МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами

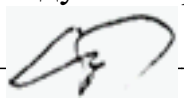
Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри



_ проф. Приходько С.Б.

«___» _____ 2025 р.

КВАЛІФІКАЦІЙНАРОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка мобільного застосунку під ОС Android для домашніх тренувань «TrainSphere»

Виконав: студент групи 4157ст

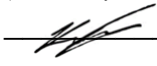
Піщиць Б.О.

(підпис, ПІБ)

Керівник роботи:

доцент, к.т.н

(посада, науковий ступень вчене звання)



Каіров В.О.

(підпис, ПІБ)

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«_08_» ___04___ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Піщицю Богдану Олександровичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка мобільного застосунку під ОС Android для домашніх тренувань «TrainSphere»

Керівник роботи Каіров Володимир Олексійович
Затверджені наказом ректора №153 від 08.04.2025 р.

2. Термін подання роботи: 02.06.2025 р.

3. Вихідні дані по роботі:_____

4. Перелік питань, що належать до розробки (найменування розділів)
Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів Титульний слайд, Мета та завдання кваліфікаційної роботи, Аналіз існуючих програмних рішень, Постановка задачі на розробку (Вимоги до складу виконуваних функцій), Аналіз вимог до мобільного застосунку (Діаграма варіантів використання), Архітектура мобільного застосунку (Діаграма компонентів та діаграма розгортання), Ескізний проєкт мобільного застосунку (Діаграми діяльності для варіантів використання «Переглянути вправи» та «Переглянути тренування), Ескізний проєкт мобільного застосунку (Діаграма діяльності для варіанту використання «Переглянути розклад»), Ескізний проєкт мобільного застосунку (Концептуальна модель), Технічний проєкт мобільного застосунку (Діаграма класів), Технічний проєкт мобільного застосунку (Діаграма послідовності для прецеденту «Переглянути вправи»), Технічний проєкт мобільного застосунку (Діаграма послідовності для прецеденту «Виконати тренування»), Технічний проєкт мобільного застосунку (Логічна модель даних), Робочий проєкт мобільного застосунку (Вибір та обґрунтування інструментів розробки мобільного застосунку), Результати розробки мобільного застосунку (Головне меню, меню авторизації та іконка мобільного застосунку), Результати розробки мобільного застосунку (Меню: налаштувань, груп м'язів, тренувань та інвентаря), Результати розробки мобільного застосунку (Меню: розкладу, вправ та виконання тренувань), Висновки, Заключний слайд.

6. Консультанти розділів роботи

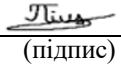
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

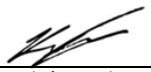
7. Дата видачі завдання 08.04.2025 р.

КАЛЕНДАРНИЙ ПЛАН

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	31.03.2025	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	07.04.2025	ПП*
3.	Аналіз вимог до ПЗ	14.04.2025	ПП*
4.	Розробка архітектури ПЗ	21.04.2025	
5.	Розробка проєкту ПЗ	05.05.2025	
6.	Реалізація та тестування ПЗ	12.05.2025	
7.	Підготовка розділу Результати розробки ПЗ	16.05.2025	
8.	Підготовка розділу з охорони праці	19.05.2025	ПП*
9.	Підготовка розділу ВИСНОВКИ	23.05.2025	
10.	Оформлення списку використаних джерел та додатків	26.05.2025	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	02.06.2025	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
12.	Підготовка презентації та доповіді	06.06.2025	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	09.06.2025	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	16.06.2025	

* - за результатами переддипломної практики (ПП), яка була з 03.02.2025 до 23.03.2025 р.

Студент  Піщиць Богдан Олександрович
(підпис) (ПІБ)

Керівник роботи  Каіров Володимир Олексійович
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота спрямована на розв’язання практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме на розробку мобільного застосунку під ОС Android для домашніх тренувань «TrainSphere».

Об’єктом роботи є процес розробки мобільного застосунку «TrainSphere», призначення якого полягає в тому, щоб допомогти користувачам створити ефективний план тренувань, який відповідає їхнім цілям і рівню підготовки. Мобільний застосунок також надасть користувачам інформацію про різні типи вправ, їхню ефективність і техніку виконання, інформацію про інвентар, групи м’язів і т.д.

В кваліфікаційній роботі проведено аналіз практичної задачі інженерії програмного забезпечення та аналіз існуючих аналогів, здійснено постановку задачі на розробку мобільного застосунку для домашніх тренувань «TrainSphere», виконано аналіз вимог до програмного забезпечення, розроблено ескізний, технічний та робочий проєкти програмного забезпечення, представлено результати розробки та розділ з охорони праці.

Кваліфікаційна робота викладена українською мовою на 123 сторінках машинописного тексту та містить: 20 рисунків, 33 таблиці, список використаних джерел із 23 найменувань та 5 додатків.

ABSTRACT

The qualification work is aimed at solving a practical software engineering problem characterized by complexity and uncertainty of conditions, namely, the development of an Android mobile application for home workouts called «TrainSphere».

The object of the work is the development process of the «TrainSphere» mobile application, which is designed to help users create effective workout plans tailored to their goals and fitness levels. The application will also provide users with information about different types of exercises, their effectiveness, and proper techniques, as well as details on equipment, muscle groups, and more.

The qualification work analyses the practical problem of software engineering and analyses existing analogues, sets the task of developing a mobile application for home training «TrainSphere», analyses software requirements, develops draft, technical and working drafts of software, presents the development results and a section on occupational safety.

The qualification work is written in Ukrainian, spans 123 typed pages, and includes: 20 figures, 33 tables, a list of 23 references, and 5 appendices.

ЗМІСТ

ВСТУП

Кваліфікаційна робота присвячена розв'язанню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці мобільного застосунку під ОС Android для домашніх тренувань «TrainSphere». Комплексність полягає в необхідності обробки значного обсягу даних про користувачів, вправи, розклад, інвентар. Необхідно забезпечувати персоналізований підбір тренувань з урахуванням статі, зросту та ваги користувача, мати зручний інтерфейс, який повинен бути інтуїтивно зрозумілий, підтримувати збереження прогресу та працювати в офлайн-режимі. Невизначеність зумовлена відсутністю єдиної ефективної методики тренувань, індивідуальними особливостями користувачів, швидкими змінами фітнес-трендів, різним рівнем фізичної підготовки та можливими технічними обмеженнями пристроїв.

Розглянувши та проаналізувавши аналогічні мобільні застосунки було виявлено ряд недоліків. Основними недоліками є те, що проаналізовані мобільні застосунки враховують загальні параметри, такі як вік, стать та вагу лише для статистики та оцінки власного прогресу, без детальної персоналізації, а план тренувань будується лише відштовхуючись від цілі користувача. Однак усі люди різні і ефективний план тренувань повинен враховувати більше факторів, повинен враховувати вік, вагу та стать, так як в побудові тренувального плану ці фактори відіграють важливу роль. Тому, виконавши аналіз практичної задачі та аналіз існуючих програмних рішень, було прийняте рішення розробити власне програмне забезпечення для домашніх тренувань «TrainSphere».

Метою даної кваліфікаційної роботи є розв'язання практичної задачі інженерії програмного забезпечення, а саме розробка якісного, зручного, багатофункціонального мобільного застосунку під ОС Android для домашніх тренувань «TrainSphere».

Для досягнення поставленої мети необхідно вирішити наступні задачі:

- провести аналіз практичної задачі інженерії програмного забезпечення;
- провести аналіз аналогічного існуючого ПЗ та обґрунтування необхідності розробки власного ПЗ;
- сформувати вимоги до розроблюваного продукту та розробити технічне завдання на розробку мобільного застосунку під ОС Android для домашніх тренувань «TrainSphere»;
- розробити архітектуру програмного забезпечення;
- провести розробку ескізного, технічного та робочого проєкту програмного забезпечення мобільного застосунку під ОС Android для домашніх тренувань «TrainSphere»;
- розробити документацію до програмного продукту.

Так як кваліфікаційна робота присвячена розробці ПЗ для домашніх тренувань «TrainSphere», то об'єктом даної роботи є власне процес розробки даного програмного забезпечення.

У результаті ми отримали програмний продукт, який допоможе користувачам створити ефективний план тренувань, який відповідає їхнім цілям і рівню підготовки.

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ МОБІЛЬНОГО ЗАСТОСУНКУ ПІД ОС ANDROID ДЛЯ ДОМАШНІХ ТРЕНУВАНЬ «TRAINSPHERE»

1.1 Аналіз практичної задачі інженерії програмного забезпечення мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

В даній роботі розв’язується практична задача інженерії програмного забезпечення, а саме розробка мобільного застосунку для домашніх тренувань під ОС Android. Дане ПЗ – інструмент для допомоги користувачам створити ефективний план тренувань, який відповідає їхнім цілям і рівню підготовки.

Доцільність фізичних тренувань полягає в їх спрямованості на досягнення конкретних фітнес-цілей та поліпшення загального фізичного стану. Основна ідея тренувань полягає в систематичному і цілеспрямованому навантаженні організму з метою зміцнення м'язів, покращення витривалості, збільшення гнучкості, покращення координації рухів та досягнення інших специфічних цілей.

Основні переваги фізичних тренувань включають:

- зміцнення м'язової системи: тренування сприяють зростанню м'язової маси, зміцненню та тонізації м'язів;
- покращення кардіоваскулярної системи: фізичні тренування сприяють зміцненню серцево-судинної системи, покращують кровообіг і підвищують витривалість кардіоваскулярної системи;
- загальне підвищення фізичної форми: тренування допомагають покращити рухову координацію, збільшити гнучкість, поліпшити реакцію та баланс;
- зниження ризику розвитку хвороб: систематичні тренування дають змогу мінімізувати загрозу появи та розвитку захворювань;

− підвищення настрою та самопочуття: фізичні вправи сприяють стимуляції синтезу ендорфінів – біологічних речовин, які підвищують настрій та знижують рівень стресу;

− сприяння здоровому сну: регулярна фізична активність сприяє вдосконаленню якості сну;

− контроль ваги та форми.

Тренування – це цілеспрямований процес формування ставлення, знань або поведінкових навичок у того, хто навчається, шляхом отримання відповідного досвіду з метою більш ефективного застосування здобутих знань і вмінь у конкретній сфері або виді діяльності.

Програма тренувань – це точний план дій для досягнення поставлених цілей. Від правильно підібраного комплексу вправ залежить кінцевий результат.

У сучасному світі людину оточує дуже багато різних обов’язків, тому економія часу дуже важлива.

Не завжди є можливість відвідувати фітнес-клуб, але це не причина відмовлятися від фізичної активності. В такому випадку чудовою альтернативою стають домашні тренування, які можуть мати різну тривалість і рівень навантаження.

Такий формат занять зараз надзвичайно популярний. По-перше, він не вимагає придбання абонемента до спортзалу чи навіть виходу з дому. По-друге, тренуватися можна у зручний для себе час. По-третє, для цього не потрібна спеціальна форма чи додатковий інвентар.

Домашні тренування здобули популярність завдяки дослідженням канадських науковців, які довели, що кілька хвилин інтенсивних м'язових вправ можуть бути такими ж ефективними, як кількагодинні заняття бігом чи їздою на велосипеді. Подібний підхід використовується в кросфіті, однак він вимагає присутності тренера, спеціального залу й обладнання, а саме тренування триває близько години і підходить лише фізично підготовленим людям. Натомість домашні програми доступні навіть для тих, чия єдина щоденна фізична активність – це прогулянка від ліфта до автомобіля.

Мобільні застосунки для контролю здоров'я стають все більш важливими в сучасному світі. Вони надають можливість людям відстежувати свої фізичні показники, активність, харчування та інші аспекти, що допомагають керувати своїм здоров'ям.

У багатьох людей є обмеження, які ускладнюють доступ до фітнес-центрів або регулярного відвідування тренера. Не у всіх є достатньо часу, ресурсів або можливостей займатися тренуваннями у зручний для них час. В таких випадках мати на своєму телефоні мобільний застосунок, який допомагає тренуватися, може бути великою перевагою.

Мобільні застосунки для тренувань надають гнучкість і можливість пристосувати тренування до свого власного графіка і потреб. Вони дозволяють займатися тренуваннями в будь-якому зручному місці і часі, не обмежуючи себе годинами роботи фітнес-центрів або тренерів. Ви можете вибрати тренування, яке відповідає вашим потребам і цілям, і проводити їх у комфортному для вас темпі.

Крім того, володіння мобільним застосунком для тренувань може бути економічно вигідним. Замість платити за членство в фітнес-центрі або витрачати гроші на тренера, є можливість використовувати безкоштовні або доступні мобільні застосунки для отримання якісної програми тренувань та підтримки.

Такі мобільні застосунки також можуть надати більш широкий спектр тренувань, включаючи різні стилі, інтенсивність і цілі. Користувач може обрати тренування, яке йому до вподоби, використовуючи відео-уроки, плани тренувань та інші ресурси, доступні у мобільному застосунку. Вищезазначені причини і є основними причинами для створення мобільного застосунку з тренуваннями, який буде завжди знаходитись у користувача під рукою, та який допоможе йому побудувати гарне тіло у будь-який момент.

Мобільний застосунок для домашніх тренувань під ОС Android буде прикладним програмним забезпеченням, яке буде розроблене з метою покращення фізичної форми користувача. Цей мобільний застосунок буде інструментом, який надає точну програму тренувань і допомагає користувачу досягти своїх цілей фізичної підготовки.

Мобільний застосунок для тренувань на базі ОС Android буде надавати різні комплекси вправ, підібрані залежно від бажаного результату та індивідуальних характеристик користувача. Він може містити в собі різноманітні вправи для різних груп м'язів, кардіотренування, розтяжки та інші фізичні активності. Користувач буде мати змогу обрати програму тренувань, яка відповідає його потребам і можливостям.

Оскільки застосунок буде доступний на мобільних пристроях під ОС Android, він завжди буде знаходитись під рукою користувача. Це буде дозволяти людям займатися фізичними тренуваннями в будь-який час, який зручний для користувача і зручному місці, навіть без відвідування спортзалу чи фітнес-клубу. Користувачі зможуть зосередитись на ефективному виконанні вправ за допомогою програми тренувань, без необхідності великих витрат часу та додаткового спортивного обладнання.

Все це робить мобільний застосунок для тренувань під ОС Android популярним і зручним інструментом для фізичної підготовки. Користувачі зможуть планувати та виконувати тренування в будь-який момент, відповідно до свого графіку і потреб.

Виходячи з цього, формуються прикладна задача інженерії програмного забезпечення – розробка мобільного застосунку для домашніх тренувань під операційну систему Android, який дозволяє користувачам формувати та проходити індивідуальні програми тренувань відповідно до їхніх цілей, фізичних можливостей та розкладу.

Мобільний застосунок під ОС Android для домашніх тренувань «Trainsphere» повинен забезпечувати:

- формування персоналізованих тренувальних планів;
- демонстрацію вправ (у вигляді відео, анімацій або текстових інструкцій);
- відстеження прогресу користувача та надання рекомендацій;
- можливість вибору рівня складності та типу тренування;
- зберігання даних та доступ до історії тренувань.

Для реалізації даного мобільного застосунку планується використання наступних технологій:

- Unity – як основне середовище розробки, що дозволяє створювати кросплатформені мобільні застосунки з гнучким інтерфейсом і анімацією;
- C# (C-Sharp) – мова програмування для реалізації логіки мобільного застосунку;
- Firebase – як хмарна платформа для зберігання даних користувачів, автентифікації, аналітики та забезпечення синхронізації між пристроями;
- підтримка Android-платформи через Unity Android Build Support.

1.2 Аналіз існуючих програмних рішень

Ринок містить низку програмних продуктів, призначених для тренувань. Розглянемо деякі з них.

1) Програмний продукт «Бібліотека вправ»

«Бібліотека вправ» – це безкоштовний мобільний застосунок під Android для тренувань [1]. Інтерфейс мобільного застосунку представлений на рисунку 1.1. Розробник FitTech Solutions позиціонує мобільний застосунок як довідник вправ, орієнтований на широке коло користувачів.

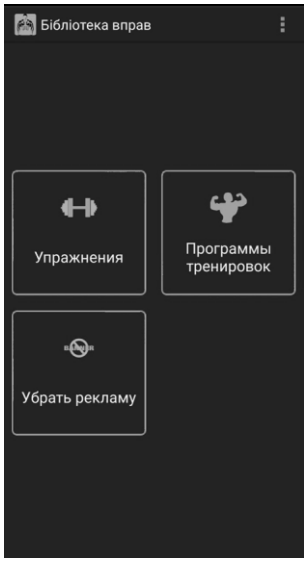


Рисунок 1.1 – Інтерфейс мобільного застосунку «Бібліотека вправ» [1]

Переваги:

- зручний в використанні;
- зрозумілий інтерфейс;
- працює без створення свого акаунту;
- детальні пояснення вправ;
- можливість використання без підключення до інтернету.

Недоліки:

- містить рекламу;
- мобільний застосунок не персоналізує програму тренувань під користувача, а просто є довідником;
- без можливості змінювати колір дизайну;
- немає інтеграції з фітнес-трекерами чи розумними годинниками;
- відсутня можливість імпорту власних вправ;
- не можна створювати або редагувати власні програми тренувань;
- відсутня синхронізація з іншими пристроями.

2) Програмний продукт «Тітан»

«Тітан» – це платний мобільний застосунок під Android зі списком всіх вправ [2]. Інтерфейс мобільного застосунку представлений на рисунку 1.2.

Рисунок 1.2 – Інтерфейс мобільного застосунку «Тітан» [2]

Розробник: TitanFit Labs

Основні функціональні можливості:

- список усіх тренувальних вправ із детальним поясненням;
- класифікація вправ за м’язовими групами та рівнем складності;
- відеоуроки для виконання вправ;
- вбудований таймер та лічильник підходів;
- нагадування про тренування.

Переваги:

- зручний в використанні;
- зрозумілий інтерфейс;
- детальні пояснення вправ.

Недоліки:

- не працює без створення свого акаунту;
- платний;
- без можливості змінювати колір дизайну;
- відсутня інтеграція з фітнес-трекерами та розумними годинниками;
- немає персоналізованих тренувальних програм;
- вимагає стабільного інтернет-з'єднання для доступу до деяких функцій.

3) Програмний продукт «Freeletics»

«Freeletics» – це популярний мобільний фітнес-застосунок під Android та iOS, орієнтований на тренування з вагою власного тіла [3]. Інтерфейс мобільного застосунку представлений на рисунку 1.3.

Розробник Freeletics GmbH позиціонує мобільний застосунок як інноваційний фітнес-тренер із використанням штучного інтелекту, що дозволяє тренуватися без необхідності в спортзалі.

Рисунок 1.3 – Інтерфейс мобільного застосунку «Freeletics» [3]

Функції:

- персоналізовані тренувальні програми на основі штучного інтелекту;
- вправи з вагою власного тіла без необхідності додаткового обладнання;
- відео-інструкції до кожної вправи;
- відстеження прогресу та аналітика тренувань;
- синхронізація з Google Fit та Apple Health.

Переваги:

- гнучкі програми для різного рівня фізичної підготовки;
- відео-інструкції сприяють правильному виконанню вправ.;
- відсутність необхідності у тренажерному залі або додатковому обладнанні;
- доступна безкоштовна версія з обмеженим функціоналом.

Недоліки:

- повний доступ до персоналізованих програм лише у платній версії;
- вимагає стабільного інтернет-з’єднання для перегляду відеоуроків;
- відсутня можливість імпорту власних вправ;
- обмежений набір вправ у безкоштовній версії;
- інколи пропоновані тренування можуть бути занадто інтенсивними для новачків;

− деякі вправи не адаптовані для людей із травмами чи фізичними обмеженнями.

Розглянувши існуючі програмні рішення виникає необхідність в розробці мобільного застосунку для домашніх тренувань «TrainSphere». Тому що всі люди різні і ефективний план тренувань повинен враховувати більше факторів, повинен враховувати вік, вагу та стать, так як для побудови тренувального плану ці фактори відіграють важливу роль, в цьому і виникає необхідність розробки мобільного застосунку «TrainSphere», який буде враховувати ці всі аспекти. Мобільний застосунок буде мати спеціалізовані програми тренувань для дітей, літніх людей та осіб з надлишковою вагою. Алгоритми будуть змінювати навантаження відповідно до прогресу користувача, уникаючи перевантаження або недостатнього тренування. Розширена база знань включатиме не лише описи вправ, а й рекомендації щодо техніки виконання.

1.3 Постановка задачі на розробку мобільного застосунку під ОС Android для домашніх тренувань «TrainSphere»

Виходячи з аналізу практичної задачі інженерії програмного забезпечення необхідно розробити мобільний застосунку під ОС Android «TrainSphere».

З мобільним застосунком будуть працювати два користувачі: адміністратор та користувач. Адміністратор буде мати повний доступ до системи, вносити та редагувати інформацію про тренування, вправи, інвентар, групи м’язів. Користувач буде мати обмежений доступ і зможе виконувати тільки ті дії, які дозволені адміністратором, такі як внесення власних даних або виконання певних задач.

Користувач це той, хто хоче займатися спортом і буде використовувати мобільний застосунок, щоб виконувати тренування, переглядати інформацію про: тренування, вправи, групи м’язів, інвентар, робити роботу з налаштуваннями. Адміністратор – це людина, яка вноситиме весь контент в мобільний застосунок. Він додаватиме всю інформацію про: вправи, тренування, групи м’язів, інвентар, розклад тренувань, а також буде мати змогу для редагування цієї інформації.

Вимоги до складу виконуваних функцій:

Функції адміністратора:

- додавати/редагувати інформацію про тренування;
- додавати/редагувати інформацію про інвентар;
- додавати/редагувати інформацію про вправи;
- додавати/редагувати інформацію про групи м’язів;
- додавати/редагувати інформацію про розклад тренувань;
- реєструватись/авторизуватись.

Функції користувача:

- переглядати інформацію про тренування;
- виконувати тренування;
- переглядати інформацію про інвентар;
- переглядати інформацію про вправи;
- переглядати інформацію про групи м’язів;
- переглядати інформацію про розклад;
- реєструватись/авторизуватись;
- робити налаштування профілю.

