.getWeight());
viewModel.setSetsCount(workout.getSets());          String
newDate;          if(workout.getDate() < 10000000)
newDate = "0" + workout.getDate();          else
newDate = "" + workout.getDate();          try {              Date
savedDate = new SimpleDateFormat("ddMMyyyy").parse(newDate);
viewModel.setDate(savedDate);          } catch (ParseException e)
{              e.printStackTrace();          }
viewModel.getDate().observe(this, chosenDate ->{              int
year = Integer.parseInt((String) DateFormat.format("yyyy",
chosenDate));              int month = Integer.parseInt((String)
DateFormat.format("MM", chosenDate));              int dayOfMonth
= Integer.parseInt((String) DateFormat.format("dd", chosenDate));
DatePickerDialog.OnDateSetListener date = new
DatePickerDialog.OnDateSetListener() {                @Override
public void onDateSet(DatePicker view, int year, int month, int
dayOfMonth) {                    calendar.set(Calendar.YEAR,
year);                    calendar.set(Calendar.MONTH, month);
calendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
viewModel.setDate(calendar.getTime());                }
};              editTextDate.setOnClickListener(v -> {
new DatePickerDialog(this, date, year, month-1,
dayOfMonth).show();              });              String myFormat
= "dd/MM/yyyy";              SimpleDateFormat dateFormat = new
SimpleDateFormat(myFormat, Locale.UK);
editTextDate.setText(dateFormat.format(chosenDate));
workout.setDate(SharedFunctions.getDate(chosenDate.getTime()));
});          decreaseSets.setOnClickListener(v -> {
if(viewModel.getSetsCount().getValue() <= 0)
Toast.makeText(this, "Minimum value of sets is 0",
Toast.LENGTH_SHORT).show();              else              {
int setsValue =
Integer.parseInt(setsCountTextView.getText().toString());
viewModel.setSetsCount(--setsValue);              }          });
increaseSets.setOnClickListener(v -> {
if(viewModel.getSetsCount().getValue() >= 10)
Toast.makeText(this, "Maximum value of sets is 10",
Toast.LENGTH_SHORT).show();              else              {
int setsValue =
Integer.parseInt(setsCountTextView.getText().toString());
viewModel.setSetsCount(++setsValue);              }          });
viewModel.getSetsCount().observe(this, new Observer<Integer>() {
@Override              public void onChanged(Integer integer) {
setsCountTextView.setText("" + integer);
workout.setSets(integer);              }          });
updateWorkoutButton.setOnClickListener(v -> {
if(workout.getDate() <= 0)                  Toast.makeText(this,
"You must set date to continue.", Toast.LENGTH_SHORT).show();
```

```
else {                 if
(weightTextNumber.getText().toString().equals(""))
workout.setWeight(0);                 else
workout.setWeight(Integer.parseInt(weightTextNumber.getText().toS
tring()));                 viewModel.update(workout);
Intent mainIntent = new Intent(this, MainActivity.class);
startActivity(mainIntent);                 finish();
}         });          deleteWorkoutButton.setOnClickListener(v -
> {             viewModel.delete(workout);             Intent
mainActivityIntent = new Intent(this, MainActivity.class);
startActivity(mainActivityIntent);             finish();
});          cancelWorkoutButton.setOnClickListener(v -> {
Intent mainActivityIntent = new Intent(this, MainActivity.class);
startActivity(mainActivityIntent);             finish();

package com.example.gymscape.ui.editworkout;  import
android.app.Application; import androidx.annotation.NonNull;
import androidx.lifecycle.AndroidViewModel; import
androidx.lifecycle.LiveData; import
androidx.lifecycle.MutableLiveData;  import
com.example.gymscape.Model.Workout; import
com.example.gymscape.architecture.shared.ExerciseRepository;
import java.util.Date;  public class EditWorkoutViewModel extends
AndroidViewModel {      ExerciseRepository repository;
MutableLiveData<Integer> setsCount;     MutableLiveData<Date>
date;      public EditWorkoutViewModel(Application application) {
super(application);         repository =
ExerciseRepository.getInstance(application);         setsCount =
new MutableLiveData<>();         date = new MutableLiveData<>();
}      public LiveData<Integer> getSetsCount() {         return
setsCount;     }      public void setSetsCount(int setsCount) {
this.setsCount.setValue(setsCount);     }      public
LiveData<Date> getDate() {         return date;     }      public
void setDate(Date date) {         this.date.setValue(date);     }
public void delete(Workout workout)     {
repository.deleteWorkout(workout);     }      public void
update(Workout workout)     {
repository.updateWorkout(workout);     } }        });     } }

package com.example.gymscape.ui.exercise;  import
androidx.annotation.NonNull; import
androidx.appcompat.app.AppCompatActivity; import
androidx.core.content.ContextCompat; import
androidx.lifecycle.ViewModelProvider;  import
android.content.Intent; import
android.graphics.drawable.ColorDrawable; import
android.os.Bundle; import android.view.Menu; import
android.view.MenuInflater; import android.view.MenuItem; import
android.widget.ImageView; import android.widget.TextView; import
android.widget.Toast;  import com.bumptech.glide.Glide; import
com.example.gymscape.Model.Exercise; import
com.example.gymscape.R; import com.example.gymscape.ui.UsedEnums;
```

```
import com.example.gymscape.ui.exerciselist.ExerciseActivity;
import com.example.gymscape.ui.newworkout.NewWorkoutActivity;
public class SpecificExerciseActivity extends AppCompatActivity {
private Exercise exercise;      ImageView exerciseImage;
TextView exerciseDescription;     TextView exerciseName;
SpecificExerciseViewModel viewModel;      @Override     protected
void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_specific_exercise);
viewModel = new ViewModelProvider(this,
ViewModelProvider.AndroidViewModelFactory.getInstance(getApplicat
ion())).get(SpecificExerciseViewModel.class);          Intent
intent = getIntent();         exercise = (Exercise)
intent.getSerializableExtra(UsedEnums.EXERCISE.toString());
viewModel.setExerciseMD(exercise);          exerciseImage =
findViewById(R.id.exerciseImage);         exerciseDescription =
findViewById(R.id.exerciseDescription);         exerciseName =
findViewById(R.id.exerciseName);
viewModel.getExerciseMD().observe(this, exerciseData -> {
this.getSupportActionBar().setTitle(exerciseData.getName());
this.getSupportActionBar().setBackgroundDrawable(new
ColorDrawable(ContextCompat.getColor(this, R.color.darkBlue)));
if(exercise.getPicture().isEmpty())
exerciseImage.setImageResource(R.drawable.exercise_universal_pict
ure);             else
Glide.with(this).load(exercise.getPicture()).into(exerciseImage);
exerciseDescription.setText(exercise.getDescription() +
"\n\n\n\n\n\n\n\n\n");
exerciseName.setText(exercise.getName().toUpperCase());
});     }      @Override     public boolean
onCreateOptionsMenu(Menu menu) {         MenuInflater inflater =
getMenuInflater();
inflater.inflate(R.menu.top_menu_exercise, menu);         return
true;     }      @Override     public boolean
onOptionsItemSelected(@NonNull MenuItem item) {
if(item.getItemId() == R.id.addToWorkoutMenu)         {
Intent intent = new Intent(this, NewWorkoutActivity.class);
intent.putExtra(UsedEnums.EXERCISE.toString(), exercise);
startActivity(intent);             finish();             return
true;         }         else if(item.getItemId() ==
R.id.deleteExerciseMenu)         {
if(exercise.isDatabase())             {
viewModel.delete(exercise);                  Intent intent = new
Intent(this, ExerciseActivity.class);
intent.putExtra(UsedEnums.CATEGORY.toString(),
exercise.getCategory());                 startActivity(intent);
finish();                 return true;             }
else             {                 Toast.makeText(this, "You
cannot delete this exercise.", Toast.LENGTH_SHORT).show();
return false;             }         }             return
super.onOptionsItemSelected(item);     } }
```