Вимоги до організації вхідних та вихідних даних:

Вхідні дані вводяться в форму на екрані та заносяться у БД, вихідні дані будуть виводитись на екран з БД.

Вхідні дані:

- дані про користувача (Ім'я, Зріст, Вага, Стать, Ціль тренувань);
- інформація про вправи (Назва вправи, Опис вправи);
- інформація про тренування(Назва тренування, Опис тренування);
- інформація про розклад (Статус тренування, День);
- інформація про групи м'язів (Назва групи м'язів, Опис групи м’язів);
- інформація про інвентар (Назва інвентаря, Опис інвентаря);

Вихідні дані:

- дані користувача(Ім'я, Зріст, Вага, Стать, Ціль тренування);
- дані вправ(Назва вправи, Опис вправи);

- списки тренувань(Назва тренування, Опис тренування);
- розклад(Статус тренування, День);
- дані про групи м'язів(Назва м'язів, Опис м’язів);
- дані про інвентар(Назва інвентаря, Опис інвентаря).

Вимоги до складу та параметрів технічних засобів:

Для роботи ПЗ необхідний наступний склад технічних засобів з мінімальними параметрами:

Процесор: не нижче MediaTek MT6737 або Snapdragon 435;

Оперативна пам’ять: 2ГБ;

Дискового простору: до 300 МБ;

Інтернет-з’єднання: присутне.

Вимоги до інформаційної та програмної сумісності:

Для коректної роботи мобільного застосунку необхідне системне програмне забезпечення з ліцензією, версія якого не нижча за Android 8.0.

Повну постановку задачі сформульовано в технічному завданні, яке знаходиться в додатку А.

2 РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ПІД ОС ANDROID ДЛЯ ДОМАШНІХ ТРЕНУВАНЬ «TRAINSPHERE»

2.1 Аналіз вимог до мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

2.1.1 Розробка моделі варіантів використання мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Для кращого розуміння принципів роботи програмного забезпечення дедалі частіше застосовується опис його функціональності за допомогою варіантів використання (Use Case Diagram).

Діаграма варіантів використання (Use Case Diagram) – це графічне представлення взаємодії акторів із прецедентами в системі [4].

Діаграма варіантів використання є важливим інструментом для моделювання функціональних вимог до програмного забезпечення. Вона дозволяє наочно зобразити взаємозв’язки між користувачами системи, яких називають акторами, та функціональністю, що реалізується системою у вигляді прецедентів. Така діаграма є зручною для аналізу, проєктування та спілкування між розробниками й замовниками, оскільки забезпечує загальне розуміння того, що саме має виконувати система.

Діаграма варіантів використання становить основу для всіх наступних етапів проєктування програмного забезпечення. Кожен варіант використання описує сценарій взаємодії системи з дійовими особами або ролями, у результаті чого досягається певний значущий результат для користувача. Дійовими особами можуть виступати не лише люди, а й інші системи чи підсистеми, що взаємодіють із програмним забезпеченням.

В процесі аналізу предметної області та формування вимог було визначено два основні актори, які взаємодіють із мобільним застосунком «Trainsphere», –

Користувач та Адміністратор. Мобільний застосунок орієнтований на звичайних користувачів, які прагнуть організувати свої домашні тренування. Надалі ця роль згадується як «Користувач». Користувач може переглядати доступні тренування, вправи, інвентар, групи м’язів, а також ознайомлюватися з розкладом тренувань. Крім того, він має змогу запускати тренування безпосередньо в мобільному застосунку, слідуючи заданій послідовності вправ.

Адміністратор здійснює керування вмістом мобільного застосунку через доступ до бази даних. Його функціональні обов’язки включають додавання та редагування інформації про тренування, інвентар, вправи, групи м’язів і розклад тренувань. Таким чином, адміністратор забезпечує актуальність та структурованість даних, якими користується кінцевий користувач під час тренувань у мобільному застосунку «Trainsphere».

Визначених акторів мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» наведено в таблиці 2.1.

Таблиця 2.1 – Визначені актори мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Актор	Короткий опис
Користувач	Має доступ до перегляду даних про тренування, інвентар, вправи, групи м’язів, розклад, а також виконує тренування та може виконувати налаштування.
Адміністратор	Відповідає за додавання та редагування інформації в БД: тренування, інвентар, вправи, групи м’язів та розклад.

Загалом, взаємодія акторів із мобільним застосунком «Trainsphere» відбувається наступним чином. Користувач використовує мобільний застосунок для перегляду інформації про тренування, вправи, інвентар, групи м’язів, розкладу, виконання налаштувань. Також він може виконувати тренування відповідно до запропонованого розкладу та змісту.

Адміністратор в межах своїх повноважень може керувати контентом через доступ до БД: додавати та редагувати інформацію про тренування, вправи, інвентар, групи м’язів, розклад.

Виявлені варіанти використання мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» наведені у таблиці 2.2.

Таблиця 2.2 – Виявлені варіанти використання мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Актор	Назва	Короткий опис
1	2	3
Користувач	Переглянути тренування	Дозволяє користувачеві переглядати тренування.
Користувач	Переглянути інвентар	Дозволяє користувачеві переглядати інвентар.
Користувач	Переглянути групи м’язів	Дозволяє користувачеві переглядати групи м’язів.
Користувач	Переглянути розклад	Дозволяє користувачеві переглядати розклад.
Користувач	Переглянути вправи	Дозволяє користувачеві переглядати вправи.
Користувач	Виконати тренування	Дозволяє користувачеві виконувати тренування.
Користувач	Робота з налаштуванням	Дозволяє користувачеві змінювати персональні налаштування.
Користувач Адміністратор	Реєстрація	Дозволяє користувачеві або адміністратору реєструватись в застосунку.

Кінець таблиці 2.2

1	2	3
Користувач Адміністратор	Авторизація	Дозволяє користувачеві або адміністратору авторизовуватись в застосунку.
Адміністратор	Додати інформацію до БД	Дозволяє адміністратору додати інформацію про інвентар, вправи, тренування, розклад, групи м’язів.
Адміністратор	Редагувати інформацію в БД	Дозволяє адміністратору редагувати інформацію про інвентар, вправи, тренування, розклад, групи м’язів.

Ці варіанти використання відіграють ключову роль у розумінні того, як саме користувачі взаємодіятимуть із мобільним застосунком «Trainsphere» – сучасним інструментом для ефективних домашніх тренувань на базі ОС Android. Вони формують основу для визначення необхідного функціоналу, що дозволить задовольнити потреби як звичайних користувачів, так і адміністраторів системи. Детальний аналіз та точне формулювання цих варіантів забезпечують не лише технічну якість, але й високу зручність, мотивацію до занять і позитивний досвід користування мобільним застосунком.

Діаграма варіантів використання мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» представлена на рисунку 2.1.

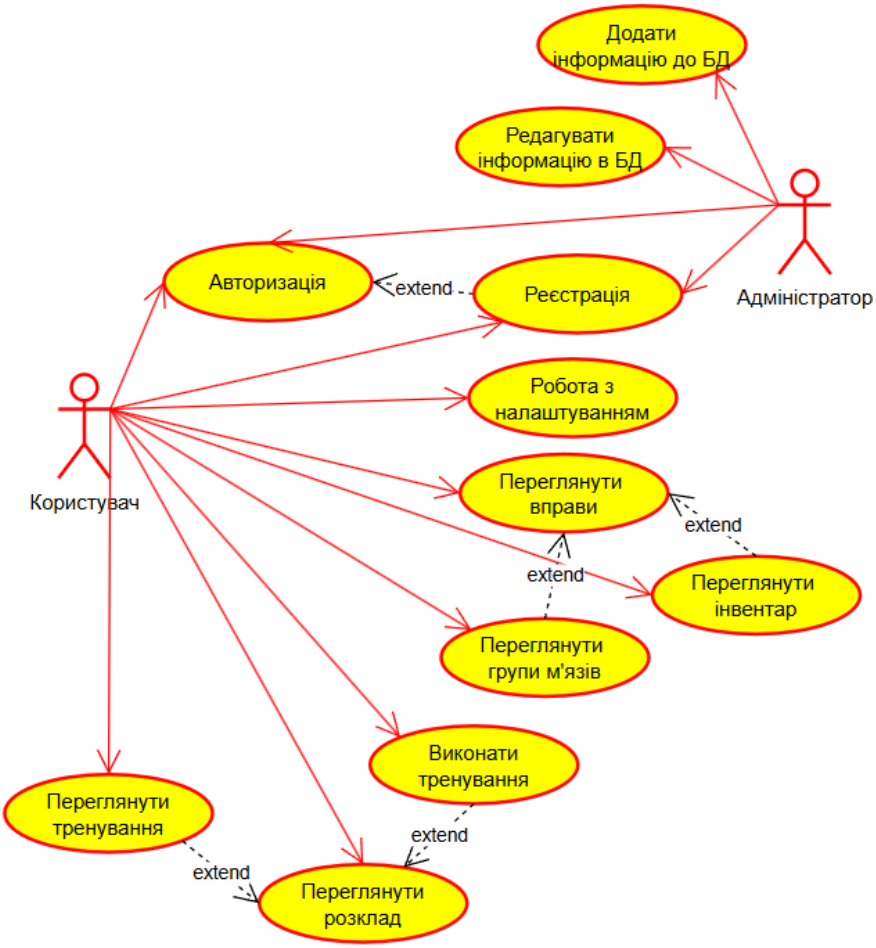


Рисунок 2.1 – Діаграма варіантів використання мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Розроблена діаграма варіантів використання мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» дає змогу відобразити результати, які користувач отримуватиме під час взаємодії з мобільним застосунком.

2.1.2 Розробка специфікацій варіантів використання мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Специфікації варіантів використання необхідні для поетапного опису дій, що виконуються користувачем під час взаємодії з програмним забезпеченням. Детально розглянемо кожен окремий варіант використання. Структура сценарію варіанта використання включає: короткий опис, передумови, основний потік подій, альтернативні сценарії та постумови [5].

Розглянемо більш детально кожен варіант використання мобільного застосунку, наводячи їх специфікації. Специфікації варіантів використання мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» представлено в таблицях 2.3 – 2.13.

Таблиця 2.3 – Специфікація варіанту використання «Переглянути тренування»

Назва прецеденту	Переглянути тренування.
Опис прецеденту	Даний варіант використання дозволяє користувачу переглянути тренування.
Дійові особи прецеденту	Користувач.
Зв’язок з іншими варіантами використання	Може розширювати варіант використання «Переглянути розклад».
Передумови	Користувач авторизований. Користувач перебуває на головному меню.
Базовий потік	1. Користувач обирає пункт меню «Тренування». 2. Мобільний застосунок демонструє список тренувань для перегляду. 3. Користувач обирає тренування зі списку для перегляду. 4. Мобільний застосунок відображає деталі тренування, відображаючи детальну інформацію про обране тренування. 6. Користувач обирає повернутись на головне меню. 7. Мобільний застосунок повертає користувача на головне меню.
Альтернативні потоки	Не визначені.
Постумови	Користувач повертається на головне меню.
Особливості	Не визначені.

Таблиця 2.4 – Специфікація варіанту використання «Переглянути вправи»

Назва прецеденту	Переглянути вправи.
1	2
Опис прецеденту	Даний варіант використання дозволяє користувачу переглянути вправи.
Дійові особи прецеденту	Користувач.
Зв’язок з іншими варіантами використання	Може бути розширений варіантами використання «Переглянути інвентар» та «Переглянути групи м’язів».
Передумови	Користувач авторизований. Користувач перебуває на головному меню.
Базовий потік	1. Користувач обирає пункт меню «Вправи». 2. Мобільний застосунок демонструє список вправ для перегляду. 3. Користувач обирає конкретну вправу зі списку. 4. Мобільний застосунок відображає деталі обраної вправи. 5. Користувач обирає повернутись на головне меню або виконується альтернативний потік 5.1 або 5.2. 6. Мобільний застосунок повертає користувача на головне меню.

Кінець таблиці 2.4

1	2
Альтернативні потоки	5.1. Користувач обирає «Переглянути інвентар». Мобільний застосунок демонструє список інвентаря для перегляду. Користувач обирає конкретний інвентар зі списку. Мобільний застосунок відображає деталі обраного інвентаря. Користувач обирає повернутись на головне меню. Мобільний застосунок повертає користувача на головне меню. 5.2. Користувач обирає «Переглянути групи м’язів». Мобільний застосунок демонструє список груп м’язів для перегляду. Користувач обирає конкретну групу м’язів зі списку. Мобільний застосунок відображає деталі обраної групи м’язів. Користувач обирає повернутись на головне меню. Мобільний застосунок повертає користувача на головне меню.
Постумови	Користувач повертається на головне меню.
Особливості	Не визначені.

Таблиця 2.5 – Специфікація варіанту використання «Переглянути групи м’язів»

Назва прецеденту	Переглянути групи м’язів.
1	2
Опис прецеденту	Даний варіант використання дозволяє користувачу переглянути групи м’язів.
Дійові особи прецеденту	Користувач.

Кінець таблиці 2.5

1	2
Зв’язок з іншими варіантами використання	Може розширювати варіант використання «Переглянути вправи».
Передумови	Користувач авторизований. Користувач перебуває на головному меню.
Базовий потік	1. Користувач обирає пункт меню «Групи м’язів». 2. Мобільний застосунок демонструє список груп м’язів для перегляду. 3. Користувач обирає конкретну групу м’язів зі списку. 4. Мобільний застосунок відображає деталі обраної групи м’язів. 5. Користувач обирає повернутись на головне меню. 6. Мобільний застосунок повертає користувача на головне меню.
Альтернативні потоки	Не визначені.
Постумови	Користувач повертається на головне меню.
Особливості	Не визначені.

Таблиця 2.6 – Специфікація варіанту використання «Переглянути інвентар»

Назва прецеденту	Переглянути інвентар.
1	2
Опис прецеденту	Даний варіант використання дозволяє користувачу переглянути інвентар.
Дійові особи прецеденту	Користувач.

Кінець таблиці 2.6

1	2
Зв’язок з іншими варіантами використання	Може розширювати варіант використання «Переглянути вправи».
Передумови	Користувач авторизований. Користувач перебуває на головному меню.
Базовий потік	1. Користувач обирає пункт меню «Переглянути інвентар». 2. Мобільний застосунок демонструє список інвентаря для перегляду. 3. Користувач обирає конкретний інвентар зі списку. 4. Мобільний застосунок відображає деталі обраного інвентарю. 5. Користувач обирає повернутись на головне меню. 6. Мобільний застосунок повертає користувача на головне меню.
Альтернативні потоки	Не визначені.
Постумови	Користувач повертається на головне меню.
Особливості	Не визначені.

Таблиця 2.7 – Специфікація варіанту використання «Робота з налаштуваннями»

Назва прецеденту	Робота з налаштуваннями.
1	2
Опис прецеденту	Даний варіант використання дозволяє користувачу виконувати налаштування для мобільного застосунку.

Кінець таблиці 2.7

1	2
Дійові особи прецеденту	Користувач.
Зв’язок з іншими варіантами використання	Немає.
Передумови	Користувач авторизований. Користувач перебуває на головному меню.
Базовий потік	1. Користувач обирає пункт меню «Налаштування». 2. Мобільний застосунок пропонує користувачу: 1) Змінити персональні дані (ім'я, вік, зріст, вагу, ціль тренувань). Користувач змінює дані про себе. Застосунок зберігає дані до БД. 2) Встановити або змінити мовні параметри застосунку. Користувач змінює налаштування мови. Застосунок зберігає дані до БД. 3. Мобільний застосунок підтверджує успішне збереження змін та повертає на головне меню.
Альтернативні потоки	Не визначені.
Постумови	Користувач повертається на головне меню.
Особливості	Не визначені.

Таблиця 2.8 – Специфікація варіанту використання «Переглянути розклад»

Назва прецеденту	Переглянути розклад.
1	2
Опис прецеденту	Даний варіант використання дозволяє користувачу переглянути розклад тренувань.

Продовження таблиці 2.8

1	2
Дійові особи прецеденту	Користувач.
Зв’язок з іншими варіантами використання	Може бути розширений варіантом використання «Переглянути тренування» та варіантом використання «Виконати тренування».
Передумови	Користувач авторизований. Користувач перебуває на головному меню.
Базовий потік	1. Користувач обирає пункт меню «Переглянути розклад». 2. Мобільний застосунок відображає поточний розклад тренувань, запланованих для користувача. 3. Користувач обирає повернутись на головне меню або виконується альтернативний потік 3.1 або 3.2. 4. Мобільний застосунок повертає користувача на головне меню.
Альтернативні потоки	3.1. Користувач обирає «Переглянути тренування». Мобільний застосунок демонструє список тренувань для перегляду. Користувач обирає конкретне тренування зі списку. Мобільний застосунок відображає деталі обраного тренування. Користувач обирає повернутись на головне меню. Мобільний застосунок повертає користувача на головне меню. 3.2. Користувач обирає «Виконати тренування». Мобільний застосунок відображає необхідне тренування. Користувач підтверджує намір виконати тренування. Мобільний застосунок запускає тренування, відображаючи відповідні інструкції. Після закінчення тренування мобільний застосунок повертає користувача на головне меню.

Кінець таблиці 2.8

1	2
Постумови	Користувач повертається на головне меню.
Особливості	Не визначені.

Таблиця 2.9 – Специфікація варіанту використання «Авторизація»

Назва прецеденту	Авторизація.
Опис прецеденту	Даний варіант використання дозволяє користувачу або адміністратору авторизуватися в мобільному застосунку.
Дійові особи прецеденту	Користувач. Адміністратор.
Зв'язок з іншими варіантами використання	Може бути розширений варіантом використання «Реєстрація».
Передумови	Користувач або адміністратор зареєстрований.
Базовий потік	1. Користувач або адміністратор відкриває застосунок. 2. Мобільний застосунок видає меню авторизації. 3. Користувач або адміністратор ініціює авторизацію. 4. Мобільний застосунок демонструє меню авторизації. 5. Користувач або адміністратор вводить необхідну інформацію для авторизації. 6. Мобільний застосунок авторизує користувача або адміністратора чи виконується альтернативний потік 6.1.
Альтернативні потоки	6.1. При невірно введених даних мобільний застосунок пропонує користувачу або адміністратору ввести дані ще раз, чи зареєструватися, якщо він це ще не зробив.
Постумови	Користувач або адміністратор авторизувався.
Особливості	Не визначені.

Таблиця 2.10 – Специфікація варіанту використання «Реєстрація»

Назва прецеденту	Реєстрація.
Опис прецеденту	Даний варіант використання дозволяє користувачу або адміністратору зареєструватися в застосунку.
Дійові особи прецеденту	Користувач. Адміністратор.
Зв'язок з іншими варіантами використання	Може розширювати варіант використання «Авторизація».
Передумови	Не визначені.
Базовий потік	1. Користувач або адміністратор відкрив застосунок. 2. Мобільний застосунок видає меню авторизації. 3. Користувач або адміністратор натискає кнопку створити аккаунт. 4. Мобільний застосунок виводить меню реєстрації. 5. Користувач або адміністратор вводить необхідні для реєстрації дані та підтверджує намір реєстрації. 6. Мобільний застосунок реєструє користувача або адміністратора та вносить дані до БД або виконується альтернативний потік 6.1. 7. Мобільний застосунок виводить повідомлення про успішну реєстрацію та переводить користувача до головного меню.
Альтернативні потоки	6.1. При невірно введених даних мобільний застосунок демонструє помилки в введених для реєстрації даних та пропонує ввести дані ще раз.
Постумови	Користувач зареєструвався.
Особливості	Не визначені.

Таблиця 2.11 – Специфікація варіанту використання «Виконати тренування»

Назва прецеденту	Виконати тренування.
Опис прецеденту	Даний варіант використання дозволяє користувачу переглянути та виконати тренування.
Дійові особи прецеденту	Користувач.
Зв’язок з іншими варіантами використання	Може розширювати варіант використання «Переглянути розклад».
Передумови	Користувач авторизований. Користувач перебуває на головному меню.
Базовий потік	1. Користувач обирає пункт меню «Виконати тренування». 2. Мобільний застосунок відображає необхідне тренування. 3. Користувач підтверджує намір виконати тренування. 4. Мобільний застосунок запускає тренування, відображаючи відповідні інструкції. 5. Після закінчення тренування мобільний застосунок повертає користувача на головне меню.
Альтернативні потоки	Немає.
Постумови	Користувач повертається на головне меню.
Особливості	Не визначені.

Таблиця 2.12 – Специфікація варіанту використання «Редагувати інформацію в БД»

Назва прецеденту	Редагувати інформацію в БД.
Опис прецеденту	Даний варіант використання дозволяє адміністратору редагувати наявну інформацію в БД (вправи, тренування, інвентар, групи м’язів).
Дійові особи прецеденту	Адміністратор.
Зв’язок з іншими варіантами використання	Немає.
Передумови	Адміністратор авторизований. Адміністратор знаходитися в розділі адміністрування.
Базовий потік	1. Адміністратор входить у розділ редагування. 2. Система відображає список доступних для редагування категорій (вправи, тренування, інвентар, групи м’язів). 3. Адміністратор обирає категорію та конкретний об’єкт для редагування. 4. Система відображає поточну інформацію про об’єкт. 5. Адміністратор вносить зміни. 6. Система зберігає оновлену інформацію до бази даних. 7. Система підтверджує успішне редагування або виконується альтернативний потік 7.1.
Альтернативні потоки	7.1. Якщо редагування не зберігається через помилку – система повідомляє про помилку і не змінює дані.
Постумови	Інформація оновлена та збережена до БД.
Особливості	Права на редагування має тільки адміністратор.

Таблиця 2.13 – Специфікація варіанту використання « Додати інформацію до БД»

Назва прецеденту	Додати інформацію до БД
Опис прецеденту	Даний варіант використання дозволяє адміністратору додати нову інформацію до БД (вправи, тренування, інвентар, групи м’язів).
Дійові особи прецеденту	Адміністратор.
Зв’язок з іншими варіантами використання	Немає.
Передумови	Адміністратор авторизований. Адміністратор знаходитися в розділі адміністрування.
Базовий потік	1. Адміністратор переходить до розділу додавання нової інформації. 2. Система пропонує вибрати категорію для додавання (вправа, тренування, інвентар, група м’язів). 3. Адміністратор обирає категорію. 4. Система надає форму для заповнення нової інформації. 5. Адміністратор вводить необхідні дані. 6. Система перевіряє, чи коректні введені дані. 7. Система зберігає новий запис у базі даних або виконується альтернативний потік 7.1. 8. Система підтверджує успішне додавання інформації.
Альтернативні потоки	7.1. Якщо дані введені некоректно – система повідомляє про помилку і пропонує повторити введення.
Постумови	Нова інформація успішно збережена в системі.
Особливості	Додавати інформацію може лише адміністратор.

Отже, спираючись на детально розроблені специфікації варіантів використання мобільного застосунку під ОС Android для домашніх тренувань «TrainSphere», можна перейти до наступного етапу – проєктування функціональних компонентів системи. Це забезпечить чітке уявлення про структуру майбутнього застосунку, його основні сценарії взаємодії з користувачем та дозволить реалізувати інтерфейс і логіку відповідно до потреб цільової аудиторії.

2.2 Розробка архітектури мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

В ході розробки мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» було прийнято рішення використовувати клієнт-серверну архітектуру.

Клієнт-серверна архітектура – це модель комп'ютерної мережі, в якій завдання та ресурси розподіляються між двома основними типами систем: клієнтами та серверами [6].

Клієнт – це програмне забезпечення або пристрій, який ініціює запити до сервера для отримання ресурсів або виконання певних операцій. Наприклад, клієнт може бути комп'ютером, мобільним пристроєм або веб-браузером.

Сервер – це високопродуктивний комп’ютер або система, що надає клієнтам доступ до послуг чи ресурсів, обробляючи їхні запити. Сервери можуть зберігати дані, виконувати обчислення, обробляти запити, а також керувати базами даних.

Можна окреслити кілька важливих аспектів, які роблять цей стиль архітектури найбільш підходящим:

− Централізоване управління даними (Вся інформація про користувачів, тренування, розклад зберігається на сервері, що забезпечує централізоване управління даними. Це дозволяє легко здійснювати резервне копіювання, оновлення даних та моніторинг стану системи. Сервер може використовувати

високопродуктивні бази даних, що гарантує швидкий доступ до інформації, навіть якщо кількість користувачів зростає);

− Забезпечення безпеки (Авторизація користувачів, обробка персональних даних і збереження чутливої інформації можуть здійснюватися безпосередньо на сервері. Система може використовувати шифрування та інші засоби захисту даних для забезпечення безпеки інформації про користувачів. Сервер може мати захищену інфраструктуру, зокрема захист від атак, що важливо для забезпечення конфіденційності даних);

− Мультимедійна підтримка (Оскільки застосунок передбачає мультимедійні матеріали (відео та анімації), сервер буде обробляти та зберігати ці матеріали, а клієнт лише запитуватиме їх по мірі необхідності. Це дозволяє зберігати великі файли на сервері та завантажувати їх за запитом, що значно зменшує навантаження на мобільний пристрій);

− Масштабованість і гнучкість (Клієнт-серверна архітектура сприяє легкому розширенню серверної логіки з мінімальними змінами в поточній структурі у мобільному застосунку. Всі основні функції, такі як обробка тренувань, збереження статистики, доступ до бази даних, здійснюються на сервері, що дозволяє знижувати навантаження на клієнтські пристрої);

− Інтеграція з іншими системами (Через сервер можна інтегрувати застосунок з іншими сервісами, такими як системи нагадувань, календарі та хмарні сховища для зберігання великих даних. Можна забезпечити синхронізацію даних з між різними пристроями користувача (мобільним телефоном, планшетом), а також додавати нові функції, що потребують доступу до зовнішніх даних).

Технічні переваги клієнт-серверної архітектури:

− Простота розробки і тестування: Основна логіка програми реалізується на сервері, що спрощує тестування та оновлення застосунку. Для клієнта достатньо забезпечити зручний інтерфейс і можливість зв'язку з сервером;

− Надійність та стійкість до збоїв: Якщо сервер правильно налаштований, то навіть у разі збою на клієнтському пристрої, застосунок може продовжувати працювати на інших пристроях з тими ж даними;

− Легке оновлення: Оновлення функціоналу програми (як на сервері, так і на клієнті) відбуваються централізовано, що дозволяє швидко реагувати на нові вимоги без значних змін у програмному коді клієнтської частини.

Для реалізації мобільного застосунку під ОС Android для домашніх тренувань «TrainSphere» був використаний патерн проєктування програмного забезпечення MVC (Model-View-Controller).

Патерн MVC (Model-View-Controller) – це один із найпоширеніших патернів проєктування, який дозволяє структуризувати програму шляхом виокремлення трьох основних частин: Model, View та Controller. Це дозволяє покращити організацію коду, зробити його гнучким і легким для тестування та підтримки [7].

Опис компонентів MVC:

Model (Модель). Модель відповідає за дані та бізнес-логіку програми. Вона інкапсулює всі операції, які можуть бути виконані з даними (наприклад, збереження, обробка та отримання інформації). У контексті мобільного застосунку для тренувань модель може зберігати дані про користувачів, тренування, розклад і так далі. Модель не має уявлення про те, як ці дані будуть представлені або як взаємодіяти з користувачем.

View (Вигляд). У шаблоні MVC View відповідає за візуалізацію інформації для користувача. Він приймає дані з моделі та подає їх у форматі, зручному для користувача (наприклад, на екрані мобільного пристрою). Вигляд не містить бізнес-логіки, він лише відповідає за інтерфейс і виведення інформації. У в розроблюваному мобільному застосунку він відповідає за екран тренувань, розклад, список вправ і так далі.

Controller (Контролер). Контролер є посередником між Model та View.

Він отримує введення від користувача (наприклад, натискання кнопки, вибір опції меню), обробляє їх (запитує модель для отримання даних або зміни стану) і оновлює вигляд. Контролер також може оновлювати модель, коли дані змінюються (наприклад, коли користувач починає тренування, результат тренування зберігається в моделі).

Нижче, на рисунку 2.2 представлено діаграму компонентів мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

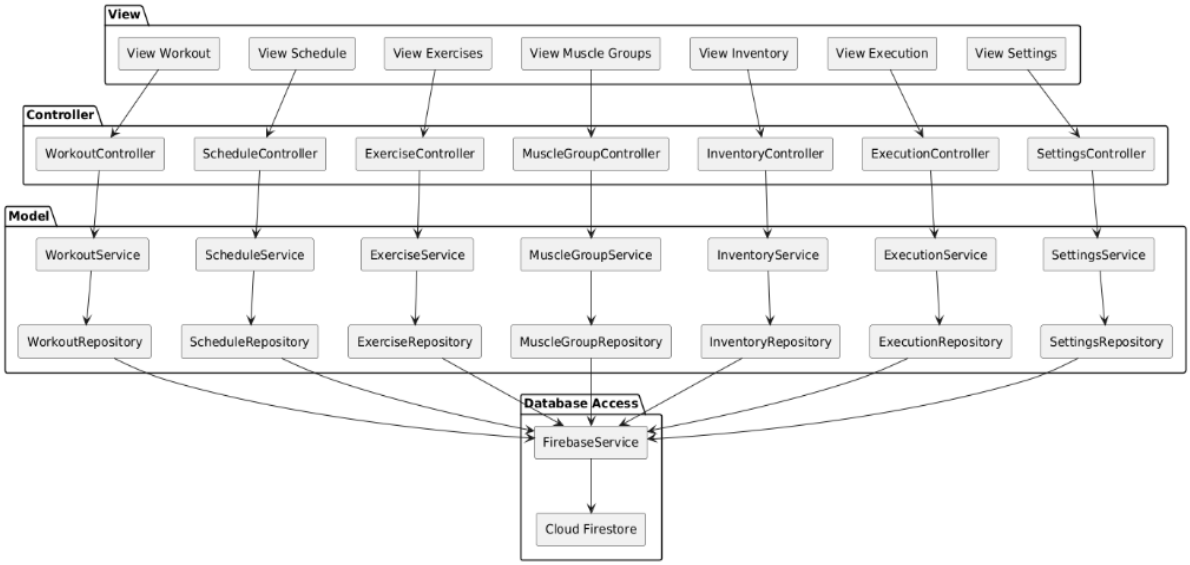


Рисунок 2.2 – Діаграма компонентів мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Діаграма компонентів відображає основні компоненти програмного забезпечення та їх взаємодії. Вона дає чітке уявлення про структуру системи, де кожен компонент має свою чітко визначену роль, і як вони взаємодіють для досягнення бажаного результату [8].

Архітектура реалізує шаблон MVC (Model–View–Controller), де:

- View – відповідає за показ інформації та інтерфейс користувача.
- Controller – приймає дії користувача, взаємодіє із сервісами.
- Model – містить бізнес-логіку та взаємодіє з базою даних.

Компоненти представлення (View) забезпечують інтерфейс користувача, через який відображається інформація та приймається ввід:

View Workout – інтерфейс для перегляду, створення або редагування тренувань.

View Schedule – представлення графіка тренувань користувача у вигляді календаря або списку.

View Exercises – інтерфейс для перегляду списку вправ з інформацією про них.

View Muscle Groups – відображення вправ за категоріями м’язових груп.

View Inventory – показує наявне спортивне спорядження або обладнання.

View Execution – режим безпосереднього виконання тренування, включає таймери, прогрес, підказки.

View Settings – дозволяє користувачу змінювати налаштування (тема, мова, одиниці виміру тощо).

Контролери (Controller) посередники, що обробляють події з View, передають їх до відповідних сервісів:

WorkoutController – обробляє дії користувача щодо створення, редагування або перегляду тренувань.

ScheduleController – відповідає за створення і редагування графіка тренувань.

ExerciseController – керує отриманням та фільтрацією вправ.

MuscleGroupController –взаємодіє із сервісами для отримання груп м’язів та відповідних вправ.

InventoryController – працює із даними інвентарю.

ExecutionController – координує процес виконання тренувань: запуск, відлік часу, реєстрація результатів.

SettingsController – керує оновленням користувацьких налаштувань.

Model містить логіку додатку, поділену на сервіси (що реалізують логіку) та репозиторії (що працюють з базою):

Сервіси:

WorkoutService – реалізує логіку створення, редагування і зберігання тренувань.

ScheduleService – будує графік тренувань, перевіряє конфлікти, тощо.

ExerciseService – фільтрує вправи, надає інформацію про техніку виконання.

MuscleGroupService – логіка взаємодії з м’язовими групами.

InventoryService – обробляє інформацію про спортивне обладнання.

ExecutionService – реалізує логіку тренування в реальному часі: таймери, індикатори, підрахунок підходів і результатів.

SettingsService – зберігає та застосовує індивідуальні налаштування користувача.

Репозиторії:

WorkoutRepository – зберігає/зчитує дані тренувань із Firebase.

ScheduleRepository – зберігає/отримує графіки.

ExerciseRepository – отримує вправи з бази.

MuscleGroupRepository – зчитує м’язові групи та їх зв’язки з вправами.

InventoryRepository – оперує даними про інвентар.

ExecutionRepository – зберігає результати виконаних тренувань.

SettingsRepository – працює з користувацькими параметрами (мова, одиниці, інтерфейс).

Database Access – цей шар забезпечує взаємодію з хмарною базою даних Firebase.

FirebaseService – універсальний сервіс, який надає методи для запису, читання, оновлення та видалення даних. Абстрагує логіку роботи з Firebase SDK, дозволяючи сервісам/репозиторіям не залежати від конкретного API.

Cloud Firestore – використовується як головна база даних. Вона дозволяє:

- зберігати структуровані документи з тренуваннями, вправами, розкладом, результатами тощо;
- працювати в режимі реального часу;
- масштабуватись автоматично;
- мати гнучку структуру колекцій та документів.

Діаграма взаємодії компонентів демонструє архітектурну структуру [9]. Вона ілюструє, як основні частини застосунку взаємодіють між собою на рівні логіки роботи – починаючи від інтерфейсу користувача (View) і закінчуючи збереженням даних у хмарі (Firebase/Firestore) .

Кожен компонент відображає окрему відповідальність:

- View відповідає за візуальне відображення і взаємодію з користувачем;

- Controller обробляє події з View і координує логіку;
- Service містить бізнес-логіку застосунку;
- Repository забезпечує взаємодію з Firebase;
- FirebaseService є посередником між додатком та хмарною базою даних Cloud Firestore.

На рисунку 2.3 представлено діаграму взаємодії компонентів мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere».

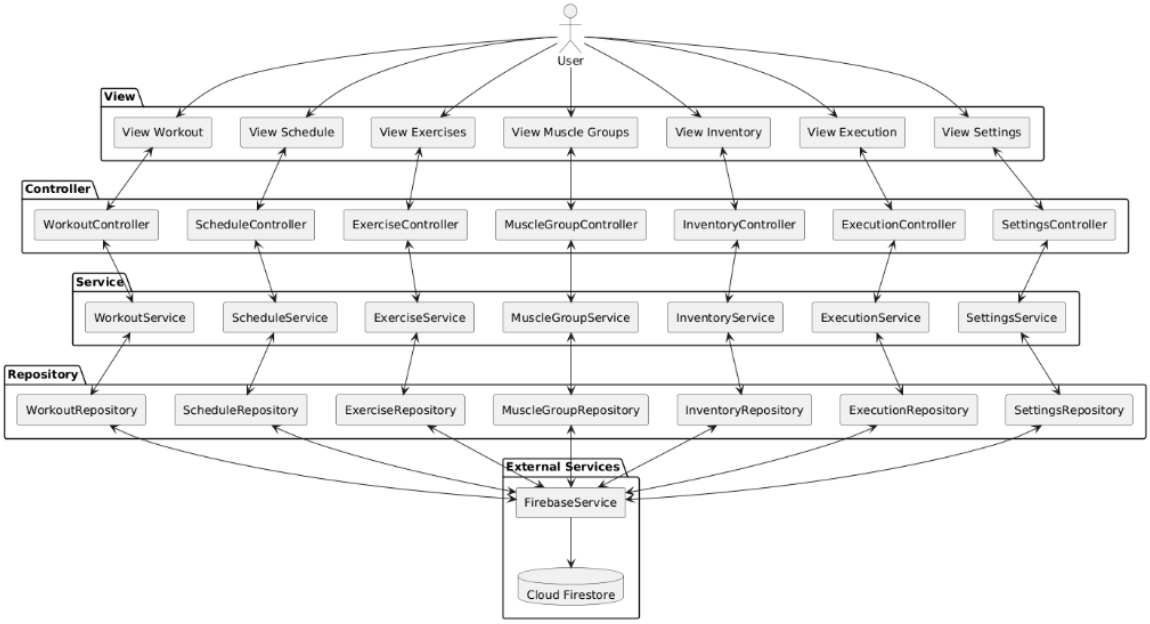


Рисунок 2.3 – Діаграма взаємодії компонентів мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Розробка діаграми взаємодії компонентів дає цілісне уявлення про те, як дані рухаються через мобільний застосунок: від дій користувача до оновлення бази даних і назад, дозволяючи ефективно аналізувати та вдосконалювати архітектуру системи.

В UML-нотації діаграма розгортання слугує для відображення фізичного розміщення програмних модулів на апаратних вузлах. Вона показує, як програмні компоненти (програми, сервіси, бази даних тощо) розгортаються на фізичних або віртуальних пристроях (вузлах).

Основна мета цієї діаграми – візуалізувати інфраструктуру системи, зрозуміти, які компоненти де виконуються, як вони взаємодіють між собою, і які канали зв'язку використовуються. Це допомагає в проєктуванні, аналізі продуктивності, розподілі навантаження, а також при розгортанні системи у виробниче середовище [10].

Діаграма розгортання мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» представлена на рисунку 2.4.

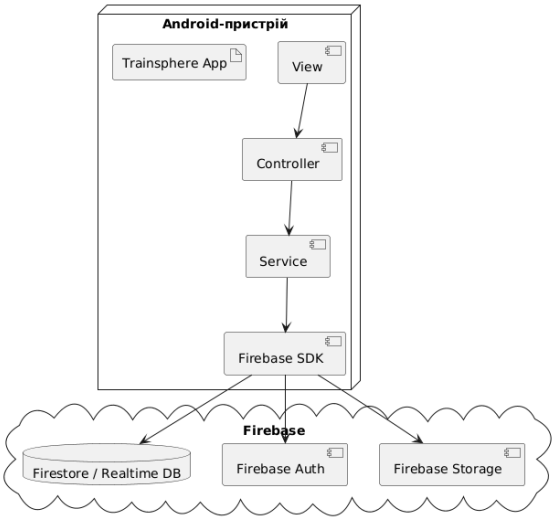


Рисунок 2.4 – Діаграма розгортання мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Розроблена діаграма розгортання програмного забезпечення відображає загальну структуру та топологію розподіленої системи, демонструючи, як програмні компоненти розміщуються на окремих вузлах цієї системи.

2.3 Проєкт мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

2.3.1 Ескізний проєкт мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Побудова діаграми діяльності мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Під час моделювання процесів виконання операцій у програмному забезпеченні можна скористатися мовою UML та використати для цього діаграми діяльності. Вони мають подібність до діаграм станів, однак між ними є принципова різниця: стани в діаграмі діяльності відображають не просто зміну стану об’єкта, а конкретні дії або кроки алгоритму. Кожен стан у такій діаграмі відповідає за виконання певної дії. Після завершення однієї дії можливий перехід до наступної – відповідно до логіки виконання.

Діаграма діяльності – це вид UML-діаграми, що використовується для опису послідовності дій або логіки поведінки системи. Вона подається у вигляді графа, де вершини представляють дії (операції), а дуги – переходи між ними, що відображають порядок виконання [11].

Діаграми діяльності зазвичай застосовують для опису алгоритмів реалізації операцій класів або логіки процесів, які відбуваються в системі. Кожен стан може представляти виконання окремої операції певного класу або її частини, що дозволяє моделювати реакцію системи на внутрішні події. Таким чином, діаграма діяльності є частковим випадком діаграми станів, але орієнтована передусім на зображення динамічної поведінки у вигляді послідовних дій.

Діаграми діяльності мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» для деяких варіантів використання наведені на рисунках 2.5 – 2.7.

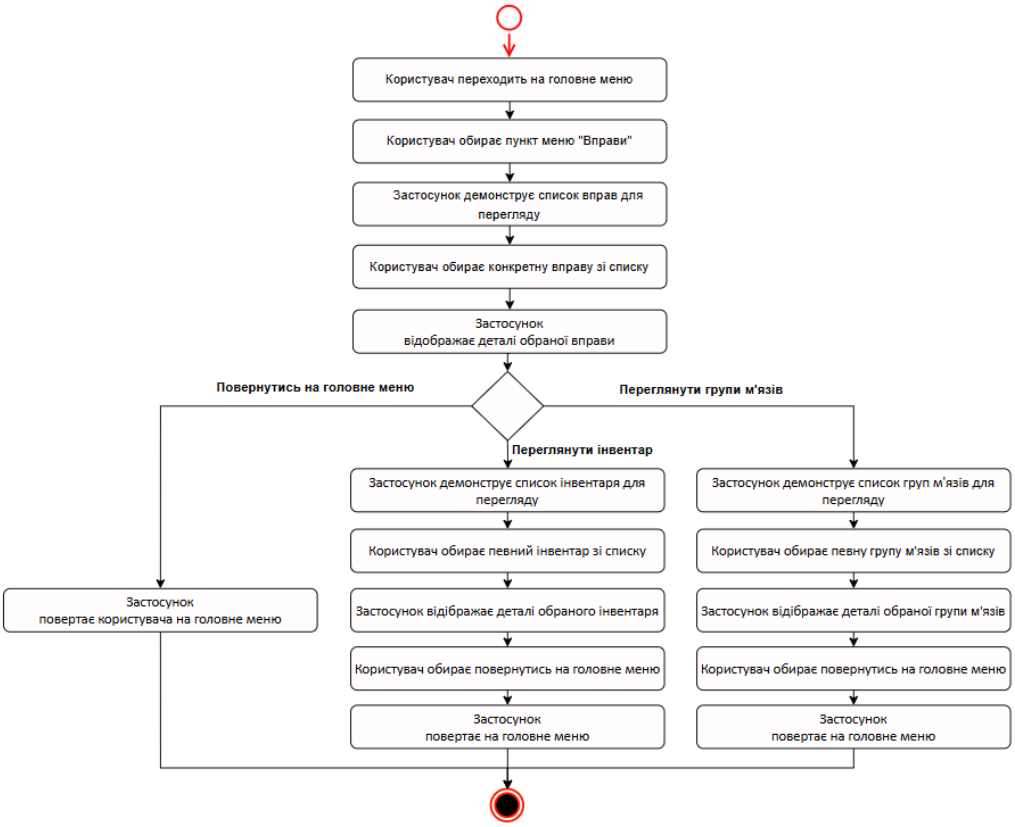


Рисунок 2.5 – Діаграма діяльності варіанту використання «Переглянути вправи»

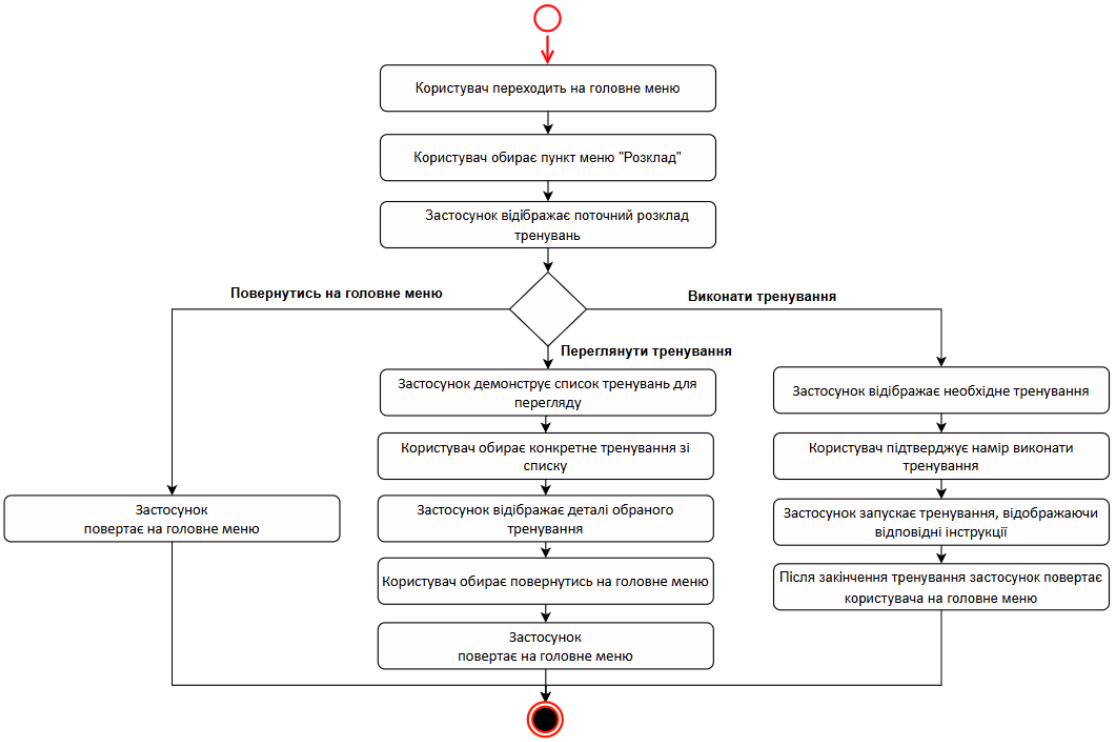


Рисунок 2.6 – Діаграма діяльності варіанту використання «Переглянути розклад»

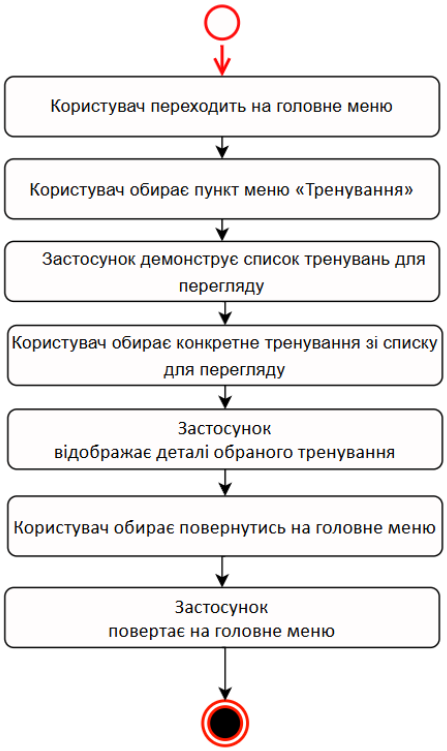


Рисунок 2.7 – Діаграма діяльності варіанту використання «Переглянути тренування»

Побудовані діаграми діяльності ефективно ілюструють як послідовність виконуваних дій, так і події, що їх запускають або є їх результатом.