```
package com.example.gymscape.ui.exercise;  import
android.app.Application; import android.view.View;  import
androidx.lifecycle.AndroidViewModel; import
androidx.lifecycle.LiveData; import
androidx.lifecycle.MutableLiveData; import
androidx.lifecycle.ViewModel;  import
com.example.gymscape.Model.Exercise; import
com.example.gymscape.architecture.shared.ExerciseRepository;
import java.util.ArrayList; import java.util.List;  public class
SpecificExerciseViewModel extends AndroidViewModel {
MutableLiveData<Exercise> exerciseMD;     ExerciseRepository
repository;      public SpecificExerciseViewModel(Application
app)     {         super(app);         exerciseMD = new
MutableLiveData<>();         repository =
ExerciseRepository.getInstance(app);     }      public
LiveData<Exercise> getExerciseMD() {         return exerciseMD;
}      public void setExerciseMD(Exercise exercise) {
this.exerciseMD.setValue(exercise);     }      public void
delete(final Exercise exercise)     {
repository.delete(exercise);     } }

package com.example.gymscape.ui.exerciselist;  import
androidx.appcompat.app.AppCompatActivity; import
androidx.core.content.ContextCompat; import
androidx.lifecycle.ViewModelProvider; import
androidx.recyclerview.widget.LinearLayoutManager; import
androidx.recyclerview.widget.RecyclerView;  import
android.content.Intent; import
android.graphics.drawable.ColorDrawable; import
android.os.Bundle; import android.view.View;  import
com.example.gymscape.Model.Exercise; import
com.example.gymscape.R; import
com.example.gymscape.ui.MainActivity; import
com.example.gymscape.ui.UsedEnums; import
com.example.gymscape.ui.exercise.SpecificExerciseActivity; import
com.example.gymscape.ui.main.exercises.ExercisesFragment; import
com.example.gymscape.ui.newexercise.NewExerciseActivity; import
com.google.android.material.floatingactionbutton.FloatingActionBu
tton;  import java.util.ArrayList; import java.util.Arrays;
public class ExerciseActivity extends AppCompatActivity
implements ExerciseAdapter.OnListItemClickListener{
ExerciseViewModel viewModel;     RecyclerView recyclerView;
ExerciseAdapter adapter;     FloatingActionButton fab;     int
category;     ArrayList<Exercise> exercises = new ArrayList<>();
@Override     protected void onCreate(Bundle savedInstanceState)
{         super.onCreate(savedInstanceState);
setContentView(R.layout.activity_exercise);
ArrayList<String> categoryNames = new
ArrayList<>(Arrays.asList("All", "Core", "Chest", "Back",
"Biceps", "Triceps", "Shoulders", "Legs", "Glutes"));
Intent intent = getIntent();         category =
intent.getIntExtra(UsedEnums.CATEGORY.toString(), 0);
```

```
this.getSupportActionBar().setTitle( categoryNames.get(category)
+ " " + getResources().getString(R.string.title_exercises));
this.getSupportActionBar().setBackgroundDrawable(new
ColorDrawable(ContextCompat.getColor(this, R.color.darkBlue)));
viewModel = new ViewModelProvider(this,
ViewModelProvider.AndroidViewModelFactory.getInstance(getApplicat
ion())).get(ExerciseViewModel.class);          fab =
findViewById(R.id.addNewExerciseButton);         if(category ==
0)             fab.setVisibility(View.GONE);
fab.setOnClickListener(v -> {             Intent newProfileIntent
= new Intent(this, NewExerciseActivity.class);
newProfileIntent.putExtra(UsedEnums.CATEGORY.toString(),
category);             startActivity(newProfileIntent);
finish();         });          recyclerView =
findViewById(R.id.recyclerView);
recyclerView.hasFixedSize();
recyclerView.setLayoutManager(new LinearLayoutManager(this));
viewModel.setExercise();          adapter = new
ExerciseAdapter(exercises, this);
recyclerView.setAdapter(adapter);          exercises.clear();
viewModel.getExercises().observe(this, exerciseCollection ->{
for(Exercise exercise : exerciseCollection)             {
if(category == 0)                      exercises.add(exercise);
else if(exercise.getCategory() == category)
exercises.add(exercise);             }
adapter.notifyDataSetChanged();          });     }      @Override
public void onListItemClick(int clickedItemIndex) {
Intent toSpecificExercise = new Intent(this,
SpecificExerciseActivity.class);
toSpecificExercise.putExtra(UsedEnums.EXERCISE.toString(),
adapter.exercisesList.get(clickedItemIndex));
startActivity(toSpecificExercise);          finish();     }
@Override     protected void onResume() {
super.onResume();
viewModel.getAllExercisesDAO().observe(this, exercisesDAO ->{
exercises.clear();             for(Exercise exercise :
exercisesDAO)             {                 if(category == 0)
exercises.add(exercise);                 else
if(exercise.getCategory() == category)
exercises.add(exercise);             }
adapter.notifyDataSetChanged();          });     } }

package com.example.gymscape.ui.exerciselist;  import
androidx.annotation.NonNull; import
androidx.recyclerview.widget.RecyclerView;  import
android.util.Log; import android.view.LayoutInflater; import
android.view.View; import android.view.ViewGroup; import
android.widget.ImageView; import android.widget.TextView; import
android.widget.Toast;  import
com.example.gymscape.Model.Exercise; import
com.example.gymscape.R;  import java.util.ArrayList;  public
class ExerciseAdapter extends
```

```
RecyclerView.Adapter<ExerciseAdapter.ViewHolder> {      public
ArrayList<Exercise> exercisesList;     final private
OnListItemClickListener mOnListItemClickListener;
ExerciseAdapter(ArrayList<Exercise> exercises,
OnListItemClickListener listener){         exercisesList =
exercises;         mOnListItemClickListener = listener;     }
@NonNull     public ViewHolder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {         LayoutInflater inflater
= LayoutInflater.from(parent.getContext());         View view =
inflater.inflate(R.layout.exercise_list_item, parent, false);
return new ViewHolder(view);     }      public void
onBindViewHolder(@NonNull ViewHolder viewHolder, int position) {
viewHolder.name.setText(exercisesList.get(position).getName());
switch (exercisesList.get(position).getCategory())         {
case 1:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_core);
break;             case 2:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_chest);
break;             case 3:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_back);
break;             case 4:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_biceps);
break;             case 5:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_triceps);
break;             case 6:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_shoulder);
break;             case 7:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_legs);
break;             case 8:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_glutes);
break;         }     }      public int getItemCount() {
return exercisesList.size();     }      class ViewHolder extends
RecyclerView.ViewHolder implements View.OnClickListener {
TextView name;         ImageView iconMuscle;
ViewHolder(View itemView) {             super(itemView);
name = itemView.findViewById(R.id.exerciseName);
iconMuscle = itemView.findViewById(R.id.iconMuscle);
itemView.setOnClickListener(this);         }          @Override
public void onClick(View v) {
mOnListItemClickListener.onListItemClick(getAdapterPosition());
}     }      public interface OnListItemClickListener {
void onListItemClick(int clickedItemIndex);     } }

package com.example.gymscape.ui.exercisepackage
com.example.gymscape.ui.exerciselist;  import
android.app.Application;  import
androidx.lifecycle.AndroidViewModel; import
androidx.lifecycle.LiveData; import
androidx.lifecycle.MutableLiveData;  import
com.example.gymscape.Model.Exercise; import
com.example.gymscape.architecture.shared.ExerciseRepository;
import java.util.ArrayList; import java.util.List;  public class
```