Розробка концептуальної моделі даних мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Початком проєктування бази даних програмного забезпечення є побудова концептуальної моделі даних на основі аналізу функціонування мобільного застосунку для тренувань користувачів.

Концептуальна модель даних – це модель, яка описує об’єкти предметної області та зв’язки між ними. Така модель є засобом комунікації між розробниками, замовниками та кінцевими користувачами. Вона розробляється незалежно від конкретного способу реалізації чи фізичного зберігання даних [12].

Найбільш поширеним підходом до концептуального проєктування є ER-модель ("сутність – зв’язок"). ER-модель дозволяє формально представити

структуру даних за допомогою сутностей, їх атрибутів та типів зв’язків між ними. Вона є мета-моделлю даних, яка забезпечує логічне відображення інформаційної структури предметної області.

Сутність – це абстрактне або конкретне поняття, яке відображає об’єкт предметної області, що має значення для інформаційної системи та про яке необхідно зберігати дані [13].

Сутність є центральним елементом концептуальної моделі даних, адже вона описує ті об'єкти, які мають стале існування, відрізняються один від одного певними характеристиками (атрибутами) і можуть вступати у взаємозв’язки з іншими сутностями. Кожна сутність у моделі представляє множину екземплярів (тобто об'єктів одного типу).

Концептуальна модель даних базується на семантичній інформації про предметну область – у цьому випадку про взаємодію користувачів з тренуваннями, вправами, інвентарем та іншими функціями застосунку. Модель відображає ключові сутності, такі як Користувач, Вправи, Тренування, Розклад, Інвентар, Група м’язів та Налаштування, а також їх взаємозв’язки. Наприклад, користувачі виконують вправи, які належать до певних тренувань, містяться у розкладі, потребують інвентарю та спрямовані на прокачку конкретних груп м’язів.

Визначення атрибутів сутностей мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» представлено нижче.

Список атрибутів сутності Користувач:

- Код користувача;
- Ціль тренування;
- Стать;
- Зріст;
- Вага;
- Ім'я.

Список атрибутів сутності Вправи:

- Код вправи;
- Назва;

- Деталі вправи.

Список атрибутів сутності Тренування:

- Код тренування;
- Назва;
- Деталі тренування.

Список атрибутів сутності Розклад:

- Код розкладу;
- День;
- Статус тренування.

Список атрибутів сутності Інвентар:

- Код інвентаря;
- Назва;
- Деталі інвентаря.

Список атрибутів сутності Група м’язів:

- Код групи м'язів;
- Назва;
- Деталі групи м'язів.

Список атрибутів сутності Налаштування:

- Код налаштування;
- Мова.

Визначення атрибутів сутностей допомогло чітко структурувати дані та встановити логічні зв’язки між основними об’єктами системи.

Зв’язок у концептуальній діаграмі – це логічна асоціація між двома або більше сутностями, яка показує, як об’єкти предметної області взаємодіють або пов’язані між собою. Зв’язки відображаються у вигляді ліній між сутностями та можуть мати різну кратність: один до одного (1:1), один до багатьох (1:N) або багато до багатьох (M:N), що визначає кількість можливих відповідностей між екземплярами сутностей. Крім того, зв’язки можуть бути обов’язковими або необов’язковими, залежно від логіки системи, та можуть мати власні атрибути, якщо зв’язок описує додаткову інформацію про взаємодію між сутностями.

Зв’язки між сутностями мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» представлені в таблиці 2.14.

Таблиця 2.14 – Зв’язки між сутностями мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Сутність 1	Сутність 2	Зв’язок
Користувач	Тренування	Багато-до-багатьох
Користувач	Тренування	Один-до-одного
Тренування	Вправи	Багато-до-багатьох
Тренування	Розклад	Один-до-одного
Вправи	Групи м’язів	Багато-до-багатьох
Вправи	Інвентар	Багато-до-багатьох

Концептуальна модель даних мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» представлена на рисунку 2.8.

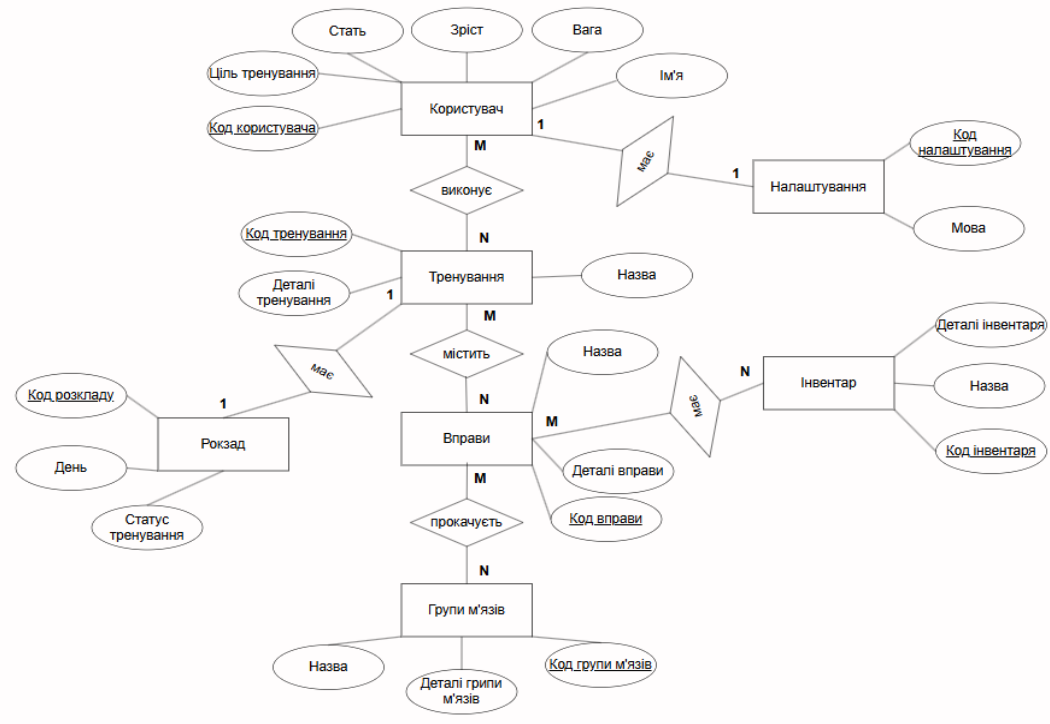


Рисунок 2.8 – Концептуальна модель даних мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Концептуальна модель даних мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» включає декілька основних сутностей, які взаємодіють між собою для забезпечення персоналізованого користувацького досвіду.

Концептуальна модель даних відображає структуру даних системи:

Користувач (Користувач) – центральна сутність, яка містить інформацію про клієнта (ім’я, стать, зріст, вага, ціль тренування тощо). Кожен користувач має свої Налаштування, зокрема мову інтерфейсу.

Тренування – кожен користувач може виконувати декілька тренувань. Тренування має назву, деталі та складається з кількох Вправ.

Вправи – є складовими тренування та можуть використовувати Інвентар. Кожна вправа впливає на певні Групи м’язів.

Інвентар – представляє аксесуари, що можуть використовуватись під час виконання вправ. Містить унікальний код та назву.

Групи м’язів – сутність, що використовується для класифікації ефекту від виконання вправ.

Розклад – вказує на день проведення тренування та його статус, забезпечуючи організацію тренувального процесу.

Модель забезпечує гнучкість у зберіганні та обробці користувацької інформації, їх активність та взаємодію з аксесуарами, дозволяючи створити адаптивний і персоналізований інтерфейс мобільного додатку.

Концептуальна модель даних стала основою для подальшого проєктування архітектури системи. Вона дозволила чітко визначити ключові сутності та зв’язки між ними, що забезпечило узгодженість під час розробки бази даних, бізнес-логіки й користувацького інтерфейсу мобільного застосунку «Trainsphere».

Розробка прототипу інтерфейсу користувача мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Оскільки якість інтерактивної взаємодії користувача з мобільним застосунком залежить від особливостей людського сприйняття (швидкість реакції, пам’ять, візуальне сприйняття тощо), інтерфейс повинен відповідати таким ключовим властивостям:

- Адаптованість – здатність інтерфейсу підлаштовуватись під різні пристрої, користувацькі налаштування та середовище використання.
- Юзабіліті (зручність користування) – простота навігації, логічність розміщення елементів і швидкий доступ до основного функціоналу.
- Достатність – наявність лише необхідних функцій без зайвого перевантаження.
- Дружність – інтуїтивна зрозумілість та естетична привабливість інтерфейсу для користувача.
- Гнучкість – можливість змінювати параметри та налаштовувати застосунок відповідно до індивідуальних потреб користувача.

З урахуванням зазначених властивостей було спроєктовано ескізи інтерфейсу користувача майбутнього мобільного застосунку для планування та виконання тренувань. Ескізи інтерфейсу продемонстровано на рисунках 2.9 – 2.10.

Рисунок 2.9 – Ескіз головного меню мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Рисунок 2.10 – Ескіз меню виконання тренування мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Побудовані ескізи інтерфейсу мобільного застосунку для тренувань покликані задовольнити основні потреби користувачів, забезпечуючи зручність, ефективність та інтуїтивність у процесі взаємодії із системою.

2.3.2 Технічний проєкт мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Побудова діаграми класів мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Діаграма класів – це статичне подання структури системи в нотації UML, яке відображає основні елементи моделі, такі як класи, типи даних, їхні атрибути та методи, а також відносини між ними. Вона показує статичні (декларативні) компоненти системи і слугує для моделювання структури програмного забезпечення в об’єктно-орієнтованому підході [14]. На діаграмі класів можуть бути представлені також інтерфейси, об’єкти, кооперації, а також позначення пакетів і вкладених пакетів.

У діаграмі класів для мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» представлені основні класи, що використовуються у функціональності системи. Користувачі можуть реєструватись, авторизовуватись,

переглядати інформацію про вправи, інвентар, групи м’язів, переглядати розклад, виконувати тренування, виконувати роботу з налаштуванням. Адміністратори мають можливість авторизовуватись та реєструватись, додавати та редагувати дані в БД. Ця діаграма відображає статичну структуру системи, демонструючи класи, їхній вміст та взаємозв’язки між ними.

На рисунку 2.11 представлено діаграму класів мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

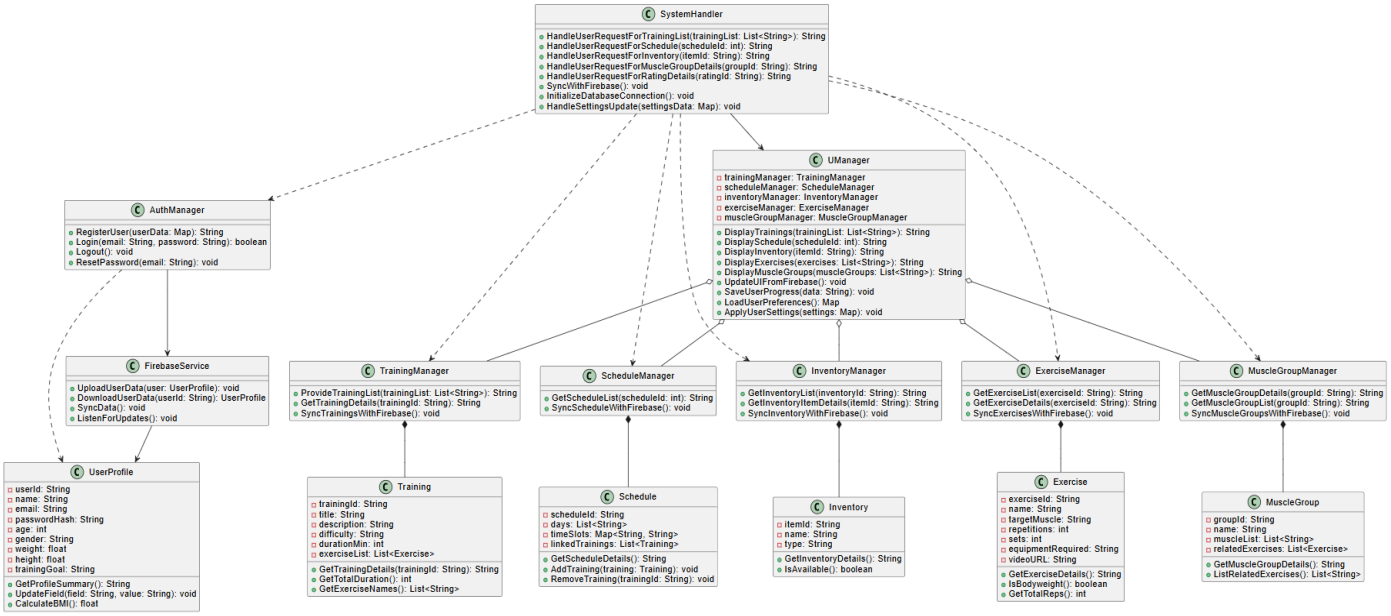


Рисунок 2.11 – Діаграма класів мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Ця діаграма класів описує архітектуру компонентів системи, яка обробляє запити користувача, пов’язані з тренуваннями, розкладом, інвентарем, вправами та музичними групами. Центральним компонентом є клас SystemHandler, який приймає запити користувача та делегує їх класу UManager. Клас UManager керує взаємодією з менеджерами доменних областей, де кожен компонент виконує свою функціональну роль: тренування, розклади, інвентар, вправи та музика. Дані синхронізуються через Firebase.

Специфікації класів мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Специфікація класу – це опис його властивостей, що визначають стан об’єкта, та методів, які реалізують поведінку цього об’єкта. Специфікації класів мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» наведено в таблицях 2.15 – 2.29.

Таблиця 2.15 – Специфікація класу SystemHandler

Характеристика	Опис
Призначення	Головний клас обробки запитів користувача і взаємодії з менеджерами та Firebase.
Атрибути	Немає атрибутів
Методи	HandleUserRequestForTrainingList(trainingList: List<String>) – Обробляє запит користувача на список тренувань. HandleUserRequestForSchedule(scheduleId: int) – Обробляє запит користувача на розклад. HandleUserRequestForInventory(itemId: String) – Обробляє запит користувача на інвентар. HandleUserRequestForMuscleGroupDetails(groupId: String) – Обробляє запит на деталі м'язової групи. HandleUserRequestForRatingDetails(ratingId: String) – Обробляє запит на деталі оцінювання. SyncWithFirebase() – Синхронізує дані з Firebase. InitializeDatabaseConnection() – Ініціалізує з'єднання з базою даних. HandleSettingsUpdate(settingsData: Map) – Обробляє оновлення налаштувань користувача.

Таблиця 2.16 – Специфікація класу UManager

Характеристика	Опис
Призначення	Управляє інтерфейсом користувача та координує запити до відповідних менеджерів.
Атрибути	trainingManager: TrainingManager – Менеджер тренувань. scheduleManager: ScheduleManager – Менеджер розкладу. inventoryManager: InventoryManager – Менеджер інвентарю. exerciseManager: ExerciseManager – Менеджер вправ. muscleGroupManager: MuscleGroupManager – Менеджер м'язових груп.
Методи	DisplayTrainings(trainingList: List<String>) – Відображає список тренувань. DisplaySchedule(scheduleId: int) – Відображає розклад. DisplayInventory(itemId: String) – Відображає інвентар. DisplayExercises(exercises: List<String>) – Відображає список вправ. DisplayMuscleGroups(muscleGroups: List<String>) – Відображає м'язові групи. UpdateUIFromFirebase() – Оновлює інтерфейс користувача з даних Firebase. SaveUserProgress(data: String) – Зберігає прогрес користувача. LoadUserPreferences() – Завантажує налаштування користувача. ApplyUserSettings(settings: Map) – Застосовує нові налаштування користувача.

Таблиця 2.17 – Специфікація класу TrainingManager

Характеристика	Опис
Призначення	Керує логікою тренувань: наданням списків, деталей та синхронізацією.
Атрибути	Немає атрибутів.
Методи	ProvideTrainingList(trainingList: List<String>) – Повертає список тренувань. GetTrainingDetails(trainingId: String) – Повертає деталі конкретного тренування. SyncTrainingsWithFirebase() – Синхронізує тренування з Firebase.

Таблиця 2.18 – Специфікація класу ScheduleManager

Характеристика	Опис
Призначення	Керує розкладом тренувань користувача.
Атрибути	Немає атрибутів.
Методи	GetScheduleList(scheduleId: int) – Повертає список розкладів або розклад по ID. SyncScheduleWithFirebase() – Синхронізує розклад з Firebase.

Таблиця 2.19 – Специфікація класу InventoryManager

Характеристика	Опис
Призначення	Відповідає за логіку управління інвентарем користувача.
Атрибути	Немає атрибутів.
Методи	GetInventoryList(inventoryId: String) – Повертає список інвентарю. GetInventoryItemDetails(itemId: String) – Повертає деталі елементу інвентарю. SyncInventoryWithFirebase() – Синхронізує інвентар з Firebase.

Таблиця 2.20 – Специфікація класу ExerciseManager

Характеристика	Опис
Призначення	Керує вправами: отримання списків, деталей та синхронізація.
Атрибути	Немає атрибутів.
Методи	GetExerciseList(exerciseId: String) – Повертає список вправ. GetExerciseDetails(exerciseId: String) – Повертає деталі вправи. SyncExercisesWithFirebase() – Синхронізує вправи з Firebase.

Таблиця 2.21 – Специфікація класу MuscleGroupManager

Характеристика	Опис
Призначення	Керує м'язовими групами: отриманням списку, деталей та синхронізацією.
Атрибути	Немає атрибутів.
Методи	GetMuscleGroupDetails(groupId: String) – Повертає деталі м'язової групи. GetMuscleGroupList(groupId: String) – Повертає список м'язових груп. SyncMuscleGroupsWithFirebase() – Синхронізує м'язові групи з Firebase.

Таблиця 2.22 – Специфікація класу Training

Характеристика	Опис
1	2
Призначення	Модель даних тренування, яка містить детальну інформацію про сесію.

Кінець таблиці 2.22

1	2
Атрибути	trainingId: String – ідентифікатор тренування. title: String – назва тренування. description: String – опис. difficulty: String – рівень складності. durationMin: int – тривалість у хвилинах. exerciseList: List<Exercise> – список вправ у тренуванні.
Методи	GetTrainingDetails(trainingId: String) – Повертає опис тренування за ID. GetTotalDuration() – Обчислює загальну тривалість тренування. GetExerciseNames() – Повертає список назв вправ, що входять до тренування.

Таблиця 2.23 – Специфікація класу Schedule

Характеристика	Опис
Призначення	Модель даних розкладу, що представляє тренувальний графік.
Атрибути	scheduleId: String – ідентифікатор розкладу. days: List<String> – список днів тижня. timeSlots: Map<String, String> – таймслоти. linkedTrainings: List<Training> – пов'язані тренування.
Методи	GetScheduleDetails() – Повертає детальну інформацію про розклад. AddTraining(training: Training) – Додає тренування до розкладу. RemoveTraining(trainingId: String) – Видаляє тренування з розкладу за його ID.

Таблиця 2.24 – Специфікація класу Inventory

Характеристика	Опис
Призначення	Модель інвентарю, що використовується у тренуваннях.
Атрибути	temId: String – ідентифікатор предмета. name: String – назва. type: String – тип інвентарю (гантелі, стрічки тощо).
Методи	GetInventoryDetails() – Повертає опис інвентарного елемента. IsAvailable() – Перевіряє наявність інвентарю.

Таблиця 2.25 – Специфікація класу Exercise

Характеристика	Опис
Призначення	Модель вправи: містить інформацію про вправи
Атрибути	exerciseId: String – ідентифікатор вправи. name: String – назва вправи. targetMuscle: String – цільовий м’яз. repetitions: int – кількість повторів. sets: int – кількість підходів. equipmentRequired: String – необхідне обладнання. videoURL: String – посилання на відео з технікою виконання.
Методи	GetExerciseDetails() – Повертає детальний опис вправи. IsBodyweight() – Визначає, чи є вправа з вагою власного тіла. GetTotalReps() – Обчислює загальну кількість повторень.

Таблиця 2.26– Специфікація класу MuscleGroup

Характеристика	Опис
1	2
Призначення	Модель м’язової групи для класифікації вправ.

Кінець таблиці 2.26

1	2
Атрибути	groupId: String – ідентифікатор групи м’язів. name: String – назва м’язової групи. muscleList: List<String> – список м’язів у групі. relatedExercises: List<Exercise> – пов’язані вправи.
Методи	GetMuscleGroupDetails() – Повертає опис групи м’язів. ListRelatedExercises() – Повертає список вправ, пов’язаних з цією групою м’язів.

Таблиця 2.27 – Специфікація класу UserProfile

Характеристика	Опис
Призначення	Зберігає інформацію про користувача: особисті дані, цілі, прогрес.
Атрибути	userId: String – Унікальний ідентифікатор користувача. name: String – Ім’я користувача. email: String – Електронна пошта користувача. passwordHash: String – Захешований пароль користувача. age: int – Вік користувача. gender: String – Стать користувача. weight: float – Вага користувача в кг. height: float – Зріст користувача в см. trainingGoal: String – Ціль тренувань користувача.
Методи	GetProfileSummary() – Повертає дані користувача. UpdateField() – Оновлює конкретне поле профілю. CalculateBMI() – Обчислює індекс маси тіла користувача.

Таблиця 2.28 – Специфікація класу FirebaseService

Характеристика	Опис
Призначення	Відповідає за централізовану синхронізацію даних з Firebase.
Атрибути	Немає атрибутів
Методи	UploadUserData(user: UserProfile) – Завантажує дані користувача до Firebase. DownloadUserData(userId: String) – Завантажує дані користувача з Firebase за його ID. SyncData() – Синхронізує локальні дані з Firebase. ListenForUpdates() – Слухає оновлення в Firebase у режимі реального часу.

Таблиця 2.29 – Специфікація класу AuthManager

Характеристика	Опис
Призначення	Відповідає за авторизацію, реєстрацію та безпеку користувача.
Атрибути	Немає атрибутів
Методи	RegisterUser(userData: Map) – Реєструє нового користувача. LoginUser(email: String, password: String) – Авторизує користувача. LogoutUser() – Виконує вихід користувача. ResetPassword(email: String) – Надсилає запит на скидання пароля.

Побудовані специфікації класів дали змогу краще зрозуміти структуру, функціональність і взаємозв’язки між основними компонентами системи. Завдяки чітко описаним атрибутам і методам стало зрозуміло, яку відповідальність несе кожен клас, які дані він обробляє та як саме взаємодіє з іншими елементами.

Побудова діаграм послідовності мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

На наступному етапі розробки програмного забезпечення здійснюється побудова динамічної моделі, яка відображає логіку взаємодії між об’єктами системи в процесі виконання її основних функцій. В основі цього етапу лежать раніше створені моделі, зокрема діаграма варіантів використання та діаграма класів.

На підставі цих моделей було розроблено діаграми послідовності, що демонструють порядок викликів методів між об'єктами для реалізації ключових сценаріїв роботи системи [15]. Діаграму послідовності прецедентів представлено на рисунках 2.12 – 2.13, вона втілює основні варіанти використання, сформовані в рамках функціональної моделі програмного забезпечення.

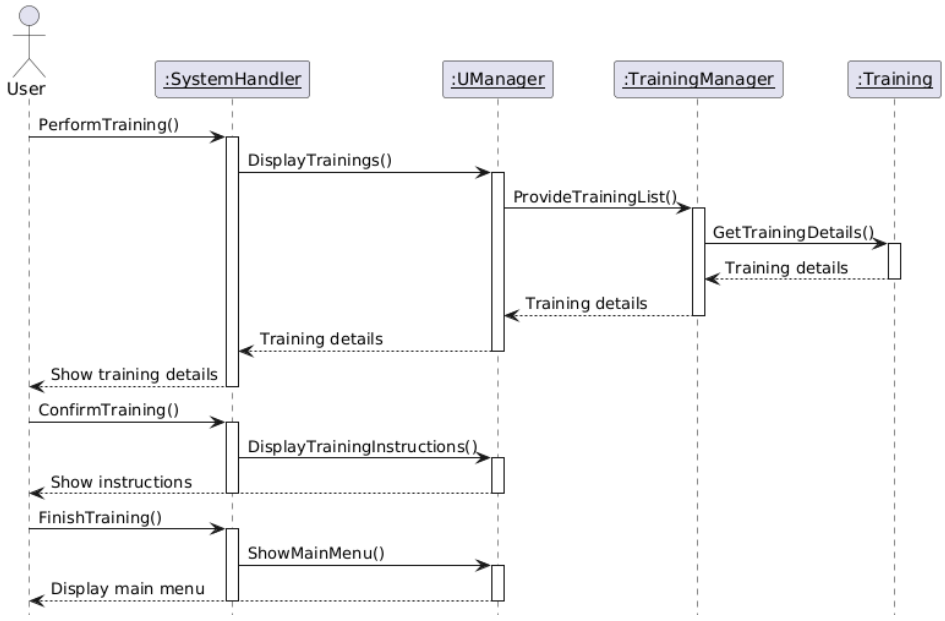


Рисунок 2.12 – Діаграма послідовності прецеденту «Виконати тренування»

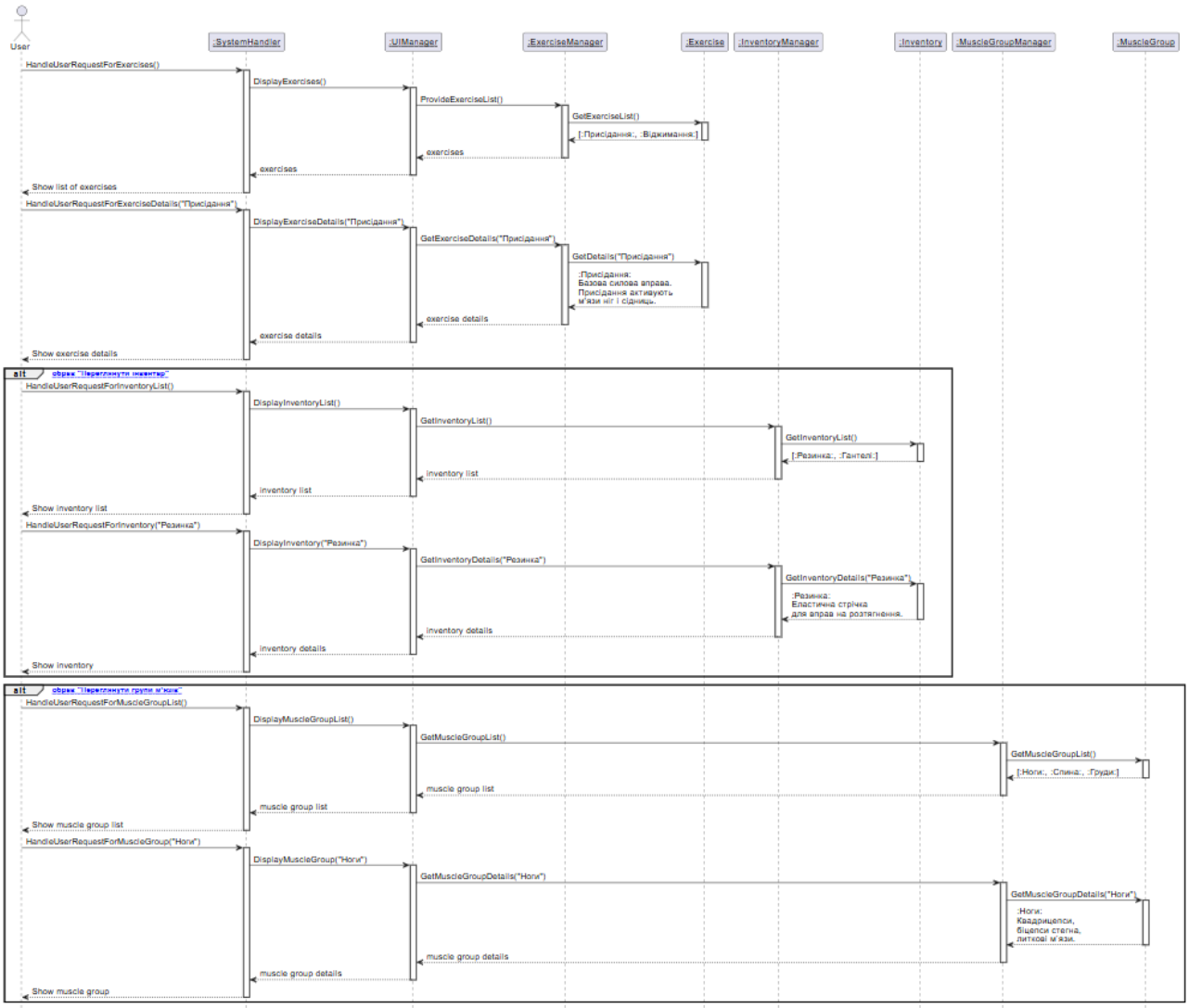


Рисунок 2.13 – Діаграма послідовності прецеденту «Переглянути вправи»

Дані діаграми взаємодії демонструють, як користувач взаємодіє із застосунком для перегляду розкладу тренувань та вправ, а також як система обробляє ці запити через послідовність викликів між ключовими компонентами.

Розробка логічної моделі даних мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Логічна модель даних програмного забезпечення – це структуроване графічне подання даних, яке відображає основні сутності, їх атрибути та зв’язки між ними відповідно до обраної моделі зберігання даних, незалежно від фізичної реалізації або конкретної платформи [16].

На рисунку 2.14 зображено логічну модель даних мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere». Вона дозволяє чітко виокремити ключові сутності, такі як користувачі, вправи, інвентар, тренування, програми тренувань, розклад тощо, а також визначити взаємозв’язки між ними. Це дає змогу сформувати основу для створення повноцінної та ефективної структури бази даних майбутнього застосунку.

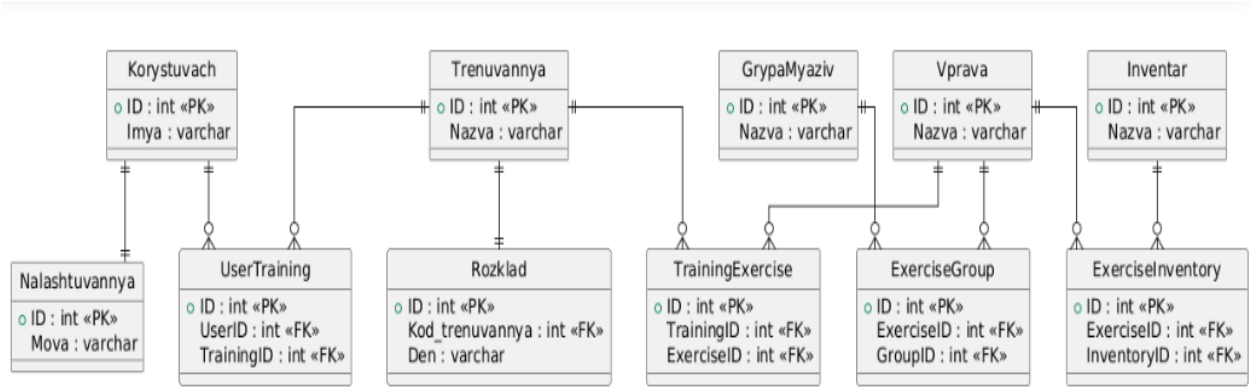


Рисунок 2.14 – Логічна модель даних мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Дана модель відображає структуру даних для системи керування тренуваннями користувачів. Вона включає сутності для збереження інформації про користувачів, тренування, розклад занять, вправи, інвентар та групи м’язів. Взаємозв’язки між сутностями відображають логіку, за якою користувачі виконують тренування, тренування містять вправи, а вправи пов’язані з інвентарем і цільовими групами м’язів. Також модель підтримує налаштування системи, пов’язані з тренуваннями, і відстежує статус виконання тренувань за розкладом. Використання типів даних з обмеженим вибором значень (наприклад, стать, ціль тренування, статус) забезпечує цілісність і правильність збереженої інформації.

2.3.3 Робочий проєкт мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Вибір та обґрунтування інструментів розробки мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

В якості засобів розробки було використано наступне:

− Середовище Unity. Unity – це багатоплатформовий інструмент та ігровий рушій для розробки відеоігор і застосунків. Програми, створені з його допомогою, можуть запускатися на настільних комп’ютерах, мобільних пристроях, ігрових консолях, а також на пристроях віртуальної та доповненої реальності, підтримуючи як 2D, так і 3D графіку [17].

Зазвичай ігрові рушії надають широкий набір функціональних можливостей, які застосовуються у різних типах ігор – від моделювання фізичних процесів до підтримки нормалей, динамічних тіней та інших графічних ефектів. Однак Unity вирізняється серед багатьох рушіїв двома ключовими перевагами. Першою перевагою є розвинене візуальне середовище, яке включає інструменти графічного моделювання, інтегровану систему розробки та ланцюг складання, що допомагає підвищити ефективність роботи розробників, особливо на етапах створення прототипів і тестування. Друга перевага – підтримка різних платформ, яка охоплює як широкий спектр пристроїв для розгортання (персональні комп’ютери, мобільні гаджети, ігрові консолі тощо), так і можливість розробки під різними операційними системами, зокрема Windows і Mac OS. Ще одною важливою перевагою є модульна система компонентів Unity, яка дозволяє створювати ігрові об'єкти шляхом поєднання окремих функціональних елементів. На відміну від традиційного об'єктно-орієнтованого підходу з механізмом успадкування, у Unity об’єкти формуються з незалежних модулів. Така гнучка структура значно полегшує створення прототипів, що є особливо важливим у процесі розробки ігор.

− Мова програмування C#. Це мова програмування з підтримкою об’єктно-орієнтованого підходу, створена компанією Microsoft для платформи

.NET. Вона поєднує зручність високорівневих мов, таких як Java, із продуктивністю низькорівневих мов, як-от C++. C# має сильну типізацію, підтримує автоматичне керування пам'яттю завдяки Garbage Collector і надає великий комплект інтегрованих бібліотек для виконання різноманітних задач. Асинхронне програмування через async/await дозволяє ефективно працювати з потоками, а LINQ спрощує роботу з колекціями та базами даних [18]. Завдяки .NET Core ця мова стала кросплатформною, що дозволяє розробляти програми для Windows, Linux і macOS. C# активно використовується у створенні десктопних додатків на Windows Forms і WPF, мобільних додатків через Xamarin і MAUI, веб-розробці на базі ASP.NET Core, а також в ігровій індустрії завдяки Unity. Це універсальна мова, яка підходить для широкого спектра завдань, особливо в екосистемі Microsoft та для розробки масштабованих застосунків.

− База даних Firebase. Кожна програма для Android може створювати і використовувати БД.

Firebase – це хмарна база даних, яка надає користувачам можливість зберігати та отримувати дані, а також пропонує зручні інструменти і методи для роботи з ними [19].

Firebase зберігає текстову інформацію у форматі JSON та забезпечує зручні інструменти для читання, оновлення й видалення даних. Крім того, сервіс підтримує функції реєстрації й автентифікації користувачів, зберігання сесій авторизованих осіб, а також зручне керування медіафайлами завдяки Cloud Storage, який спрощує доступ до них.

Усі зміни в базі даних негайно синхронізуються між усіма клієнтами або пристроями, що працюють з однією й тією ж базою. Інакше кажучи, оновлення в Firebase відбуваються в режимі реального часу.

Реалізація основних класів мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Мобільний застосунок розроблений мовою C# із використанням Unity як ігрового движка та Firebase як бекенд-сервісу для зберігання даних і автентифікації користувачів. Для управління залежностями та організації проєкту використовується стандартна структура проєкту Unity з розподілом коду по окремих класах і скриптах.

Застосунок реалізує логіку роботи з базою даних Firebase Realtime Database (або Firestore) через офіційний SDK Firebase для Unity, що дозволяє зчитувати, записувати та оновлювати дані користувачів, тренувань, розкладів та інших сутностей.

Нижче наведено приклад основного класу для ініціалізації підключення до Firebase та приклад класу для роботи з даними тренування.

FirebaseInit

```
using Firebase;
using Firebase.Database;
using Firebase.Extensions;
using UnityEngine;

public class FirebaseInit : MonoBehaviour
{
    public static DatabaseReference DBReference;

    void Start()
    {
        FirebaseApp.CheckAndFixDependenciesAsync().ContinueWithOnMainThread(task
=>
        {
            var dependencyStatus = task.Result;
            if (dependencyStatus == DependencyStatus.Available)
            {
                DBReference = FirebaseDatabase.DefaultInstance.RootReference;
                Debug.Log("Firebase Initialized successfully");
            }
            else
            {
                Debug.LogError($"Could not resolve all Firebase dependencies:
{dependencyStatus}");
            }
        });
    }
}
```

Training

```
using System;
using System.Threading.Tasks;
using Firebase.Database;
using UnityEngine;

[Serializable]
public class Training
{
    public string trainingId;
    public string name;
    public string description;

    public Training() { }

    public Training(string id, string name, string desc)
    {
        this.trainingId = id;
        this.name = name;
        this.description = desc;
    }
}

public class TrainingManager : MonoBehaviour
{
    private DatabaseReference trainingRef;

    void Start()
    {
        trainingRef = FirebaseInit.DBReference.Child("trainings");
    }

    public void SaveTraining(Training training)
    {
        string json = JsonUtility.ToJson(training);

trainingRef.Child(training.trainingId).SetRawJsonValueAsync(json).ContinueWithOnMa
inThread(task =>
        {
            if (task.IsCompleted)
            {
                Debug.Log("Training saved successfully");
            }
            else
            {
                Debug.LogError("Failed to save training: " + task.Exception);
            }
        });
    }

    public async Task<Training> GetTraining(string id)
    {
        var snapshot = await trainingRef.Child(id).GetValueAsync();
        if (snapshot.Exists)
        {
            string json = snapshot.GetRawJsonValue();
            return JsonUtility.FromJson<Training>(json);
        }
        else
        {
            Debug.LogWarning("Training not found");
            return null;
        }
```

```
    }
}
```

Представлені коди реалізують ініціалізацію Firebase у середовищі Unity та взаємодію з базою даних Realtime Database. Клас FirebaseInit відповідає за перевірку та налаштування залежностей Firebase, а також отримання кореневого посилання на базу даних. Клас Training забезпечує збереження та завантаження об’єктів типу Training, які серіалізуються у формат JSON і зберігаються в розділі «trainings» бази даних за відповідним ідентифікатором.

Повний код розробленого мобільного застосунку представлений в додатку Б.

Тестування мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Тестування мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere», проводилося за стратегією тестування «чорної скриньки». У межах тестування було виконано повну перевірку функціональності додатка.

Протокол тестування:

1) Перевірка доступності функціоналу перегляду тренувань.

Ідентифікатор: TC-WORKOUT-01.

Предмети тестування: Функціонал перегляду тренувань.

Специфікація вхідних даних: Натискання на кнопку перегляду тренувань.

Специфікація результатів: Розділ тренувань відкривається без помилок.

Тестове оточення: Android- застосунок.

Спеціальні процедурні вимоги: Відсутні.

Залежність з іншими тестами: Відсутні.

Дії користувача:

- Відкрити застосунок.
- Натиснути на кнопку «Тренування».
- Перевірити, що система відображає тренування для вибору.

2) Перевірка відображення лише запланованих тренувань у розділі майбутніх тренувань.

Ідентифікатор: TC-FUTURE-01.

Предмети тестування: Відображення списку майбутніх тренувань.

Специфікація вхідних даних: Відкриття розділу майбутніх тренувань.

Специфікація результатів: Відображаються лише заплановані тренування.

Тестове оточення: Android- застосунок.

Спеціальні процедурні вимоги: У базі мають бути заплановані, виконані та пропущені тренування.

Залежність з іншими тестами: Відсутні.

Дії користувача:

- Відкрити застосунок.
- Перейти в розділ «Майбутні тренування».
- Переконатися, що у списку відображаються лише заплановані тренування.

3) Перевірка доступу до списку м’язових груп.

Ідентифікатор: TC-MUSCLE-01Предмети тестування: Доступ до списку м’язових груп.

Специфікація вхідних даних: Виклик функціоналу перегляду м’язових груп

Специфікація результатів: Користувач отримує доступ до списку м’язових груп.

Тестове оточення: Android- застосунок.

Спеціальні процедурні вимоги: Відсутні.

Залежність з іншими тестами: Відсутні.

Дії користувача:

- Відкрити застосунок.
- Перейти до розділу перегляду м’язових груп.
- Переконатися, що список м’язових груп доступний для перегляду.

4) Перевірка доступу до списку інвентаря.

Ідентифікатор: TC-INVENTORY-01.

Предмети тестування: Відображення списку інвентаря.

Специфікація вхідних даних: Відкриття розділу інвентаря.

Специфікація результатів: Користувач має доступ до списку інвентаря.

Тестове оточення: Android- застосунок.

Спеціальні процедурні вимоги: Відсутні.

Залежність з іншими тестами: Відсутні.

Дії користувача:

- Відкрити застосунок.
- Перейти в розділ «Інвентар».
- Переконатися, що список інвентаря доступний для перегляду.

5) Перевірити, що у момент введення некоректних даних у поля реєстрації валідація спрацьовує та відображається відповідне повідомлення.

Ідентифікатор: TC-REG-03.

Предмети тестування: Форма авторизації.

Специфікація вхідних даних:

Email: b.pish@gmail.com.