```
ExerciseViewModel extends AndroidViewModel {
ExerciseRepository exerciseRepository;      public
ExerciseViewModel(Application app)     {         super(app);
exerciseRepository = ExerciseRepository.getInstance(app);     }
LiveData<List<Exercise>> getExercises()     {         return
exerciseRepository.getExercisesData();     }      public void
setExercise()     {
exerciseRepository.setAllExercisesData();     }      /** DATABASE
**/     public LiveData<List<Exercise>> getAllExercisesDAO()
{         return exerciseRepository.getAllExercisesDao();     }
public void insert(final Exercise exercise)     {
exerciseRepository.insert(exercise);     }      public void
delete(final Exercise exercise)     {
exerciseRepository.delete(exercise);     } }list;  import
androidx.annotation.NonNull; import
androidx.recyclerview.widget.RecyclerView;  import
android.util.Log; import android.view.LayoutInflater; import
android.view.View; import android.view.ViewGroup; import
android.widget.ImageView; import android.widget.TextView; import
android.widget.Toast;  import
com.example.gymscape.Model.Exercise; import
com.example.gymscape.R;  import java.util.ArrayList;  public
class ExerciseAdapter extends
RecyclerView.Adapter<ExerciseAdapter.ViewHolder> {      public
ArrayList<Exercise> exercisesList;     final private
OnListItemClickListener mOnListItemClickListener;
ExerciseAdapter(ArrayList<Exercise> exercises,
OnListItemClickListener listener){         exercisesList =
exercises;         mOnListItemClickListener = listener;     }
@NonNull     public ViewHolder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {         LayoutInflater inflater
= LayoutInflater.from(parent.getContext());         View view =
inflater.inflate(R.layout.exercise_list_item, parent, false);
return new ViewHolder(view);     }      public void
onBindViewHolder(@NonNull ViewHolder viewHolder, int position) {
viewHolder.name.setText(exercisesList.get(position).getName());
switch (exercisesList.get(position).getCategory())         {
case 1:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_core);
break;             case 2:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_chest);
break;             case 3:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_back);
break;             case 4:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_biceps);
break;             case 5:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_triceps);
break;             case 6:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_shoulder);
break;             case 7:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_legs);
break;             case 8:
```

```
viewHolder.iconMuscle.setImageResource(R.drawable.icon_glutes);
break;          }     }      public int getItemCount() {
return exercisesList.size();     }      class ViewHolder extends
RecyclerView.ViewHolder implements View.OnClickListener {
TextView name;          ImageView iconMuscle;
ViewHolder(View itemView) {             super(itemView);
name = itemView.findViewById(R.id.exerciseName);
iconMuscle = itemView.findViewById(R.id.iconMuscle);
itemView.setOnClickListener(this);          }          @Override
public void onClick(View v) {
mOnListItemClickListener.onListItemClick(getAdapterPosition());
}     }      public interface OnListItemClickListener {
void onListItemClick(int clickedItemIndex);     } }

package com.example.gymscape.ui.login;  import
androidx.annotation.NonNull; import
androidx.appcompat.app.AppCompatActivity;  import
android.content.Intent; import android.content.SharedPreferences;
import android.os.Bundle; import android.util.Log; import
android.widget.Button; import android.widget.EditText; import
android.widget.TextView; import android.widget.Toast;  import
com.example.gymscape.R; import
com.example.gymscape.SharedFunctions; import
com.google.android.gms.tasks.OnCompleteListener; import
com.google.android.gms.tasks.Task; import
com.google.firebase.auth.AuthResult; import
com.google.firebase.auth.FirebaseAuth; import
com.example.gymscape.ui.MainActivity;  public class LoginActivity
extends AppCompatActivity {      TextView loginRegister,
passwordReset;     FirebaseAuth firebaseAuth;     EditText
emailAddress, password;     Button loginButton;      @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
this.getSupportActionBar().hide();
setContentView(R.layout.activity_login);          firebaseAuth =
FirebaseAuth.getInstance();
if(firebaseAuth.getCurrentUser() != null)         {
Log.i("Logged in", "true");             startActivity(new
Intent(this, MainActivity.class));             finish();
}          loginRegister = findViewById(R.id.loginRegister);
passwordReset = findViewById(R.id.passwordReset);
loginRegister.setOnClickListener(v -> {
startActivity(new Intent(this, RegisterActivity.class));
finish();         });          passwordReset.setOnClickListener(v
-> {             startActivity(new Intent(this,
ResetPasswordActivity.class));             finish();         });
emailAddress = findViewById(R.id.loginUsername);         password
= findViewById(R.id.loginPassword);         loginButton =
findViewById(R.id.loginButton);
loginButton.setOnClickListener(v -> {
if(emailAddress.getText().toString().isEmpty() ||
password.getText().toString().isEmpty())
```

```
Toast.makeText(this, "Please fill all information.",
Toast.LENGTH_SHORT).show();             else
if(!SharedFunctions.isEmailValid(emailAddress.getText().toString(
)))                 Toast.makeText(this, "Wrong email format.",
Toast.LENGTH_SHORT).show();             else {
firebaseAuth.signInWithEmailAndPassword(emailAddress.getText().to
String().trim(),
password.getText().toString()).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {                      @Override
public void onComplete(@NonNull Task<AuthResult> task) {
if (task.isSuccessful()) {
if(firebaseAuth.getCurrentUser().isEmailVerified()) {
startActivity(new Intent(getApplicationContext(),
MainActivity.class));                                   finish();
}                               else
Toast.makeText(LoginActivity.this, "You need to verify your
email.", Toast.LENGTH_SHORT).show();                            }
else {
Toast.makeText(LoginActivity.this, "This email is not registered
yet.", Toast.LENGTH_SHORT).show();                              }
}                 });             }         });     } }

package com.example.gymscape.ui.login;  import
androidx.annotation.NonNull; import
androidx.appcompat.app.AppCompatActivity;  import
android.content.Intent; import android.content.SharedPreferences;
import android.os.Bundle; import android.widget.Button; import
android.widget.EditText; import android.widget.TextView; import
android.widget.Toast;  import com.example.gymscape.R; import
com.example.gymscape.SharedFunctions; import
com.google.android.gms.tasks.OnCompleteListener; import
com.google.android.gms.tasks.Task; import
com.google.firebase.auth.AuthResult; import
com.google.firebase.auth.FirebaseAuth; import
com.example.gymscape.ui.MainActivity;  public class
RegisterActivity extends AppCompatActivity {      TextView
alreadyRegistered;     EditText emailAddress, password,
password2;     Button registerButton;     FirebaseAuth
firebaseAuth;     SharedPreferences preferences;      @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
this.getSupportActionBar().hide();
setContentView(R.layout.activity_register);          firebaseAuth
= FirebaseAuth.getInstance();
if(firebaseAuth.getCurrentUser() != null)         {
startActivity(new Intent(this, MainActivity.class));
finish();         }          preferences =
getSharedPreferences("profile", MODE_PRIVATE);
alreadyRegistered = findViewById(R.id.alreadyRegistered);
alreadyRegistered.setOnClickListener(v -> {
startActivity(new Intent(this, LoginActivity.class));
finish();         });          emailAddress =
```