Пароль: 12345@dfhgd (невірний пароль).

Специфікація результатів: Відображається повідомлення про помилку:

Дані некоректні, введіть ще раз, або зареєструйтесь.

Тестове оточення: Android пристрій.

Спеціальні процедурні вимоги: Відстні.

Залежність з іншими тестами: TC-REG-01.

Дії користувача:

- Відкрити застосунок.
- Перейти до екрану реєстрації.
- Заповнити поля:
  - Email: b.pish@gmail.com.
  - Пароль: 12345@dfhgd.
- Натиснути кнопку «Увійти».
- Перевірити, чи з'являється повідомлення про помилки.

У таблиці 2.30 представлено тестування мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere».

Таблиця 2.30 – Тестування мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere».

Дія	Очікуваний результат	Результат
Перевірка доступності функціоналу перегляду тренувань.	Розділ тренувань відкривається без помилок.	Розділ тренувань відкривається успішно.
Перевірка відображення лише запланованих тренувань у розділі майбутніх тренувань.	Відображаються лише заплановані тренування.	Відображаються лише заплановані тренування.
Перевірка доступу до списку м’язових груп.	Користувач отримує доступ до списку м’язових груп.	Доступ до списку м’язових груп успішно надано.
Перевірка доступу до списку інвентаря.	Користувач має доступ до списку інвентаря.	Список інвентаря доступний для перегляду.
Перевірка відображення відповідного повідомлення у момент введення некоректних даних у поля авторизації.	Виводиться повідомлення про некоректність введених даних.	Користувач бачить повідомлення про некоректність введених даних.

На рисунку 2.15 продемонстровано результат тестування відображення відповідного повідомлення у момент введення некоректних даних у поля авторизації мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere».

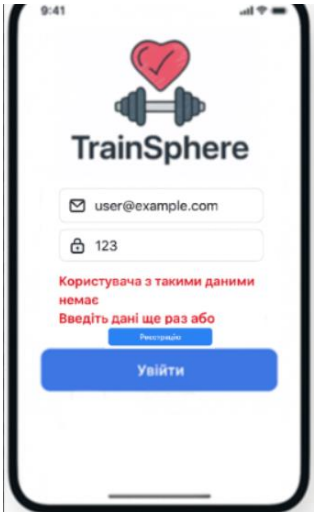


Рисунок 2.15 – Результат тестування відображення відповідного повідомлення у момент введення некоректних даних у поля авторизації мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere».

В повному обсязі тестування програмного забезпечення наведено у програмі та методиці випробувань у додатку Д.

3 РЕЗУЛЬТАТИ РОЗРОБКИ МОБІЛЬНОГО ЗАСТОСУНКУ ПІД ОС ANDROID ДЛЯ ДОМАШНІХ ТРЕНУВАНЬ «TRAINSPHERE»

В межах кваліфікаційної роботи було вирішено прикладну задачу інженерії програмного забезпечення, яка має складну структуру та містить елементи невизначеності. Результатом стала розробка мобільного застосунку

На початковому етапі виконано аналіз практичної задачі інженерії програмного забезпечення, аналіз існуючих програмних рішень та зроблено постановку задачі на розробку мобільного застосунку під ОС Android для домашніх тренувань «TrainSphere».

Далі було проведено аналіз вимог до мобільного застосунку, де було розроблено модель варіантів використання та розроблені їх специфікації. Наступним етапом була розробка його архітектури та виконання поетапного проєктування: ескізного, технічного та робочого. Під час ескізного проєктування розроблено діаграми діяльності, концептуальну модель даних та ескізи інтерфейсу користувача. У рамках технічного проєкту побудовано діаграму класів з специфікаціями кожного класу, діаграми послідовності, а також логічну модель даних мобільного застосунку. На етапі робочого проєкту визначено інструменти розробки, виконано програмування та проведено тестування мобільного застосунку.

Програмне забезпечення було реалізоване мовою програмування C# з використанням платформи Unity, яка надає широкі можливості для створення кросплатформених застосунків із графічним інтерфейсом. Unity дозволяє ефективно поєднувати логіку, візуальні елементи та взаємодію з користувачем, що є особливо важливим для створення сучасного та зручного інтерфейсу.

У якості бази даних для зберігання та обробки інформації про користувачів і пов’язані з ними дані використано хмарну платформу Firebase. Ця система забезпечує надійне та швидке зберігання інформації, підтримку автентифікації користувачів, синхронізацію даних у реальному часі, а також можливість

масштабування в разі зростання обсягів даних або кількості користувачів. Взаємодія між Unity-додатком і Firebase здійснюється за допомогою відповідного SDK, що забезпечує безпечний та безперебійний процес передачі даних між клієнтом і сервером.

Таким чином, обрані технології та засоби реалізації забезпечують швидкодію, зручність у використанні, розширюваність та відповідність сучасним вимогам до програмного забезпечення.

На момент реалізації кількість рядків коду становить орієнтовно близько 1500, що є достатнім для реалізації основного функціоналу без надмірної складності. Після компіляції та збірки мобільний застосунок має приблизний розмір у межах 50 мегабайт, що є цілком прийнятним для зберігання на мобільних пристроях і не викликає значного навантаження на систему.

Зовнішній вигляд головного меню мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» наведено на рисунку 3.1.

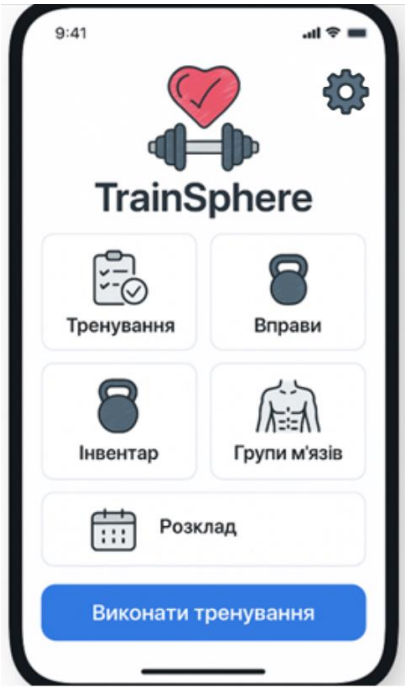


Рисунок 3.1 – Головне вікно мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

Меню виконання тренувань мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» представлене на рисунку 3.2.

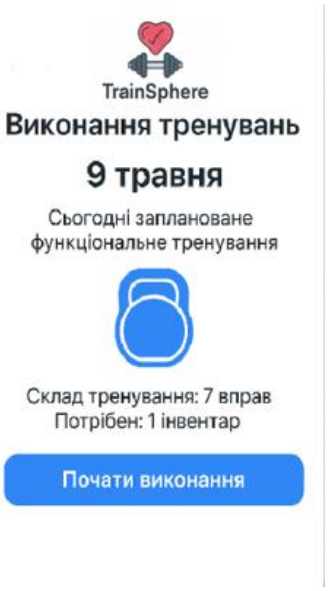


Рисунок 3.2 – Меню виконання тренувань мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere»

В рамках реалізації проєкту також була підготовлена вся необхідна програмна документація. Технічне завдання на розробку мобільного застосунку під ОС Android для домашніх тренувань «Trainsphere» наведене в додатку А, код мобільного застосунку наведено в додатку Б, опис мобільного застосунку представлено в додатку В. Для забезпечення зручності користувача було створено керівництво користувача для мобільного застосунку, яке міститься в додатку Г. Крім того, у додатку Д наведено програму та методику випробувань, що застосовувалися для перевірки працездатності застосунку.

Мобільний застосунок був випробуваний згідно програми та методики випробувань наведених у додатку Д.

Результати випробувань підтвердили, що мобільний застосунок повністю виконує поставлені завдання та відповідає вимогам, визначеним у технічному завданні на розробку. Усі недоліки, виявлені в процесі тестування, були задокументовані в протоколах і своєчасно усунуті до моменту впровадження програми в експлуатацію.

4 ОХОРОНА ПРАЦІ

4.1 Вступ

Охорона праці передбачає, щоб трудові умови на робочому місці підприємства співпадали із умовами, вказаними у законодавстві.

Вона є ключовим елементом трудових відносин і спрямована на підтримку безпечних і сприятливих для здоров'я умов праці. Вона охоплює комплекс правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів, що мають на меті попередження виробничого травматизму, професійних захворювань та забезпечення комфортних умов праці.

Відповідно до статті 43 Конституції України, кожен громадянин має право працювати в умовах, які гарантують безпеку, належний рівень комфорту та охорону здоров'я [20]. Це право гарантується державою через прийняття відповідних законів і нормативно-правових актів.

Основним нормативно-правовим актом охорони праці є Закон України «Про охорону праці». Відповідно до Закону України «Про охорону праці», роботодавець повинен надати працівникам засоби індивідуального захисту. Також роботодавець повинен організовувати навчання та інструктажі з охорони праці.

Важливою складовою правового регулювання у сфері охорони праці є Кодекс законів про працю України (КЗпП України), що встановлює обов'язки роботодавця щодо забезпечення безпечних умов праці. Стаття 153 КЗпП зобов'язує роботодавця впроваджувати сучасні засоби техніки безпеки та гігієни праці [21].

4.2 Обладнання робочого місця

Зону простору, в якій на підприємстві триває робочий процес із участю одного працівника або групи працівників, і яка має задовільні умови для праці, називають робочим місцем.

Під час написання кваліфікаційної роботи робоче місце комплектується невеликим столом для однієї людини і стільцем. Дані елементи робочого місця мають бути розстановлені із забезпеченням відповідності нормативним вимогам.

Стіл укомплектований двома горизонтальними поверхнями різної висоти – основною робочою та додатковою. Розміри ширини і глибини обох поверхонь дозволяють виконувати роботу в межах моторного поля користувача. Висота обох поверхонь має регулюватися в діапазоні від 46 см до 76 см. Для столу, що використовується на комп'ютеризованому робочому місці, передбачене кріплення до підлоги.

Під час організації робочого місця враховано ключові вимоги: відстань до стін із вікнами становить приблизно 100 см, до інших стін – близько 50 см; відстань між боковими поверхнями відеомоніторів не менше 120 см. Склад ПК, за яким проходило написання кваліфікаційної роботи, та технічні характеристики:

- У системному блоці розміщені материнська плата (МП), пристрої для зберігання даних, а також контролери монітора та локальної обчислювальної мережі (ЛОМ) і аудіопристрою (звукову карту). На материнській платі розташовані ЦП (центральний процесор) та ОЗП (оперативний запам'ятовуючий пристрій); монітор;
- клавіатура;
- комп'ютерна миша;
- гарнітура, що включає навушники та мікрофон.

Різноманітні пристрої можна приєднувати до системного блоку комп'ютера, що дозволяє збільшити його можливості. Для цього використовуються спеціальні роз'єми, які зазвичай розташовані на задній панелі системного блоку. До периферійних пристроїв, крім монітора, клавіатури, принтера і комп'ютерної миші, також належать модеми, факс-модеми, сканери та інші подібні пристрої.

Підключення таких пристроїв здійснюється через відповідні кабельні інтерфейси. Для запобігання помилковому підключенню роз'єми мають різну форму, що не дозволяє вставити кабель у невідповідний гнізд. Деякі кабелі додатково фіксуються гвинтами, які необхідно закручувати вручну або за

допомогою викрутки, щоб кабель міцно тримався в роз'ємі.

Під час увімкненого комп'ютера не слід підключати або відключати кабелі пристроїв, оскільки це може призвести до пошкодження системи. Винятком є пристрої з інтерфейсом USB, які можна підключати без вимкнення. Деякі пристрої також можуть встановлюватися безпосередньо всередину системного блоку.

Під час взаємодії з комп'ютером необхідно правильно відпочивати і працювати. Інакше виникне напруга очей у разі виникнення скарг на незадоволення роботою, головний біль та підвищену дратівливість, порушення сну та інша симптоматика.

Робоча поза сидячи забезпечує мінімальне стомлення програміста. Раціональне облаштування робочого місця передбачає чіткий порядок і постійне розміщення інструментів, обладнання та документації. Найчастіше використовувані предмети розташовуються в зоні легкої досяжності. Положення екрану визначається кутом огляду: центр екрану має бути розташований приблизно на 20 градусів нижче горизонтальної лінії зору, причому екран розміщується перпендикулярно напрямку погляду.

Також має бути передбачена опція регулювання екрану:

- по висоті – 3 см;
- по нахилу – від –10 до +20 відносно вертикалі;
- у лівий та правий бокові напрямки.

Важливими для ефективної та якісної роботи за комп'ютером є розмір символів, їх щільність, контрастність а також узгодженість рівнів яскравості між символами та фоном екрану. За відстані 60–80 см від очей до екрана висота символів повинна становити не менше 3 мм. Рекомендоване співвідношення ширини до висоти символу – 3:4, а інтервал між символами має бути в межах 15–20 % їх висоти. Рівень контрасту має складати від 1:2 до 1:15.

Якщо екран розміщений надто високо, а очі широко розкриті, може порушуватися природний механізм моргання – очі залишаються напіввідкритими та недостатньо зволожуються слізною рідиною, що призводить до швидкої втоми та дискомфорту. Якщо екран розміщений занадто високо, очі залишаються широко

відкритими, моргання порушене – повне закриття не відбувається, що призводить до недостатнього зволоження слізною рідиною, що призводить до швидкої втоми та дискомфорту.

4.3 Електробезпека приміщення

Електробезпека – це комплекс заходів і засобів (як організаційних, так і технічних), спрямованих на охорона людей від шкідливого впливу електричного струму, електричної дуги, електричного поля та статичної електрики. У приміщенні, де проводилася розробка, застосовується чотирипровідна трифазна електромережа із заземленим нульовим проводом (нейтраллю) напруга становить 380/220 В. Це означає, що фазна напруга (між фазою та нулем) дорівнює 220 В, а міжфазна або лінійна напруга (між двома фазами) – 380 В.

Категорія умов без підвищеної небезпеки електротравматизму, оскільки відсутні такі фактори, які підвищують ризик, зокрема: температура повітря понад 35°С, вологість вище 75%, струмопровідна підлога та струмопровідний пил, також сюди відносять ймовірність одночасного контакту обслуговуючого персоналу з металевим корпусом електрообладнання та металевими конструкціями, які пов’язані із землею. Також відсутні чинники особливої небезпеки, до яких належать: вологість повітря практично насичене вологою з конденсацією на поверхнях обладнання та будівельних елементів (100%), а також наявність хімічно активного середовища, яке пошкоджує ізоляцію, або біологічне середовище (наприклад, пліснява), що накопичується на обладнанні й електропровідних частинах.

З метою запобігання електротравмам у приміщенні впроваджуються наступні заходи:

1) Заходи технічного характеру, що запобігають електротравмам при контакті з струмоведучими елементами електроустаткування – це ізоляція електропровідних частин відповідно до вимог чинних нормативних документів;

2) Технічні рішення для запобігання електротравмам у разі появи напруги

на непротічних струм елементах електроустаткування включають застосування захисного заземлення з використанням природних заземлювачів.

4.4 Мікроклімат

Основними показниками мікроклімату, які підлягають контролю та регулюванню, є температура повітря (t°C), його відносна вологість (W, %), швидкість повітряного потоку (м/с) та рівень теплового випромінювання (Вт/м²). Умови мікроклімату на робочому місці дослідника визначаються і підтримуються згідно з вимогами ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

Згідно з енерговитратами, робота розробника відноситься до категорії 1а.

Для робіт категорії 1а нормативні параметри мікроклімату наведені у таблиці 4.1 [22].

Таблиця 4.1 – Параметри мікроклімату [22]

Період року	Допустимі		
	t,°C	W, %	V, м/с
Теплий	22-28	55	0,1-0,2
Холодний	21-25	75	0,1

З метою досягнення необхідних за нормами параметрів мікроклімату в приміщенні заплановано:

1) В холодний сезон будівля опалюється за допомогою централізованої парової системи опалення.

2) Допустимі метеорологічні умови праці забезпечуються за рахунок системи вентиляції з припливом і витяжкою та регулярного провітрювання приміщення.

4.5 Виробниче освітлення

Освітлення повинно задовольняти ряд гігієнічних норм: бути достатнім за яскравістю, рівномірним, не спричиняти засліплення очей і не створювати надмірної контрастності на робочій поверхні.

Приміщення освітлюється за допомогою штучного світла.

Параметри освітленості для штучного освітлення та коефіцієнт природного освітлення (КПО) в умовах III світлового кліматичного поясу наведені в таблиці 4.2 [22]. Вони враховують дуже високу точність зорової роботи (розряд 2, підрозряд г), великий контраст між об'єктом і фоном, а також світлий фон.

Таблиця 4.2 – Норми освітленості в приміщенні [22]

Характеристика зорової роботи	Найменший розмір об'єкта розрізнювання	Розряд зорової роботи	Підрозряд зорової роботи	Контраст об'єкта розрізнення з фоном	Характеристика фона	Освітленість, лк		КПО, $e_н$,%			
						Штучне освітлення		Природнє освітлення		Сумісне освітлення	
						Комбіноване	Загальне	Верхнє або верхнє і бокове	Бокове	Верхнє або верхнє і бокове	Бокове
Дуже високої точності	Від 0,15 до 0,3	2	г	великий	світлий	1000	300	7	20,5	4,2	10,5

Для створення необхідного рівня освітлення здійснюються такі заходи:

1) Повне використання природного світла з бічного боку.

2) Систематичне очищення скла від забруднень здійснюється щонайменше двічі на рік.

3) Загальне освітлення штучного типу організоване за допомогою люмінесцентних ламп.

4.6 Виробничий шум

У робочому просторі, де проходила розробка, джерелами шуму є персональний комп'ютер (система охолодження, жорсткий диск, вентилятор) та периферійні пристрої. Тут спостерігаються шуми як механічного, так і аеродинамічного походження, а також широкосмугові шуми з аперіодичним підсиленням під час експлуатації принтерів.

Санітарні вимоги до виробничого шуму, ультразвуку та інфразвуку встановлені у ДСН 3.3.6.037-99. Для конкретних умов роботи, з урахуванням її характеру та типу шуму, допустимі рівні звукового тиску мають відповідати гігієнічним стандартам (ГС). Крім того, рівень звуку не повинен перевищувати 50 дБА – див. таблицю 4.3 [22].

Таблиця 4.3 – Допустимі рівні звукового тиску і рівні звуку для постійного широкополосного шуму [22]

Характер робіт	Допустимі рівні звукового тиску (дБ) в стандартизованих октавних смугах зі середньогеометричними частинами (Гц)									Допустимий рівеньзвуку, дБА
	32	63	25	50	500	1000	2000	4000	8000	
Виробничі приміщення	86	71	61	54	49	45	42	40	38	50

Для забезпечення необхідних за нормативами параметрів акустичного середовища в виробничих приміщеннях рівні звукового тиску повинні не перевищувати встановлених нормативних меж, зазначених в стандартних октавних смугах частот. Зокрема, допустимі значення зменшуються зі зростанням частоти: від 86 дБ на частоті 32 Гц до 38 дБ на частоті 8000 Гц. Загальний допустимий рівень шуму складає 50 дБА. Дотримання цих меж є важливим для збереження здоров'я працівників та підвищення ефективності трудового процесу.

4.7 Пожежна безпека

У приміщенні, де виконувалася розробка, використовуються лише негорючі холодні матеріали та речовини, через що його класифікують як категорію «Д» за ступенем пожежної та вибухонебезпечності. Єдине джерело пожежної небезпеки – це електричні кабелі, що живлять обладнання, відповідне вимогам цієї категорії.

За показниками вогнестійкості приміщення належить до другої категорії відповідно до вимог ДБН В.1.1.7-2002. Робоча зона дослідника належить до категорії вибухонебезпечного класу В-IIа та пожежонебезпечного класу П-IIа, оскільки вибухонебезпечна концентрація пилу та волокон можлива лише за умов аварійної ситуації або технічної несправності.

4.8 Висновки

Приміщення повністю відповідає умовам, передбаченим пожежною безпекою та іншим нормам. Рекомендується приділяти увагу підтриманню засобів протипожежного захисту в належному робочому стані та оперативному повідомленню пожежної служби про будь-які несправності пожежного обладнання.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи мета, яка ставилась її початку, а саме, розв’язання практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, і полягала в розробці мобільного застосунку під ОС Android для домашніх тренувань «TrainSphere», була досягнута.

Для досягнення поставленої мети були вирішені наступні задачі:

- проведено аналіз практичної задачі інженерії програмного забезпечення;
- проведено аналіз аналогічного існуючого ПЗ та обґрунтування необхідності розробки власного ПЗ;
- сформовано вимоги до розроблюваного продукту та розроблено технічне завдання на розробку мобільного застосунку під ОС Android для домашніх тренувань «TrainSphere»;
- розроблено архітектуру програмного забезпечення;
- проведено розробку ескізного, технічного та робочого проєкту програмного забезпечення мобільного застосунку під ОС Android для домашніх тренувань «TrainSphere»;
- розроблено документацію до програмного продукту.

У процесі розробки мобільного застосунку «Trainsphere» для домашніх тренувань під операційну систему Android були використані різні інструменти, зокрема Unity, мова програмування C#, а також хмарна база даних Firebase. Ці технології забезпечили реалізацію функціоналу, необхідного для створення зручного та інтуїтивно зрозумілого інтерфейсу користувача, персоналізації тренувань, збереження прогресу та синхронізації даних через Інтернет.

Мобільний застосунок «Trainsphere» був успішно розроблений для надання користувачам можливості ефективно проводити домашні тренування. Також було

створено базу даних, яка забезпечує зберігання та обробку необхідної інформації для функціонування мобільного застосунку.

У результаті отримали програмний продукт, який допоможе користувачам створити ефективний план тренувань, який відповідає їхнім цілям і рівню підготовки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. «Бібліотека вправ». URL: https://play.google.com/store/apps/details ?id=softin.my.fast.fitness&hl=uk (дата звернення 09.02.2025).

2. «Тітан». URL: https://play.google.com/store/apps/details?id=com. powerups.titan&hl=uk (дата звернення 09.02.2025).

3. «Freeletics». URL: https://play.google.com/store/apps/details? id=com.freeletics.lite&hl=uk (дата звернення 09.02.2025).