```
findViewById(R.id.registerEmailAddress);         password =
findViewById(R.id.registerPassword1);         password2 =
findViewById(R.id.registerPassword2);         registerButton =
findViewById(R.id.registerButton);
registerButton.setOnClickListener(v -> {
if(password.getText().toString().isEmpty() ||
password2.getText().toString().isEmpty() ||
emailAddress.getText().toString().isEmpty())
Toast.makeText(this, "Please fill all information.",
Toast.LENGTH_SHORT).show();             else
if(!password.getText().toString().equals(password2.getText().toSt
ring()))                  Toast.makeText(this, "Passwords do not
match.", Toast.LENGTH_SHORT).show();             else
if(password.getText().toString().length() < 6)
Toast.makeText(this, "Password must have at least 6 characters",
Toast.LENGTH_SHORT).show();             else
if(!SharedFunctions.isEmailValid(emailAddress.getText().toString(
)))                 Toast.makeText(this, "Wrong email format.",
Toast.LENGTH_SHORT).show();             else            {
firebaseAuth.createUserWithEmailAndPassword(emailAddress.getText(
).toString().trim(),
password.getText().toString()).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {                    @Override
public void onComplete(@NonNull Task<AuthResult> task) {
if(task.isSuccessful())                          {
firebaseAuth.getCurrentUser().sendEmailVerification();
SharedPreferences.Editor editor = preferences.edit();
editor.putString("email", emailAddress.getText().toString());
editor.apply();
firebaseAuth.signOut();
Toast.makeText(RegisterActivity.this, "Registration was
successful. Account created.", Toast.LENGTH_LONG).show();
startActivity(new Intent(getApplicationContext(),
LoginActivity.class));                              finish();
}                         else
Toast.makeText(RegisterActivity.this, "Email is already
registered.", Toast.LENGTH_SHORT).show();                    }

package com.example.gymscape.ui.login;  import
androidx.annotation.NonNull; import
androidx.appcompat.app.AppCompatActivity;  import
android.content.Intent; import android.os.Bundle; import
android.widget.Button; import android.widget.EditText; import
android.widget.TextView; import android.widget.Toast;  import
com.example.gymscape.R; import
com.example.gymscape.SharedFunctions; import
com.google.android.gms.tasks.OnCompleteListener; import
com.google.android.gms.tasks.Task; import
com.google.firebase.auth.FirebaseAuth;  public class
ResetPasswordActivity extends AppCompatActivity {      EditText
emailAddress;     TextView backToLogin;     Button confirmReset;
FirebaseAuth firebaseAuth;      @Override     protected void
```

```
onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_reset_password);
this.getSupportActionBar().hide();          firebaseAuth =
FirebaseAuth.getInstance();          emailAddress =
findViewById(R.id.passwordResetEmail);         backToLogin =
findViewById(R.id.loginTextView);         confirmReset =
findViewById(R.id.resetButton);
backToLogin.setOnClickListener(v -> {
startActivity(new Intent(this, LoginActivity.class));
finish();         });          confirmReset.setOnClickListener(v
-> {             if(emailAddress.getText().toString().isEmpty())
Toast.makeText(this, "Please fill your email address",
Toast.LENGTH_SHORT).show();             else
if(!SharedFunctions.isEmailValid(emailAddress.getText().toString(
)))                  Toast.makeText(this, "Wrong email format.",
Toast.LENGTH_SHORT).show();             else             {
firebaseAuth.sendPasswordResetEmail(emailAddress.getText().toStri
ng().trim()).addOnCompleteListener(new OnCompleteListener<Void>()
{                     @Override                     public void
onComplete(@NonNull Task<Void> task) {
if(task.isSuccessful())                          {
Toast.makeText(ResetPasswordActivity.this, "Password was reset.
Check your email.", Toast.LENGTH_LONG).show();
startActivity(new Intent(ResetPasswordActivity.this,
LoginActivity.class));                                 finish();
}                          else
Toast.makeText(ResetPasswordActivity.this, "This email is not
registered yet.", Toast.LENGTH_LONG).show();
}                 });             }         });     } }

package com.example.gymscape.ui.main.calendar;  import
android.content.Intent; import android.os.Bundle; import
android.util.Log; import android.view.LayoutInflater; import
android.view.View; import android.view.ViewGroup; import
android.widget.Adapter; import android.widget.CalendarView;
import android.widget.TextView; import android.widget.Toast;
import androidx.annotation.NonNull; import
androidx.annotation.Nullable; import
androidx.fragment.app.Fragment; import
androidx.lifecycle.Observer; import
androidx.lifecycle.ViewModelProvider; import
androidx.recyclerview.widget.LinearLayoutManager; import
androidx.recyclerview.widget.RecyclerView;  import
com.example.gymscape.Model.Exercise; import
com.example.gymscape.Model.Workout; import
com.example.gymscape.R; import
com.example.gymscape.SharedFunctions; import
com.example.gymscape.ui.UsedEnums; import
com.example.gymscape.ui.editworkout.EditWorkoutActivity; import
com.example.gymscape.ui.main.workout.WorkoutAdapter; import
com.example.gymscape.ui.main.workout.WorkoutViewModel;  import
```

```
java.text.SimpleDateFormat; import java.util.ArrayList; import
java.util.Calendar; import java.util.Date; import
java.util.GregorianCalendar; import java.util.Locale;  public
class CalendarFragment extends Fragment implements
WorkoutAdapter.OnListItemClickListener {      private
CalendarViewModel viewModel;      CalendarView calendarView;
RecyclerView recyclerView;     WorkoutAdapter adapter;
TextView textView;      int chosenDateInt;
ArrayList<Workout> allWorkouts = new ArrayList<>();      public
View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
viewModel = new ViewModelProvider(this,
ViewModelProvider.AndroidViewModelFactory.getInstance(getActivity
().getApplication())).get(CalendarViewModel.class);          Date
c = Calendar.getInstance().getTime();         chosenDateInt =
SharedFunctions.getDate(c.getTime());          View root =
inflater.inflate(R.layout.fragment_calendar, container, false);
calendarView = root.findViewById(R.id.calendarView);
recyclerView = root.findViewById(R.id.calendarRecycleView);
textView = root.findViewById(R.id.text_calendar);
recyclerView.hasFixedSize();
recyclerView.setLayoutManager(new
LinearLayoutManager(getActivity()));          adapter = new
WorkoutAdapter(allWorkouts, this);
recyclerView.setAdapter(adapter);
calendarView.setOnDateChangeListener(new
CalendarView.OnDateChangeListener() {             @Override
public void onSelectedDayChange(@NonNull CalendarView view, int
year, int month, int dayOfMonth) {                 Date
chosenDate = new GregorianCalendar(year, month,
dayOfMonth).getTime();                  chosenDateInt =
SharedFunctions.getDate(chosenDate.getTime());
Log.i("Date", "" + chosenDateInt);
notifyDateChange();             }         });
notifyDateChange();          return root;     }      @Override
public void onListItemClick(int clickedItemIndex) {
Intent toWorkout = new Intent(getContext(),
EditWorkoutActivity.class);
toWorkout.putExtra(UsedEnums.WORKOUT.toString(),
adapter.workoutList.get(clickedItemIndex));
startActivity(toWorkout);     }      private void
notifyDateChange()     {
viewModel.getAllWorkouts().observe(getActivity(), workouts -> {
allWorkouts.clear();             for(Workout w : workouts)
{                 if(w.getDate() == chosenDateInt) {
allWorkouts.add(w);                   }             }
adapter.notifyDataSetChanged();         });
viewModel.setExerciseText(allWorkouts.isEmpty());
viewModel.getExerciseText().observe(getViewLifecycleOwner(),
exerciseText -> {             if(exerciseText)
textView.setText("No workouts for this date.");             else
textView.setText("");         });     } }                });
```

```
package com.example.gymscape.ui.main.calendar;  import
android.app.Application;  import
androidx.lifecycle.AndroidViewModel; import
androidx.lifecycle.LiveData; import
androidx.lifecycle.MutableLiveData; import
androidx.lifecycle.ViewModel;  import
com.example.gymscape.Model.Workout; import
com.example.gymscape.architecture.shared.ExerciseRepository;
import java.util.ArrayList; import java.util.List;  public class
CalendarViewModel extends AndroidViewModel {      private
ExerciseRepository repository;     MutableLiveData<Boolean>
noExercises;      public CalendarViewModel(Application app) {
super(app);         repository =
ExerciseRepository.getInstance(app);         noExercises = new
MutableLiveData<>();         noExercises.setValue(false);     }
public LiveData<Boolean> getExerciseText()     {          return
noExercises;     }      public void setExerciseText(boolean
exerciseText)     {          noExercises.setValue(exerciseText);
}       public LiveData<List<Workout>> getAllWorkouts()     {
return repository.getAllWorkouts();     }      public
List<Workout> getAllWorkoutsList()     {          return
repository.getAllWorkouts().getValue();     } }           }
});     } }
```

```
package com.example.gymscape.ui.main.exercises;  import
android.os.Bundle; import android.view.LayoutInflater; import
android.view.View; import android.view.ViewGroup;  import
androidx.annotation.NonNull; import
androidx.fragment.app.Fragment; import
androidx.lifecycle.ViewModelProvider;  import
com.example.gymscape.R;  public class ExercisesFragment extends
Fragment {      private ExercisesViewModel exercisesViewModel;
public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
exercisesViewModel =                  new
ViewModelProvider(this).get(ExercisesViewModel.class);
View root = inflater.inflate(R.layout.fragment_exercises,
container, false);         /*final TextView textView =
root.findViewById(R.id.text_exercises);
exercisesViewModel.getText().observe(getViewLifecycleOwner(), new
Observer<String>() {             @Override             public
void onChanged(@Nullable String s) {
textView.setText(s);             }         });*/          return
root;     }   }
```

```
package com.example.gymscape.ui.main.exercises;  import
androidx.lifecycle.LiveData; import
androidx.lifecycle.MutableLiveData; import
androidx.lifecycle.ViewModel;  public class ExercisesViewModel
extends ViewModel {      private MutableLiveData<String> mText;
public ExercisesViewModel() {         /* mText = new
MutableLiveData<>();         mText.setValue("This is exercises
```

```
fragment");*/     }      public LiveData<String> getText() {
return mText;     } }

package com.example.gymscape.ui.main.profile;  import
android.app.AlertDialog; import android.content.Context; import
android.content.DialogInterface; import android.content.Intent;
import android.content.SharedPreferences; import
android.os.Bundle; import android.util.Log; import
android.view.LayoutInflater; import android.view.View; import
android.view.ViewGroup; import android.widget.TextView; import
android.widget.Toast;  import androidx.annotation.NonNull; import
androidx.constraintlayout.widget.ConstraintLayout; import
androidx.fragment.app.Fragment; import
androidx.lifecycle.ViewModelProvider;  import
com.example.gymscape.R; import
com.example.gymscape.ui.login.LoginActivity; import
com.google.firebase.auth.FirebaseAuth;  public class
ProfileFragment extends Fragment {      private ProfileViewModel
viewModel;      ConstraintLayout textLogout, deleteAll;
TextView textUsername;      FirebaseAuth firebaseAuth;
public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
firebaseAuth = FirebaseAuth.getInstance();         viewModel =
new ViewModelProvider(this,
ViewModelProvider.AndroidViewModelFactory.getInstance(getActivity
().getApplication())).get(ProfileViewModel.class);         View
root = inflater.inflate(R.layout.fragment_profile, container,
false);          textUsername =
root.findViewById(R.id.textUsername);
textUsername.setText(firebaseAuth.getCurrentUser().getEmail().toL
owerCase());          textLogout =
root.findViewById(R.id.logoutLayout);
textLogout.setOnClickListener(v -> {
firebaseAuth.signOut();             startActivity(new
Intent(getContext(), LoginActivity.class));
getActivity().finish();         });          deleteAll =
root.findViewById(R.id.deleteAllLayout);
deleteAll.setOnClickListener(v -> {
DialogInterface.OnClickListener dialogClickListener = new
DialogInterface.OnClickListener() {                 @Override
public void onClick(DialogInterface dialog, int which) {
switch (which){                         case
DialogInterface.BUTTON_POSITIVE:
viewModel.deleteAll();
Toast.makeText(getActivity(), "All data deleted.",
Toast.LENGTH_SHORT).show();                             break;
case DialogInterface.BUTTON_NEGATIVE:
// do nothing                           break;
}                 }             };
AlertDialog.Builder builder = new
AlertDialog.Builder(getContext());
builder.setMessage("All saved workouts and exercises created by
```

```
you will be deleted. Do you want to
continue?").setPositiveButton("Yes", dialogClickListener)
.setNegativeButton("No", dialogClickListener).show();         });
return root;     } }

package com.example.gymscape.ui.main.profile;  import
android.app.Application;  import
androidx.lifecycle.AndroidViewModel; import
androidx.lifecycle.LiveData; import
androidx.lifecycle.MutableLiveData; import
androidx.lifecycle.ViewModel; import
com.example.gymscape.architecture.shared.ExerciseRepository;
public class ProfileViewModel extends AndroidViewModel {
ExerciseRepository repository;      public
ProfileViewModel(Application application)     {
super(application);         repository =
ExerciseRepository.getInstance(application);     }      public
void deleteAll()     {         repository.deleteAllExercises();
repository.deleteAllWorkouts();     } }

package com.example.gymscape.ui.main.workout;  import
android.view.LayoutInflater; import android.view.View; import
android.view.ViewGroup; import android.widget.ImageView; import
android.widget.TextView;  import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;  import
com.example.gymscape.Model.Exercise; import
com.example.gymscape.Model.Workout; import
com.example.gymscape.R;  import java.util.ArrayList;  public
class WorkoutAdapter extends
RecyclerView.Adapter<WorkoutAdapter.ViewHolder> {      public
ArrayList<Workout> workoutList;     final private
OnListItemClickListener mOnListItemClickListener;      public
WorkoutAdapter(ArrayList<Workout> workouts,
OnListItemClickListener listener) {         workoutList =
workouts;         mOnListItemClickListener = listener;     }
@NonNull     public ViewHolder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {         LayoutInflater inflater
= LayoutInflater.from(parent.getContext());         View view =
inflater.inflate(R.layout.workout_list_item, parent, false);
return new ViewHolder(view);     }      public void
onBindViewHolder(@NonNull ViewHolder viewHolder, int position) {
viewHolder.name.setText(workoutList.get(position).getExerciseName
());         switch (workoutList.get(position).getCategory()) {
case 1:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_core);
break;             case 2:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_chest);
break;             case 3:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_back);
break;             case 4:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_biceps);
break;             case 5:
```

```
viewHolder.iconMuscle.setImageResource(R.drawable.icon_triceps);
break;             case 6:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_shoulder);
break;             case 7:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_legs);
break;             case 8:
viewHolder.iconMuscle.setImageResource(R.drawable.icon_glutes);
break;         }         if(workoutList.get(position).getSets()
== 0 || workoutList.get(position).getWeight() == 0)
viewHolder.information.setText("");         else
viewHolder.information.setText(workoutList.get(position).getSets(
) + " sets | weight: " + workoutList.get(position).getWeight() +
" kgs");     }      public int getItemCount() {         return
workoutList.size();     }      class ViewHolder extends
RecyclerView.ViewHolder implements View.OnClickListener {
TextView name;         ImageView iconMuscle;         TextView
information;          ViewHolder(View itemView) {
super(itemView);             name =
itemView.findViewById(R.id.exerciseName);             iconMuscle
= itemView.findViewById(R.id.iconMuscle);             information
= itemView.findViewById(R.id.informationTextView);
itemView.setOnClickListener(this);         }          @Override
public void onClick(View v) {
mOnListItemClickListener.onListItemClick(getAdapterPosition());
}     }      public interface OnListItemClickListener {
void onListItemClick(int clickedItemIndex);     } }

package com.example.gymscape.ui.main.workout;  import
android.content.Intent; import android.os.Bundle; import
android.util.Log; import android.view.LayoutInflater; import
android.view.View; import android.view.ViewGroup; import
android.widget.TextView;  import androidx.annotation.NonNull;
import androidx.annotation.Nullable; import
androidx.fragment.app.Fragment; import
androidx.lifecycle.Observer; import
androidx.lifecycle.ViewModelProvider; import
androidx.recyclerview.widget.LinearLayoutManager; import
androidx.recyclerview.widget.RecyclerView;  import
com.example.gymscape.Model.Workout; import
com.example.gymscape.R; import com.example.gymscape.ui.UsedEnums;
import com.example.gymscape.ui.editworkout.EditWorkoutActivity;
import com.example.gymscape.ui.exercise.SpecificExerciseActivity;
import com.example.gymscape.ui.newexercise.NewExerciseViewModel;
import com.example.gymscape.ui.newworkout.NewWorkoutActivity;
import java.text.SimpleDateFormat; import
java.time.LocalDateTime; import
java.time.format.DateTimeFormatter; import java.util.ArrayList;
import java.util.Calendar; import java.util.Date; import
java.util.Locale;  public class WorkoutFragment extends Fragment
implements WorkoutAdapter.OnListItemClickListener {      private
WorkoutViewModel viewModel;      RecyclerView recyclerView;
WorkoutAdapter adapter;     TextView textView;      int
```