4. Як будувати UML-діаграми. URL: https://dou.ua/forums/topic/40575 (дата звернення 09.02.2025).

5. Проєктування програмного забезпечення засобами UML. URL: https://www.ba.in.ua/2024/02/23/zastosuvannya-tehniky-uml-diagrama-variantivvy

korystannya-use-case-diagram/ (дата звернення 09.02.2025).

6. Клієнт-серверна архітектура. URL: https://training.qatestlab.com/ blog/technical-articles/client-server-architecture/ (дата звернення 09.02.2025).

7. Общие сведения ASP.NET Core MVC. URL: https://learn.microsoft.com/ru-ru/aspnet/core/mvc/overview?view=aspnetcore-9.0 (дата звернення 09.02.2025).

8. Повне розуміння діаграми компонентів UML за допомогою легкого методу. URL: https://www.mindonmap.com/uk/blog/uml-component-diagram/ (дата звернення 09.02.2025).

9. Що таке діаграма компонентів в UML? URL: https://www.guru99.com/uk/component-diagram-uml-example.html (дата звернення 09.02.2025).

10. Діаграма розгортання: Підручник з UML із ПРИКЛАДОМ. URL: https://www.guru99.com/uk/deployment-diagram-uml-example.html (дата звернення 09.02.2025).

11. Діаграма діяльності в UML: символ, компоненти та приклад. URL: https://www.guru99.com/uk/uml-activity-diagram.html (дата звернення 09.02.2025).

12. Концептуальні, логічні та фізичні моделі даних. URL: https://data-life-

ua.com/db/kontseptualni-lohichni-ta-fizychni-modeli-danykh/ (дата звернення 09.02.2025).

13. Для чого потрібні UML діаграми? URL: https://foxminded.ua/uml-diagramy/ (дата звернення 09.02.2025).

14. Що таке діаграми класів? URL: https://ua5.org/oop/392-diagrami-klasiv.html (дата звернення 09.02.2025).

15. Діаграма послідовності (Sequence Diagrams). URL: https://www.maxzosim.com/sequence-diagrams/ (дата звернення 09.02.2025).

16. Логічна модель даних. URL: https://data-life-ua.com/db/lohichna-model-danykh/ (дата звернення 09.02.2025).

17. Що таке Unity? URL: https://lemon.school/blog/shho-take-unity (дата звернення 09.02.2025).

18. Документація по C#. URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/ (дата звернення 09.02.2025).

19. Що таке Firebase? URL: https://qagroup.com.ua/publications/what-is-firebase/ (дата звернення 09.02.2025).

20. Конституція України – Розділ II. URL: https://www.president.gov.ua/ua/documents/constitution/konstituciya-ukrayini-rozdil (дата звернення 09.02.2025).

21. Кодекс законів про працю України Стаття 153. Створення безпечних і нешкідливих умов праці. URL: https://kodeksy.com.ua/kodeks_zakoniv_pro_pratsyu_ukraini/statja-153.htm (дата звернення 09.02.2025).

22. Створення безпечних і нешкідливих умов праці. Державний нагляд за охороною праці. URL: https://legalaid.wiki/index.php (дата звернення 09.02.2025).

23. Начало разработки под Android. URL: https://docs.unity3d.com/ru/530/Manual/android-GettingStarted.html (дата звернення 09.02.2025).

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ МОБІЛЬНОГО ЗАСТОСУНКУ ПІД ОС ANDROID ДЛЯ ДОМАШНІХ ТРЕНУВАНЬ «TRAINSPHERE»

Вступ

Повна назва програмного продукту: «Мобільний застосунок під ОС Android для домашніх тренувань «TrainSphere»».

Галузь застосування: фізичні тренування, покращення фізичної форми, схуднення.

1 Підстави для розробки

Підставою для розробки даного програмного забезпечення є завдання кваліфікаційної роботи на здобуття ступеня вищої освіти «бакалавр».

2 Призначення розробки

2.1 Експлуатаційне призначення програми

Експлуатаційне призначення програми «TrainSphere» полягає в тому, щоб забезпечити користувачам зручний і простий у використанні інструмент для планування та виконання тренувань. Мобільний застосунок повинен забезпечувати доступність і комфортне користування для осіб незалежно від їхнього рівня фізичної форми чи попереднього досвіду.

2.2 Функціональне призначення

Функціональне призначення програми «TrainSphere» полягає в тому, щоб допомогти користувачам створити ефективний план тренувань, який відповідає їхнім цілям і рівню підготовки. Програма також надасть користувачам інформацію про різні типи вправ, їхню ефективність і техніку виконання.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Функції адміністратора:

- додавати/редагувати інформацію про тренування;
- додавати/редагувати інформацію про інвентар;
- додавати/редагувати інформацію про вправи;
- додавати/редагувати інформацію про групи м’язів;
- додавати/редагувати інформацію про розклад тренувань;
- реєструватись/авторизуватись.

Функції користувача:

- переглядати інформацію про тренування;
- виконувати тренування;
- переглядати інформацію про інвентар;
- переглядати інформацію про вправи;
- переглядати інформацію про групи м’язів;
- переглядати інформацію про розклад;
- реєструватись/авторизуватись;
- робити налаштування профілю.

3.1.2 Вимоги до організації вхідних та вихідних даних

Вхідні дані вводяться в форму на екрані та заносяться у БД, вихідні дані будуть виводитись на екран з БД.

Вхідні дані:

- дані про користувача (Ім'я, Зріст, Вага, Стать ,Ціль тренувань);
- інформація про вправи (Назва вправи, Опис вправи);
- інформація про тренування(Назва тренування, Опис тренування);
- інформація про розклад (Статус тренування, День);
- інформація про групи м'язів (Назва м'язів, Опис м’язів);
- інформація про інвентар (Назва інвентаря, Опис інвентаря);

Вихідні дані:

- дані користувача(Ім'я, Зріст, Вага, Стать, Ціль тренувань);
- дані вправ(Назва вправи, Опис вправи);
- списки тренувань(Назва тренування, Опис тренування);
- розклад(Статус тренування, День);
- дані про групи м'язів(Назва м'язів, Опис м’язів);
- дані про інвентар(Назва інвентаря, Опис інвентаря).

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійного (сталого) функціонування програми

Програма повинна забезпечувати відновлення коректної роботи після перезавантаження, викликаного збоєм програми.

3.3 Вимоги до умов експлуатації

3.3.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватись задані характеристики, повинні відповідати вимогам, що пред’являються до технічних засобів в частині умов їх експлуатації.

3.3.2 Вимоги кваліфікації користувачів

Користувач повинен мати базові навички роботи з мобільним телефоном.

3.4 Вимоги до складу та параметрів технічних засобів

Технічні вимоги :

Процесор: не нижче MediaTek MT6737 або Snapdragon 435;

Оперативна пам’ять: 2ГБ;

Дисковий простір: до 300 МБ;

Інтернет-з’єднання: присутне.

3.5 Вимоги до інформаційної та програмної сумісності

Системні програмні засоби, які використовуються програмою, повинні бути представлені ліцензійною версією операційної системи не нижче чим Android 8.0.

Вимоги до захисту інформації та програм не представляються.

3.6 Спеціальні вимоги

Програма має надавати користувачу можливість взаємодії через графічний інтерфейс, створений з урахуванням рекомендацій виробника операційної системи.

4 Вимоги до програмної документації

Склад програмної документації повинен включати в себе :

- технічне завдання;
- програму та методику випробувань;
- керівництво користувача;
- опис програми;
- код програми.

5 Техніко-економічні показники

У даній роботі техніко-економічні показники не розраховуються.

6 Стадії та етапи розробки

Етапи розробки продемонстровані у таблиці А.1.

Таблиця А.1 – Етапи розробки

Номер	Назва етапів розробки	Терміни виконання
1	2	3
1	Аналіз практичної задачі інженерії ПЗ	07.04.2025
2	Аналіз вимог до ПЗ	14.04.2025
3	Розробка архітектури ПЗ	21.04.2025

Кінець таблиці А.1

1	2	3
4	Розробка проєкту ПЗ	05.05.2025
5	Реалізація та тестування ПЗ	12.05.2025

7 Порядок контролю та приймання

Контроль процесу аналізу та проєктування кожного компонента програмного забезпечення проводиться на всіх етапах з урахуванням вимог, визначених технічним завданням. Кожна стадія розробки має бути виконана в установлені терміни та погоджена відповідними сторонами.

Приймання виконується згідно з затвердженою програмою та методикою тестування і документується відповідними записами у протоколі проведення тестування. У разі виявлення помилок під час приймання складається акт про виявлені недоліки, який підписують представники замовника і розробника, а також затверджують керівники обох організацій. Розробник зобов’язаний усунути виявлені помилки протягом максимум двох тижнів та повідомити замовника про готовність до повторного приймання.

ДОДАТОК Б – КОД МОБІЛЬНОГО ЗАСТОСУНКУ ПІД ОС ANDROID ДЛЯ ДОМАШНІХ ТРЕНУВАНЬ «TRAINSPHERE»

UserProfile.cs

```
using System;

[Serializable]
public class UserProfile
{
    public string UserId;
    public string Name;
    public int Age;
    public float Weight;
    public float Height;
    public string TrainingGoal;

    public UserProfile() { }

    public UserProfile(string userId, string name, int age, float weight, float
height, string goal)
    {
        UserId = userId;
        Name = name;
        Age = age;
        Weight = weight;
        Height = height;
        TrainingGoal = goal;
    }
}
```

UserManager.cs

```
using System;
using System.Threading.Tasks;
using Firebase.Auth;
using Firebase.Firestore;
using UnityEngine;

public class UserManager
{
    private FirebaseAuth auth;
    private FirebaseFirestore db;

    public UserProfile CurrentUser { get; private set; }

    public UserManager()
    {
        auth = FirebaseAuth.DefaultInstance;
        db = FirebaseFirestore.DefaultInstance;
    }

    public async Task<bool> RegisterUser(string email, string password, UserProfile
profile)
    {
        try
        {
```

```
            var                userCredential                =                await
auth.CreateUserWithEmailAndPasswordAsync(email, password);
            profile.UserId = userCredential.User.UserId;
            CurrentUser = profile;
            await SaveUserProfile(profile);
            Debug.Log("User registered: " + profile.Name);
            return true;
        }
        catch (Exception e)
        {
            Debug.LogError("RegisterUser error: " + e.Message);
            return false;
        }
    }

    public async Task<bool> LoginUser(string email, string password)
    {
        try
        {
            var userCredential = await auth.SignInWithEmailAndPasswordAsync(email,
password);
            await LoadUserProfile(userCredential.User.UserId);
            Debug.Log("User logged in: " + CurrentUser.Name);
            return true;
        }
        catch (Exception e)
        {
            Debug.LogError("LoginUser error: " + e.Message);
            return false;
        }
    }

    public async Task SaveUserProfile(UserProfile profile)
    {
        var docRef = db.Collection("users").Document(profile.UserId);
        await docRef.SetAsync(profile);
    }

    public async Task LoadUserProfile(string userId)
    {
        var docRef = db.Collection("users").Document(userId);
        var snapshot = await docRef.GetSnapshotAsync();
        if (snapshot.Exists)
        {
            CurrentUser = snapshot.ConvertTo<UserProfile>();
        }
    }
}
```

Training.cs

```
using System;

[Serializable]
public class Training
{
    public string TrainingId;
    public string Title;
    public string Description;

    public Training() { }

    public Training(string id, string title, string desc)
```

```
    {
        TrainingId = id;
        Title = title;
        Description = desc;
    }
}
```

TrainingManager.cs

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Firebase.Firestore;
using UnityEngine;

public class TrainingManager
{
    private FirebaseFirestore db;

    public List<Training> Trainings = new List<Training>();

    public TrainingManager()
    {
        db = FirebaseFirestore.DefaultInstance;
    }

    public async Task<List<Training>> GetTrainingsAsync()
    {
        var snapshot = await db.Collection("trainings").GetSnapshotAsync();
        Trainings.Clear();

        foreach (var doc in snapshot.Documents)
        {
            var training = doc.ConvertTo<Training>();
            Trainings.Add(training);
        }

        return Trainings;
    }

    public async Task AddOrUpdateTraining(Training training)
    {
        var docRef = db.Collection("trainings").Document(training.TrainingId);
        await docRef.SetAsync(training);
    }

    public async Task DeleteTraining(string trainingId)
    {
        var docRef = db.Collection("trainings").Document(trainingId);
        await docRef.DeleteAsync();
    }
}
```

Schedule.cs

```
using System;

[Serializable]
public class Schedule
{
    public string ScheduleId;
```

```
    public string Title;
    public DateTime Date;

    public Schedule() { }

    public Schedule(string id, string title, DateTime date)
    {
        ScheduleId = id;
        Title = title;
        Date = date;
    }
}
```

ScheduleManager.cs

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Firebase.Firestore;
using UnityEngine;

public class ScheduleManager
{
    private FirebaseFirestore db;

    public List<Schedule> Schedules = new List<Schedule>();

    public ScheduleManager()
    {
        db = FirebaseFirestore.DefaultInstance;
    }

    public async Task<List<Schedule>> GetSchedulesAsync()
    {
        var snapshot = await db.Collection("schedules").GetSnapshotAsync();
        Schedules.Clear();

        foreach (var doc in snapshot.Documents)
        {
            var schedule = doc.ConvertTo<Schedule>();
            Schedules.Add(schedule);
        }

        return Schedules;
    }

    public async Task AddOrUpdateSchedule(Schedule schedule)
    {
        var docRef = db.Collection("schedules").Document(schedule.ScheduleId);
        await docRef.SetAsync(schedule);
    }

    public async Task DeleteSchedule(string scheduleId)
    {
        var docRef = db.Collection("schedules").Document(scheduleId);
        await docRef.DeleteAsync();
    }
}
```

Inventory.cs

```
using System;

[Serializable]
public class Inventory
{
    public string ItemId;
    public string Name;
    public int Quantity;

    public Inventory() { }

    public Inventory(string id, string name, int quantity)
    {
        ItemId = id;
        Name = name;
        Quantity = quantity;
    }
}
```

InventoryManager.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using Firebase.Firestore;
using UnityEngine;

public class InventoryManager
{
    private FirebaseFirestore db;

    public List<Inventory> Items = new List<Inventory>();

    public InventoryManager()
    {
        db = FirebaseFirestore.DefaultInstance;
    }

    public async Task<List<Inventory>> GetInventoryAsync()
    {
        var snapshot = await db.Collection("inventory").GetSnapshotAsync();
        Items.Clear();

        foreach (var doc in snapshot.Documents)
        {
            var item = doc.ConvertTo<Inventory>();
            Items.Add(item);
        }

        return Items;
    }

    public async Task AddOrUpdateItem(Inventory item)
    {
        var docRef = db.Collection("inventory").Document(item.ItemId);
        await docRef.SetAsync(item);
    }

    public async Task DeleteItem(string itemId)
    {
        var docRef = db.Collection("inventory").Document(itemId);
        await docRef.DeleteAsync();
    }
}
```

Exercise.cs

```
using System;

[Serializable]
public class Exercise
{
    public string ExerciseId;
    public string Name;
    public string Description;

    public Exercise() { }

    public Exercise(string id, string name, string desc)
    {
        ExerciseId = id;
        Name = name;
        Description = desc;
    }
}
```

ExerciseManager.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using Firebase.Firestore;
using UnityEngine;

public class ExerciseManager
{
    private FirebaseFirestore db;

    public List<Exercise> Exercises = new List<Exercise>();

    public ExerciseManager()
    {
        db = FirebaseFirestore.DefaultInstance;
    }

    public async Task<List<Exercise>> GetExercisesAsync()
    {
        var snapshot = await db.Collection("exercises").GetSnapshotAsync();
        Exercises.Clear();

        foreach (var doc in snapshot.Documents)
        {
            var exercise = doc.ConvertTo<Exercise>();
            Exercises.Add(exercise);
        }

        return Exercises;
    }

    public async Task AddOrUpdateExercise(Exercise exercise)
    {
        var docRef = db.Collection("exercises").Document(exercise.ExerciseId);
        await docRef.SetAsync(exercise);
    }

    public async Task DeleteExercise(string exerciseId)
```

```
    {
        var docRef = db.Collection("exercises").Document(exerciseId);
        await docRef.DeleteAsync();
    }
}
```

MuscleGroup.cs

```
using System;

[Serializable]
public class MuscleGroup
{
    public string GroupId;
    public string Name;
    public string Description;

    public MuscleGroup() { }

    public MuscleGroup(string id, string name, string desc)
    {
        GroupId = id;
        Name = name;
        Description = desc;
    }
}
```

MuscleGroupManager.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using Firebase.Firestore;
using UnityEngine;

public class MuscleGroupManager
{
    private FirebaseFirestore db;

    public List<MuscleGroup> MuscleGroups = new List<MuscleGroup>();

    public MuscleGroupManager()
    {
        db = FirebaseFirestore.DefaultInstance;
    }

    public async Task<List<MuscleGroup>> GetMuscleGroupsAsync()
    {
        var snapshot = await db.Collection("muscleGroups").GetSnapshotAsync();
        MuscleGroups.Clear();

        foreach (var doc in snapshot.Documents)
        {
            var group = doc.ConvertTo<MuscleGroup>();
            MuscleGroups.Add(group);
        }

        return MuscleGroups;
    }

    public async Task AddOrUpdateMuscleGroup(MuscleGroup group)
    {
        var docRef = db.Collection("muscleGroups").Document(group.GroupId);
        await docRef.SetAsync(group);
```

```
    }

    public async Task DeleteMuscleGroup(string groupId)
    {
        var docRef = db.Collection("muscleGroups").Document(groupId);
        await docRef.DeleteAsync();
    }
}
```

Settings.cs

```
using System;

[Serializable]
public class Settings
{
    public string UserId;
    public bool NotificationsEnabled;
    public string PreferredUnits; // "metric" or "imperial"

    public Settings() { }

    public Settings(string userId, bool notifications, string units)
    {
        UserId = userId;
        NotificationsEnabled = notifications;
        PreferredUnits = units;
    }
}
```

SettingsManager.cs

```
using System.Threading.Tasks;
using Firebase.Firestore;
using UnityEngine;

public class SettingsManager
{
    private FirebaseFirestore db;

    public Settings CurrentSettings { get; private set; }

    public SettingsManager()
    {
        db = FirebaseFirestore.DefaultInstance;
    }

    public async Task LoadSettings(string userId)
    {
        var docRef = db.Collection("settings").Document(userId);
        var snapshot = await docRef.GetSnapshotAsync();

        if (snapshot.Exists)
            CurrentSettings = snapshot.ConvertTo<Settings>();
        else
            CurrentSettings = new Settings(userId, true, "metric");
    }

    public async Task SaveSettings(Settings settings)
    {
        var docRef = db.Collection("settings").Document(settings.UserId);
```

```
        await docRef.SetAsync(settings);
        CurrentSettings = settings;
    }
}
```

FirebaseManager.cs