```
currentDate;      ArrayList<Workout> allWorkouts = new
ArrayList<>();      public View onCreateView(@NonNull
LayoutInflater inflater,                                ViewGroup
container, Bundle savedInstanceState) {          View root =
inflater.inflate(R.layout.fragment_workout, container, false);
textView = root.findViewById(R.id.text_workout);
viewModel = new ViewModelProvider(this,
ViewModelProvider.AndroidViewModelFactory.getInstance(getActivity
().getApplication())).get(WorkoutViewModel.class);          Date
c = Calendar.getInstance().getTime();         currentDate =
getDate(c.getTime());          recyclerView =
root.findViewById(R.id.workoutRecycleView);
recyclerView.hasFixedSize();
recyclerView.setLayoutManager(new
LinearLayoutManager(getActivity()));          adapter = new
WorkoutAdapter(allWorkouts, this);
recyclerView.setAdapter(adapter);
viewModel.getAllWorkouts().observe(getActivity(), workouts -> {
allWorkouts.clear();             for(Workout w : workouts)
{                 if(w.getDate() == currentDate) {
allWorkouts.add(w);
viewModel.setIsWorkout(true);                 }            }
adapter.notifyDataSetChanged();
if(allWorkouts.isEmpty())
viewModel.setIsWorkout(false);         });
viewModel.isWorkout().observe(getActivity(), isWorkout ->{
Log.i("Is workout?", isWorkout+"");             if(!isWorkout)
textView.setText("No planned workouts for today.");
else                textView.setText("");         });
return root;     }      @Override     public void
onListItemClick(int clickedItemIndex) {         Intent toWorkout
= new Intent(getContext(), EditWorkoutActivity.class);
toWorkout.putExtra(UsedEnums.WORKOUT.toString(),
adapter.workoutList.get(clickedItemIndex));
startActivity(toWorkout);     }      private int getDate(long
date)     {         Date date1 = new Date(date);         String
myFormat = "ddMMyyyy";         SimpleDateFormat dateFormat = new
SimpleDateFormat(myFormat, Locale.UK);         String
desiredFormat = dateFormat.format(date1);         return
Integer.parseInt(desiredFormat);     } }

package com.example.gymscape.ui.main.workout;  import
android.app.Application;  import
androidx.lifecycle.AndroidViewModel; import
androidx.lifecycle.LiveData; import
androidx.lifecycle.MutableLiveData; import
androidx.lifecycle.ViewModel;  import
com.example.gymscape.Model.Workout; import
com.example.gymscape.architecture.shared.ExerciseRepository;
import java.util.List;  public class WorkoutViewModel extends
AndroidViewModel {      private ExerciseRepository repository;
MutableLiveData<Boolean> isWorkout;      public
```

```
WorkoutViewModel(Application app) {         super(app);
repository = ExerciseRepository.getInstance(app);
isWorkout = new MutableLiveData<>();
isWorkout.setValue(false);     }      public void
setIsWorkout(boolean isWorkout) {
this.isWorkout.setValue(isWorkout);     }      public
LiveData<Boolean> isWorkout() {         return isWorkout;     }
public LiveData<List<Workout>> getAllWorkouts()     {
return repository.getAllWorkouts();     } }

package com.example.gymscape.ui.newexercise;  import
androidx.annotation.Nullable; import
androidx.appcompat.app.AppCompatActivity; import
androidx.core.content.ContextCompat; import
androidx.lifecycle.ViewModelProvider;  import
android.app.Activity; import android.content.Intent; import
android.content.pm.ActivityInfo; import android.graphics.Bitmap;
import android.graphics.drawable.ColorDrawable; import
android.os.Bundle; import android.os.Environment; import
android.provider.MediaStore; import android.util.Log; import
android.view.View; import android.widget.Button; import
android.widget.EditText; import android.widget.ImageView; import
android.widget.Toast;  import
com.example.gymscape.Model.Exercise; import
com.example.gymscape.R; import
com.example.gymscape.ui.MainActivity; import
com.example.gymscape.ui.UsedEnums; import
com.example.gymscape.ui.exercise.SpecificExerciseViewModel;
import com.example.gymscape.ui.exerciselist.ExerciseActivity;
import java.io.File; import java.io.FileNotFoundException; import
java.io.FileOutputStream; import java.io.IOException; import
java.text.SimpleDateFormat; import java.util.Date;  public class
NewExerciseActivity extends AppCompatActivity {      private
NewExerciseViewModel viewModel;      EditText nameField;
EditText descriptionField;     Button selectPhoto;     Button
cancelExercise;     ImageView previewImage;      int category;
File mediaFile;     final int CAMERA_REQUEST = 1;      @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_new_exercise);
viewModel = new ViewModelProvider(this,
ViewModelProvider.AndroidViewModelFactory.getInstance(getApplicat
ion())).get(NewExerciseViewModel.class);
this.getSupportActionBar().setTitle("Add Exercise");
this.getSupportActionBar().setBackgroundDrawable(new
ColorDrawable(ContextCompat.getColor(this, R.color.darkBlue)));
Intent intent = getIntent();         category =
intent.getIntExtra(UsedEnums.CATEGORY.toString(), 0);
nameField = findViewById(R.id.nameEditText);
descriptionField = findViewById(R.id.descriptionEditText);
selectPhoto = findViewById(R.id.imageButton);
cancelExercise = findViewById(R.id.cancelButton);
```

```
previewImage = findViewById(R.id.previewImage);
cancelExercise.setOnClickListener(v -> {             Intent
backIntent = new Intent(this, ExerciseActivity.class);
backIntent.putExtra(UsedEnums.CATEGORY.toString(), category);
startActivity(backIntent);             finish();         });
selectPhoto.setOnClickListener(v -> {             //@TODO: full
resolution picture             Intent cameraIntent = new
Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(cameraIntent, CAMERA_REQUEST);         });
}      public void saveExercise(View v)     {
if(nameField.getText().toString().length() > 20)
Toast.makeText(this, "Exercise name is too long.",
Toast.LENGTH_SHORT).show();         else
if(nameField.getText().toString().isEmpty() ||
descriptionField.getText().toString().isEmpty())
Toast.makeText(this, "Description or/and exercise name are
empty.", Toast.LENGTH_SHORT).show();         else
if(!viewModel.getPictureTaken().getValue())
Toast.makeText(this, "You must take a picture of exercise.",
Toast.LENGTH_SHORT).show();         else {
storeImage(viewModel.getExercisePhoto().getValue());
viewModel.insert(new Exercise(nameField.getText().toString(),
category, descriptionField.getText().toString(),
mediaFile.getAbsolutePath(), true));             Intent intent =
new Intent(this, ExerciseActivity.class);
intent.putExtra(UsedEnums.CATEGORY.toString(), category);
startActivity(intent);             finish();         }     }
@Override     protected void onActivityResult(int requestCode,
int resultCode, @Nullable Intent data) {
super.onActivityResult(requestCode, resultCode, data);
if (requestCode == CAMERA_REQUEST && resultCode ==
Activity.RESULT_OK)         {
viewModel.setPictureTaken(true);
viewModel.setExercisePhoto((Bitmap)
data.getExtras().get("data"));
previewImage.setImageBitmap(viewModel.getExercisePhoto().getValue
());         }         else
viewModel.setPictureTaken(false);     }      private void
storeImage(Bitmap image) {         File pictureFile =
getOutputMediaFile();         if (pictureFile == null) {
Log.d("Image Save", "Error creating media file, check storage
permissions: ");             return;         }         try {
FileOutputStream fos = new FileOutputStream(pictureFile);
image.compress(Bitmap.CompressFormat.PNG, 90, fos);
fos.close();         } catch (FileNotFoundException e) {
Log.d("Image Save", "File not found: " + e.getMessage());
} catch (IOException e) {             Log.d("Image Save", "Error
accessing file: " + e.getMessage());         }     }      private
File getOutputMediaFile(){         File mediaStorageDir = new
File(Environment.getExternalStorageDirectory()                 +
"/Android/data/"                 +
getApplicationContext().getPackageName()                 +
```

```
"/Files");         // Create the storage directory if it does not
exist         if (! mediaStorageDir.exists()){             if (!
mediaStorageDir.mkdirs()){                 return null;
}         }         // Create a media file name
(EXERCISE_ddMMyyy_HHmmss.jpg)         String timeStamp = new
SimpleDateFormat("ddMMyyyy_HHmmss").format(new Date());
String mImageName="EXERCISE_"+ timeStamp +".jpg";
mediaFile = new File(mediaStorageDir.getPath() + File.separator +
mImageName);         return mediaFile;     }      @Override
protected void onResume() {         super.onResume();
previewImage.setImageBitmap(viewModel.getExercisePhoto().getValue
());     } }

package com.example.gymscape.ui.newexercise;  import
android.app.Application; import android.graphics.Bitmap;  import
androidx.annotation.NonNull; import
androidx.lifecycle.AndroidViewModel; import
androidx.lifecycle.LiveData; import
androidx.lifecycle.MutableLiveData;  import
com.example.gymscape.Model.Exercise; import
com.example.gymscape.architecture.shared.ExerciseRepository;
public class NewExerciseViewModel extends AndroidViewModel {
private final ExerciseRepository repository;
MutableLiveData<Bitmap> exercisePhoto;
MutableLiveData<Boolean> pictureTaken;      public
NewExerciseViewModel(@NonNull Application application) {
super(application);         repository =
ExerciseRepository.getInstance(application);
exercisePhoto = new MutableLiveData<>();         pictureTaken =
new MutableLiveData<>(false);     }      public void insert(final
Exercise exercise)     {         repository.insert(exercise);
}      public LiveData<Bitmap> getExercisePhoto() {
return exercisePhoto;     }      public void
setExercisePhoto(Bitmap exercisePhoto) {
this.exercisePhoto.setValue(exercisePhoto);     }      public
LiveData<Boolean> getPictureTaken() {         return
pictureTaken;     }      public void setPictureTaken(boolean
pictureTaken) {         this.pictureTaken.setValue(pictureTaken);
} }

package com.example.gymscape.ui.newworkout;  import
androidx.appcompat.app.AppCompatActivity; import
androidx.core.content.ContextCompat; import
androidx.lifecycle.Observer; import
androidx.lifecycle.ViewModelProvider;  import
android.app.DatePickerDialog; import android.content.Intent;
import android.graphics.drawable.ColorDrawable; import
android.os.Bundle; import android.util.Log; import
android.view.View; import android.widget.Button; import
android.widget.DatePicker; import android.widget.EditText; import
android.widget.ImageView; import android.widget.TextView; import
android.widget.Toast;  import com.bumptech.glide.Glide; import
```

```
com.example.gymscape.Model.Exercise; import
com.example.gymscape.Model.Workout; import
com.example.gymscape.R; import
com.example.gymscape.SharedFunctions; import
com.example.gymscape.ui.MainActivity; import
com.example.gymscape.ui.UsedEnums; import
com.example.gymscape.ui.exercise.SpecificExerciseActivity; import
com.example.gymscape.ui.exercise.SpecificExerciseViewModel;
import com.example.gymscape.ui.exerciselist.ExerciseActivity;
import com.example.gymscape.ui.main.calendar.CalendarViewModel;
import com.example.gymscape.ui.newexercise.NewExerciseViewModel;
import java.sql.Date; import java.text.SimpleDateFormat; import
java.util.Calendar; import java.util.Locale;  public class
NewWorkoutActivity extends AppCompatActivity {      private
NewWorkoutViewModel viewModel;      EditText datePicker;
EditText weightTextNumber;     Calendar calendar;     Exercise
exercise;     ImageView exerciseImageCategory;     TextView
exerciseName;     Button increaseSets;     Button decreaseSets;
Button saveWorkoutButton;     Button cancelWorkoutButton;
TextView setsCountTextView;      Workout newWorkout;
@Override     protected void onCreate(Bundle savedInstanceState)
{         super.onCreate(savedInstanceState);
setContentView(R.layout.activity_new_workout);          viewModel
= new ViewModelProvider(this,
ViewModelProvider.AndroidViewModelFactory.getInstance(getApplicat
ion())).get(NewWorkoutViewModel.class);
this.getSupportActionBar().setTitle("Add exercise to workout");
this.getSupportActionBar().setBackgroundDrawable(new
ColorDrawable(ContextCompat.getColor(this, R.color.darkBlue)));
Intent intent = getIntent();         exercise = (Exercise)
intent.getSerializableExtra(UsedEnums.EXERCISE.toString());
exerciseImageCategory = findViewById(R.id.exerciseImageCategory);
exerciseName = findViewById(R.id.exerciseName);
weightTextNumber = findViewById(R.id.weightTextNumber);
increaseSets = findViewById(R.id.increaseSetsButton);
decreaseSets = findViewById(R.id.decreaseSetsButton);
setsCountTextView = findViewById(R.id.setsCountTextView);
saveWorkoutButton = findViewById(R.id.saveWorkoutButton);
cancelWorkoutButton = findViewById(R.id.cancelWorkoutButton);
newWorkout = new Workout();
viewModel.setExercise(exercise);
viewModel.getExercise().observe(this, exerciseData -> {
/* Text can be too long to display it in actionbar
this.getSupportActionBar().setTitle("Add " +
exerciseData.getName() + " to workout");
this.getSupportActionBar().setBackgroundDrawable(new
ColorDrawable(ContextCompat.getColor(this, R.color.darkBlue)));
*/              exerciseName.setText(exerciseData.getName());
exerciseImageCategory.setImageResource(SharedFunctions.getIcon(ex
erciseData.getCategory()));
newWorkout.setExerciseName(exerciseData.getName());
newWorkout.setCategory(exerciseData.getCategory());         });
```

```
calendar = Calendar.getInstance();         datePicker =
findViewById(R.id.editTextDate);
DatePickerDialog.OnDateSetListener date = new
DatePickerDialog.OnDateSetListener() {             @Override
public void onDateSet(DatePicker view, int year, int month, int
dayOfMonth) {                 calendar.set(Calendar.YEAR, year);
calendar.set(Calendar.MONTH, month);
calendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
viewModel.setDate(calendar.getTime());             }         };
datePicker.setOnClickListener(v -> {             new
DatePickerDialog(this, date, calendar.get(Calendar.YEAR),
calendar.get(Calendar.MONTH),
calendar.get(Calendar.DAY_OF_MONTH)).show();         });
viewModel.getDate().observe(this, chosenDate -> {
String myFormat = "dd/MM/yyyy";             SimpleDateFormat
dateFormat = new SimpleDateFormat(myFormat, Locale.UK);
datePicker.setText(dateFormat.format(chosenDate));
newWorkout.setDate(SharedFunctions.getDate(chosenDate.getTime()))
;         });          decreaseSets.setOnClickListener(v -> {
if(viewModel.getSets().getValue() <= 0)
Toast.makeText(this, "Minimum value of sets is 0",
Toast.LENGTH_SHORT).show();             else             {
int setsValue =
Integer.parseInt(setsCountTextView.getText().toString());
viewModel.setSets(--setsValue);             }         });
increaseSets.setOnClickListener(v -> {
if(viewModel.getSets().getValue() >= 10)
Toast.makeText(this, "Maximum value of sets is 10",
Toast.LENGTH_SHORT).show();             else             {
int setsValue =
Integer.parseInt(setsCountTextView.getText().toString());
viewModel.setSets(++setsValue);             }         });
viewModel.getSets().observe(this, new Observer<Integer>() {
@Override             public void onChanged(Integer integer) {
setsCountTextView.setText("" + integer);
newWorkout.setSets(integer);             }         });
saveWorkoutButton.setOnClickListener(v -> {
if(newWorkout.getDate() <= 0)
Toast.makeText(this, "You must set date to continue.",
Toast.LENGTH_SHORT).show();             else {                 if
(weightTextNumber.getText().toString().equals(""))
newWorkout.setWeight(0);                 else
newWorkout.setWeight(Integer.parseInt(weightTextNumber.getText().
toString()));                 viewModel.insert(newWorkout);
Log.i("Date", newWorkout.getDate() + "");                 Intent
mainIntent = new Intent(this, MainActivity.class);
startActivity(mainIntent);                 finish();
}         });          cancelWorkoutButton.setOnClickListener(v -
> {             Intent exerciseIntent = new Intent(this,
SpecificExerciseActivity.class);
exerciseIntent.putExtra(UsedEnums.EXERCISE.toString(), exercise);
```

```
startActivity(exerciseIntent);             finish();         });
} }

package com.example.gymscape.ui.newworkout;  import
android.app.Application;  import
androidx.lifecycle.AndroidViewModel; import
androidx.lifecycle.LiveData; import
androidx.lifecycle.MutableLiveData;  import
com.example.gymscape.Model.Exercise; import
com.example.gymscape.Model.Workout; import
com.example.gymscape.architecture.shared.ExerciseRepository;
import java.util.Date;  public class NewWorkoutViewModel extends
AndroidViewModel {     MutableLiveData<Date> date;
MutableLiveData<Integer> sets;     MutableLiveData<Exercise>
exercise;     ExerciseRepository repository;      public
NewWorkoutViewModel(Application application)     {
super(application);         date = new MutableLiveData<>();
sets = new MutableLiveData<>();         sets.setValue(1);
exercise = new MutableLiveData<>();         repository =
ExerciseRepository.getInstance(application);     }      public
LiveData<Date> getDate() {         return date;     }      public
void setDate(Date date) {         this.date.setValue(date);     }
public LiveData<Integer> getSets() {         return sets;     }
public void setSets(int sets) {         this.sets.setValue(sets);
}      public LiveData<Exercise> getExercise() {         return
exercise;     }      public void setExercise(Exercise exercise) {
this.exercise.setValue(exercise);     }      public void
insert(final Workout workout)     {
repository.insertWorkout(workout);     } }
```

```
package com.example.gymscape.ui;  import android.content.Intent;
import android.os.Bundle; import android.view.View;  import
com.example.gymscape.R; import
com.example.gymscape.ui.exerciselist.ExerciseActivity; import
com.google.android.material.bottomnavigation.BottomNavigationView
;  import androidx.appcompat.app.AppCompatActivity; import
androidx.navigation.NavController; import
androidx.navigation.Navigation; import
androidx.navigation.ui.AppBarConfiguration; import
androidx.navigation.ui.NavigationUI;  public class MainActivity
extends AppCompatActivity {      @Override     protected void
onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
this.getSupportActionBar().hide();
setContentView(R.layout.activity_main);
BottomNavigationView navView = findViewById(R.id.nav_view);
// Passing each menu ID as a set of Ids because each         //
menu should be considered as top level destinations.
AppBarConfiguration appBarConfiguration = new
AppBarConfiguration.Builder(
R.id.navigation_exercises, R.id.navigation_workout,
R.id.navigation_calendar, R.id.navigation_profile)
.build();         NavController navController =
Navigation.findNavController(this, R.id.nav_host_fragment);
NavigationUI.setupActionBarWithNavController(this, navController,
appBarConfiguration);
NavigationUI.setupWithNavController(navView, navController);
}      public void openExercise(View view)     {
if(view.getId() == R.id.muscleCore)         {             Intent
intent = new Intent(this, ExerciseActivity.class);
intent.putExtra(UsedEnums.CATEGORY.toString(), 1);
startActivity(intent);         }         else if(view.getId() ==
R.id.muscleChest)         {             Intent intent = new
Intent(this, ExerciseActivity.class);
intent.putExtra(UsedEnums.CATEGORY.toString(), 2);
startActivity(intent);         }         else if(view.getId() ==
R.id.muscleBack)         {             Intent intent = new
Intent(this, ExerciseActivity.class);
intent.putExtra(UsedEnums.CATEGORY.toString(), 3);
startActivity(intent);         }         else if(view.getId() ==
R.id.muscleBiceps)         {             Intent intent = new
Intent(this, ExerciseActivity.class);
intent.putExtra(UsedEnums.CATEGORY.toString(), 4);
startActivity(intent);         }         else if(view.getId() ==
R.id.muscleTriceps)         {             Intent intent = new
Intent(this, ExerciseActivity.class);
intent.putExtra(UsedEnums.CATEGORY.toString(), 5);
startActivity(intent);         }         else if(view.getId() ==
R.id.muscleShoulders)         {             Intent intent = new
Intent(this, ExerciseActivity.class);
intent.putExtra(UsedEnums.CATEGORY.toString(), 6);
startActivity(intent);         }         else if(view.getId() ==
```

```
R.id.muscleLegs)         {             Intent intent = new
Intent(this, ExerciseActivity.class);
intent.putExtra(UsedEnums.CATEGORY.toString(), 7);
startActivity(intent);         }         else if(view.getId() ==
R.id.muscleGlutes)         {             Intent intent = new
Intent(this, ExerciseActivity.class);
intent.putExtra(UsedEnums.CATEGORY.toString(), 8);
startActivity(intent);         }         else         {
Intent intent = new Intent(this, ExerciseActivity.class);
intent.putExtra(UsedEnums.CATEGORY.toString(), 0);
startActivity(intent);         }     } }

package com.example.gymscape;  import com.example.gymscape.R;
import java.sql.Date; import java.text.SimpleDateFormat; import
java.util.Locale; import java.util.regex.Matcher; import
java.util.regex.Pattern;  public class SharedFunctions {
public static int getDate(long date)     {         Date date1 =
new Date(date);         String myFormat = "ddMMyyyy";
SimpleDateFormat dateFormat = new SimpleDateFormat(myFormat,
Locale.UK);         String desiredFormat =
dateFormat.format(date1);         return
Integer.parseInt(desiredFormat);     }      public static int
getIcon(int category)     {         int icon = 0;         switch
(category)         {             case 1:                 return
R.drawable.icon_core;             case 2:                 return
R.drawable.icon_chest;             case 3:                 return
R.drawable.icon_back;             case 4:                 return
R.drawable.icon_biceps;             case 5:
return R.drawable.icon_triceps;             case 6:
return R.drawable.icon_shoulder;             case 7:
return R.drawable.icon_legs;             case 8:
return R.drawable.icon_glutes;         }         return icon;
}      public static boolean isEmailValid(String email)     {
Pattern email_regex = Pattern.compile("^[A-Z0-9._%+-]+@[A-Z0-9.-
]+\\.[A-Z]{2,6}$", Pattern.CASE_INSENSITIVE);         Matcher
matcher = email_regex.matcher(email.trim());         return
matcher.find();     } }
```