```
using System;
using System.Threading.Tasks;
using Firebase;
using Firebase.Auth;
using Firebase.Firestore;
using UnityEngine;

public class FirebaseManager : MonoBehaviour
{
    public static FirebaseManager Instance;

    public FirebaseAuth Auth;
    public FirebaseFirestore Firestore;

    private void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
            DontDestroyOnLoad(gameObject);
            InitializeFirebase();
        }
        else
        {
            Destroy(gameObject);
        }
    }

    private void InitializeFirebase()
    {
        FirebaseApp.CheckAndFixDependenciesAsync().ContinueWith(task =>
        {
            var dependencyStatus = task.Result;
            if (dependencyStatus == DependencyStatus.Available)
            {
                Auth = FirebaseAuth.DefaultInstance;
                Firestore = FirebaseFirestore.DefaultInstance;
                Debug.Log("Firebase initialized.");
            }
            else
            {
                Debug.LogError($"Could not resolve all Firebase dependencies:
{dependencyStatus}");
            }
        });
    }
}
```

ДОДАТОК В – ОПИС МОБІЛЬНОГО ЗАСТОСУНКУ ПІД ОС ANDROID ДЛЯ ДОМАШНІХ ТРЕНУВАНЬ «TRAINSPHERE»

1 Загальні відомості

Розроблений мобільний застосунок під ОС Android для домашніх тренувань «TrainSphere» представляє собою інструмент, який надає точну програму тренувань і допомагає користувачу досягти своїх цілей фізичної підготовки.

Для розробки програмного забезпечення були обрані наступні засоби:

– в якості мови програмування була обрана C#;

– в якості БД була обрана Firebase;

– в якості середовища розробки було обрано середовище Visual Studio та Unity. Також для розробки необхідна наявність встановленого .NET 6 SDK.

Для фукнціонування програмного забезпечення необхідна лише наявність пристрою з операційною системою Android версії 8.0 (Oreo) або новішої. Також необхідна наявність підключення до Інтернету для коректної роботи з базою даних Firebase.

2 Функціональне призначення

Програма виконує наступні основні функції:

− реєстрація та авторизація;

− перегляд інформації про тренування;

− виконання тренувань;

− перегляд інформації про інвентар;

− перегляд інформації про вправи;

− перегляд інформації про групи м’язів;

− робота з налаштуваннями;

− перегляд інформації про розклад тренувань;

3 Опис логічної структури

Мобільний застосунок створений за модульним принципом, де кожен компонент (модуль) виконує певну конкретну функцію, наприклад: формування тренувального плану (відповідає за формування тренувального плану), відображення інформації про вправи(відповідає за відображення інформації про вправи). Такий підхід забезпечує зручність у супроводі, масштабуванні та повторному використанні коду.

Крім того, програмне забезпечення побудоване за клієнт-серверною архітектурою, де клієнтська частина (мобільний застосунок) виконує взаємодію з серверною частиною, представленою базою даних Firebase. Це дозволяє зберігати дані користувача у хмарі, синхронізувати інформацію між пристроями та забезпечувати неперервну роботу додатку при наявності доступу до Інтернету.

4 Технічні засоби

Система повинна задовольняти вказаним вимогам на мобільному пристрої наступної мінімальної комплекції:

- Процесор: не нижче MediaTek MT6737 або Snapdragon 435;
- Оперативна пам’ять: 2ГБ;
- Дисковий простір: до 300 МБ;

5 Виклик та завантаження

Мобільний застосунок для домашніх тренувань «TrainSphere» встановлюється безпосередньо на пристрій користувача під керуванням операційної системи Android. Запуск застосунку здійснюється через відповідну іконку на головному екрані або в меню застосунків пристрою. Застосунок має один основний режим використання.

Після запуску «TrainSphere» відкривається вікно авторизації користувача, де потрібно ввести логін та пароль для входу в систему. У разі правильного введення даних відкривається головна сторінка застосунку, яка надає доступ до основних функцій: перегляду тренувань, складання розкладу та обліку результатів.

6 Вхідні та вихідні дані

Дані зберігаються в хмарній базі даних Firebase з підтримкою довільного доступу.

Вхідні дані:

Введення даних здійснюється за допомогою сенсорного екрану мобільного пристрою. Дані можуть вводитися вручну (наприклад, введення особистих даних або результатів тренувань) або обиратися із представлених списків (наприклад, вибір вправ чи планів тренувань).

Вхідними даними будуть:

- дані про користувача (Ім'я, Зріст, Вага, Стать, Ціль тренувань);
- інформація про вправи (Назва вправи, Опис вправи);
- інформація про тренування(Назва тренування, Опис тренування);
- інформація про розклад (Статус тренування, День);
- інформація про групи м'язів (Назва м'язів, Опис м’язів);
- інформація про інвентар (Назва інвентаря, Опис інвентаря);

Вихідні дані:

Виведення даних здійснюється через екран мобільного пристрою у вигляді текстової та графічної інформації (наприклад, відображення тренувальних планів, розкладів, статистики досягнень).

Вихідними даними будуть:

- дані користувача(Ім'я, Зріст, Вага, Стать, Ціль тренувань);
- дані вправ(Назва вправи, Опис вправи);
- списки тренувань(Назва тренування, Опис тренування);
- розклад(Статус тренування, День);
- дані про групи м'язів(Назва м'язів, Опис м’язів);
- дані про інвентар(Назва інвентаря, Опис інвентаря).

ДОДАТОК Г – КЕРІВНИЦТВО КОРИСТУВАЧА МОБІЛЬНОГО ЗАСТОСУНКУ ПІД ОС ANDROID ДЛЯ ДОМАШНІХ ТРЕНУВАНЬ «TRAINSPHERE»

1 Призначення програми

Розроблений мобільний застосунок під ОС Android для домашніх тренувань «TrainSphere» представляє собою інструмент, який надає точну програму тренувань і допомагає користувачу досягти своїх цілей фізичної підготовки.

1.1 Функціональне призначення

Функціональне призначення програми «TrainSphere» полягає в тому, щоб допомогти користувачам створити ефективний план тренувань, який відповідає їхнім цілям і рівню підготовки. Програма також надасть користувачам інформацію про різні типи вправ, їхню ефективність і техніку виконання.

1.2 Експлуатаційне призначення

Експлуатаційне призначення програми «TrainSphere» полягає в тому, щоб забезпечити користувачам зручний і простий у використанні інструмент для планування та виконання тренувань. Програма повинна бути доступною для користувачів з різними рівнями фізичної підготовки та досвіду.

1.3 Склад функцій

Для забезпечення повноцінної роботи користувача з системою передбачено низку основних функціональних можливостей, які охоплюють всі аспекти взаємодії з додатком. Зокрема, користувачі мають змогу:

- реєстрація та авторизація;
- перегляд інформації про тренування;
- виконання тренувань;
- перегляд інформації про інвентар;

- перегляд інформації про вправи;
- перегляд інформації про групи м’язів;
- робота з налаштуваннями;
- перегляд інформації про розклад тренувань;

2 Умови виконання програми

2.1 Мінімальний склад апаратних засобів

Система повинна задовольняти вказаним вимогам на мобільному пристрої наступної мінімальної комплекції:

- Процесор: не нижче MediaTek MT6737 або Snapdragon 435;
- Оперативна пам’ять: 2ГБ;
- Дисковий простір: до 300 МБ;

2.2 Мінімальний склад програмних засобів

Для забезпечення встановлення та стабільної роботи програми на мобільному пристрої необхідна наявність наступного мінімального програмного забезпечення:

Операційна система: Android 8.0 (Oreo) або новіша версія;

Менеджер додатків (Play Market): для завантаження та оновлення програми;

Інтернет-з’єднання: для авторизації, оновлення інформації та синхронізації даних з сервером;

Бібліотеки Android SDK: базові компоненти, необхідні для запуску програми, встановлюються автоматично разом з додатком.

2.3 Вимоги до користувачів

Програма розрахована на користувачів, які володіють базовими навичками користування мобільними пристроями.

Вимоги до користувача:

- вміння працювати з мобільними застосунками на базі Android;
- здатність зареєструватися в системі та виконувати вхід до облікового запису;

− розуміння базових принципів тренувального процесу (не обов’язково для всіх функцій);

− бажання стежити за фізичною активністю та користуватися додатком.

3 Виконання мобільного застосунку

Виконання програми

Запуск та вхід до мобільного застосунку «TrainSphere»

Для запуску мобільного застосунку «TrainSphere» необхідно натиснути на іконку застосунку на головному вікні мобільного пристрою або з списку застосунків мобільного пристрою. Іконка мобільного застосунку «TrainSphere» представлена на рисунку Г.1. Після цього відобразиться сторінка авторизації.

Рисунок Г.1 – Іконка мобільного застосунку «TrainSphere»

Після запуску мобільного застосунку користувач повинен пройти авторизацію для підтвердження права доступу до персональних тренувальних даних. На рисунку Г.2 представлено вікно авторизації. Для авторизації необхідно вказати електронну пошту та пароль, після чого натиснути кнопку «Увійти». Далі мобільний застосунок перенесе користувача до головного меню мобільного застосунку.

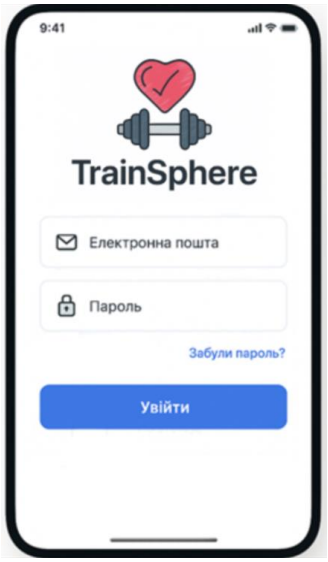


Рисунок Г.2 – Вікно авторизації мобільного застосунку «TrainSphere»

Головне меню мобільного застосунку «TrainSphere»

При успішній авторизації відкривається форма, яка містить головне меню мобільного застосунку «TrainSphere», яке представлене на рисунку Г.3.

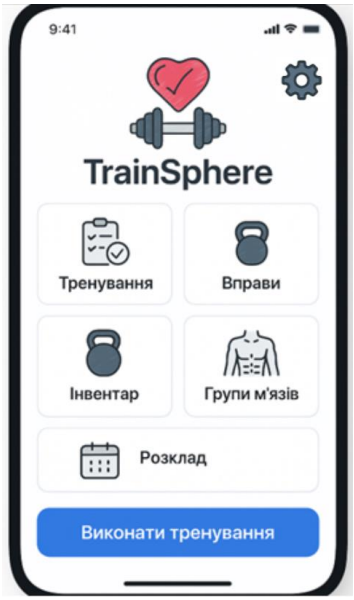


Рисунок Г.3 – Головне меню мобільного застосунку «TrainSphere»

Головна сторінка є центральною точкою взаємодії з мобільним застосунком і містить інтерфейс, що дозволяє зручно перемикатися між основними розділами. У мобільному застосунку реалізовані такі функціональні можливості, як перегляд видів тренувань (силові, кардіо, функціональні), робота з вправами, доступ до

інвентарю, створення та перегляд розкладу, запуск тренування в реальному часі, вивчення груп м’язів, а також налаштування профілю користувача. Залежно від обраного розділу, основна частина інтерфейсу динамічно змінюється, відображаючи відповідний контент – список тренувань, деталі вправ, інформацію про інвентар або таблицю запланованих занять.

Меню перегляду тренувань мобільного застосунку «TrainSphere»

Перебуваючи на головному меню можна переглянути тренування. Меню перегляду тренувань мобільного застосунку «TrainSphere», представлене на рисунку Г.4, воно дозволяє ознайомитися з різними видами тренувань, доступними в застосунку. Після переходу до цього розділу демонструється зручний список категорій тренувань, серед яких – силові, кардіо, функціональні та інші типи.

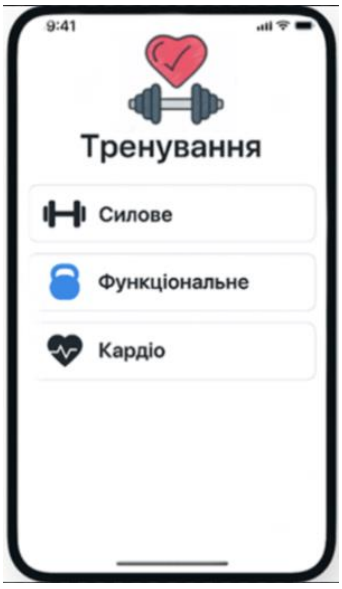


Рисунок Г.4 – Меню перегляду тренувань мобільного застосунку «TrainSphere»

Кожна категорія відображається як окремий елемент списку. Натиснувши на будь-яку з них мобільний застосунок переносить користувача на сторінку з детальною інформацією про обраний тип тренування. Тут можна побачити опис тренування, його основну мету, рівень складності, тривалість, а також перелік вправ, що входять до нього.

Меню перегляду вправ мобільного застосунку «TrainSphere»

Перебуваючи на головному меню можна переглянути вправи. Меню перегляду вправ мобільного застосунку «TrainSphere», представлене на рисунку Г.5, надає можливість ознайомитися з повним переліком доступних вправ, які можна використовувати під час тренувань. Це меню дозволяє легко знайти потрібну вправу, незалежно від рівня підготовки чи типу тренування.

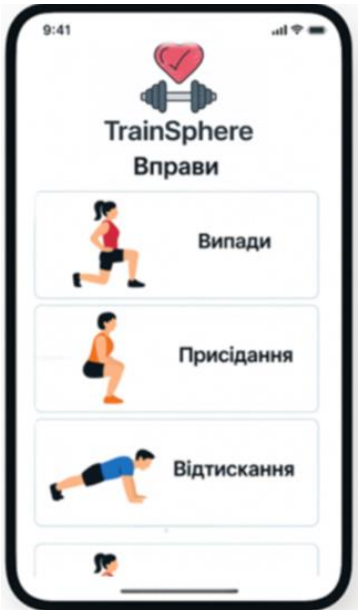


Рисунок Г.5 – Меню перегляду вправ мобільного застосунку «TrainSphere»

Після відкриття розділу «Вправи» демонструється список вправ, структурований у зручному вигляді. Кожна вправа представлена у вигляді окремого елемента списку. Натиснувши на будь-яку вправу, мобільний застосунок надає детальну інформацію: назву, опис виконання, а також відео виконання.

Меню перегляду інвентаря мобільного застосунку «TrainSphere»

Перебуваючи на головному меню можна переглянути інвентар. Меню перегляду інвентаря мобільного застосунку «TrainSphere» представлене на рисунку Г.6.

Рисунок Г.6 – Меню перегляду інвентаря мобільного застосунку «TrainSphere»

Меню дає змогу переглянути список доступного інвентарю, який можна використовувати під час тренувань. У цьому розділі продемонстровано перелік усіх доступних предметів, таких як гантелі, еспандери, килимки для йоги та інші аксесуари. Кожен елемент інвентаря відображається у вигляді окремої позиції зі стислим описом, що допомагає швидко зорієнтуватися. Натиснувши на будь-який інвентар, відображається детальна інформація.

Меню перегляду груп м’язів мобільного застосунку «TrainSphere»

Перебуваючи на головному меню можна переглянути групи м’язів. Меню перегляду груп м’язів мобільного застосунку «TrainSphere», представлене на рисунку Г.7, дозволяє ознайомитися з основними м’язовими групами, які задіюються під час різних видів тренувань. Цей розділ допоможе краще розуміти анатомію тіла та більш свідомо підходити до планування тренувального процесу.

Рисунок Г.7 – Меню перегляду груп м’язів мобільного застосунку «TrainSphere»

Після відкриття меню демонструється перелік основних груп м’язів, таких як ноги, руки, спина, прес тощо. Кожна група представлена окремою позицією списку. Вибравши конкретну групу, користувач переходить на сторінку з детальним описом.

Меню перегляду розкладу мобільного застосунку «TrainSphere»

Перебуваючи на головному меню можна переглянути розклад. Меню перегляду розкладу мобільного застосунку «TrainSphere», представлене на рисунку Г.8, дозволяє контролювати та керувати своїм тренувальним графіком у зручному форматі. Це один із ключових розділів застосунку, який дає змогу бачити як загальну структуру розкладу, так і деталі окремих тренувальних сесій.

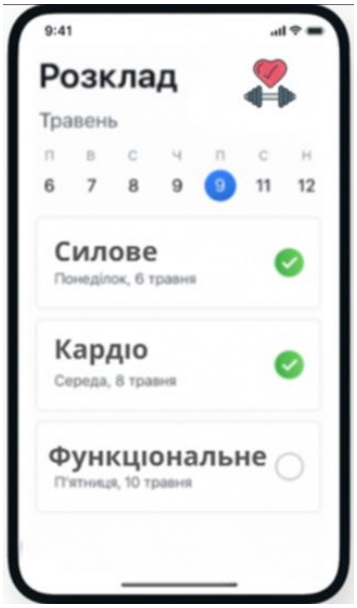


Рисунок Г.8 – Меню перегляду розкладу мобільного застосунку «TrainSphere»

Меню роботи з налаштуваннями мобільного застосунку «TrainSphere»

Перебуваючи на головному меню можна переглянути розклад. Меню роботи з налаштуваннями мобільного застосунку «TrainSphere», представлене на рисунку Г.9, дозволяє персоналізувати мовні налаштування та внести необхідні зміни до особистих даних.

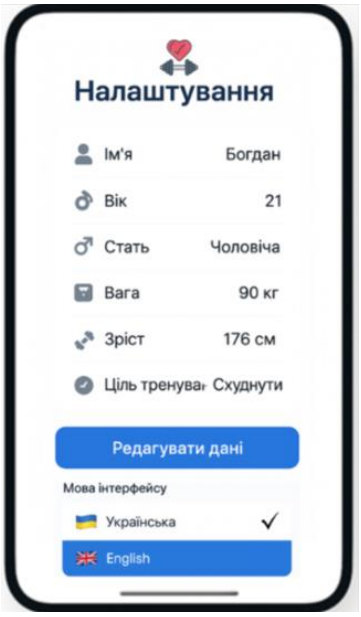


Рисунок Г.9 – Меню роботи з налаштуваннями мобільного застосунку «TrainSphere»

У цьому розділі можна редагувати такі дані про себе: ім’я, вік, зріст, вагу, стать, а також вказати свою ціль тренувань (наприклад, схуднення, набір м’язової маси або підтримка форми). Вказані параметри використовуються мобільним застосунком для надання більш точних рекомендацій і формування ефективніших тренувальних програм.

Крім того, у меню налаштувань передбачена можливість вибору мови інтерфейсу – української або англійської, що забезпечує додаткову зручність для користувача.

Меню виконання тренувань мобільного застосунку «TrainSphere»

Меню виконання тренувань мобільного застосунку «TrainSphere», яке представлене на рисунку Г.10, надає змогу швидко перейти до практичної частини тренувального процесу. У цьому розділі користувач може виконати тренування, якщо воно заплановане розкладом.

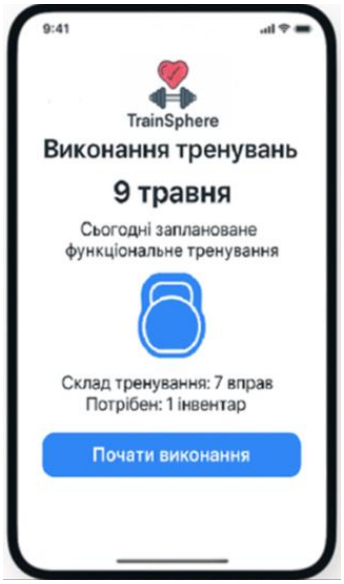


Рисунок Г.10 – Меню виконання тренувань мобільного застосунку «TrainSphere»

Після відкриття меню відображається поточна дата, тип тренування, яке заплановано (наприклад, силове, кардіо чи функціональне), загальну кількість вправ у цьому тренуванні, а також необхідний інвентар.

Для початку тренування необхідно натиснути кнопку «Почати виконання», після натискання якої запускається покроковий режим проходження тренування з детальними інструкціями та вказівками.

Завершення роботи програми

Щоб завершити роботу з мобільним застосунком «TrainSphere», достатньо скористатися стандартним способом закриття програм на мобільному пристрої. Можна натиснути кнопку «Назад» або змахнути застосунок із панелі відкритих додатків.

Усі внесені зміни до профілю, розкладу чи результатів тренувань автоматично зберігаються у хмарній базі даних Firebase, тому при наступному вході в систему ви зможете продовжити користування додатком без втрати даних.

ДОДАТОК Д – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ МОБІЛЬНОГО ЗАСТОСУНКУ ПІД ОС ANDROID ДЛЯ ДОМАШНІХ ТРЕНУВАНЬ «TRAINSPHERE»

1 Об’єкт випробувань

1.1 Найменування випробуваної програми

Повна назва об’єкта випробувань: мобільний застосунку під ОС Android для домашніх тренувань «TrainSphere».

1.2 Область застосування випробовуємої програми

Призначення програми «TrainSphere» полягає в тому, щоб забезпечити користувачам зручний і простий у використанні інструмент для планування та виконання тренувань.

Програма забезпечує:

- реєстрацію та авторизацію;
- перегляд інформації про тренування;
- виконання тренувань;
- перегляд інформації про інвентар;
- перегляд інформації про вправи;
- перегляд інформації про групи м’язів;
- роботу з налаштуваннями;
- перегляд інформації про розклад тренувань;

1.3 Позначення випробуваної програми

Мобільний застосунок «TrainSphere».

2 Мета випробувань

Метою проведення випробувань є необхідність перевірки відповідоності програмного забезпечення вимогам, які викладені у технічному завданні, що наводиться у додатку А.

3 Вимоги до мобільного застосунку

Мобільний застосунок має забезпечувати реалізацію функціональних можливостей, визначених у технічному завданні, що наведене в додатку А.

4 Вимоги до програмної документації

Для здійснення випробувань слід надати наступний пакет документації:

– технічне завдання на розробку мобільного застосунку;
– текст програми;
– опис програми;
– інструкцію користувача;
– програму і методику випробувань програмного забезпечення.

5 Засоби і порядок випробувань

5.1 Програмні засоби випробувань

До програмних засобів, необхідних для випробувань, належать:

− операційна система Android версії 8.0 (Oreo) або новішої;
− наявність активного підключення до Інтернету для обміну даними з хмарною базою даних Firebase;
− встановлене програмне середовище TrainSphere у вигляді APK-файлу.

5.2 Апаратні засоби випробувань

Мобільний застосунок має коректно функціонувати на пристроях з такою мінімальною апаратною конфігурацією:

− процесор: не нижче MediaTek MT6737 або Snapdragon 435;
− оперативна пам’ять: не менше 2 ГБ;

- дисковий простір: до 300 МБ вільної пам’яті.

5.3 Порядок проведення випробувань

Проведення випробування програмного забезпечення повинно відбуватися в наступному порядку:

- виконуються інструкції до кожної функції;
- проводиться аналіз отриманих результатів з метою визначення відповідності програмного продукту вимогам та системній документації.

Якщо в роботі системи виявляються помилки, складається відповідний звіт про їх перелік, після чого з розробником узгоджуються строки їх усунення. Після внесення виправлень замовник здійснює повторне тестування у повному обсязі, за потреби з використанням оновлених сценаріїв або додаткових тестів.

6 Методи випробувань мобільного застосунку «TrainSphere»

Випробування повинні проводитись в типових умовах використання, включаючи запуск застосунку, введення даних у форми, обробку подій і перевірку відображення повідомлень. Результати тестів фіксуються у вигляді тест-кейсів з урахуванням тестового оточення, залежностей та дій користувача.

6.1 Методика проведення перевірки вимог до програмної документації мобільного застосунку «TrainSphere»

Під час перевірки вимог до програмної документації необхідно здійснити перевірку наявності необхідних документів, їх відповідність вимогам до змісту та формату. На наступному етапі потрібно зробити аналіз документів з метою перевірки їх відповідності встановленим вимогам до змісту. Особливу увагу необхідно приділити наявності ключової інформації, опису функціональності програми, а також узгодженості тестової методики з вимогами до проведення випробувань. Також слід розглянути інструкцію користувача на предмет доступності викладення – чи забезпечує вона зрозуміле покрокове керівництво для ефективного використання мобільного застосунку. У процесі перевірки структури

та оформлення документації необхідно оцінити наявність необхідних елементів: заголовків, змісту, нумерації сторінок. Також слід проаналізувати якість текстового оформлення з точки зору читабельності, стилістики, а також відповідності граматичним та орфографічним правилам. Необхідно буде перевірити, що документи відповідають стандартам документації та характеризуються логічною й доступною структурою.

6.2 Методика проведення перевірки вимог до функціональних характеристик мобільного застосунку «TrainSphere»

Під час тестування мобільного застосунку «TrainSphere» будуть проведені наступні тести для перевірки вимог до функціональних характеристик:

- Перевірка успішної реєстрація з валідними даними.
- Перевірка помилки при введенні некоректного email.
- Перевірка помилки при введенні некоректного пароля.
- Перевірка валідації незаповнених полів.
- Перевірка успішної авторизація з правильними обліковими даними.
- Перевірка обробки порожніх полів.
- Перевірка функції «Забув пароль».
- Перевірка відображення даних користувача після авторизації.
- Перевірка перегляду списку тренувань.
- Перевірка відображення детальної інформації про тренування.
- Перевірка перегляду списку вправ.
- Перевірка відображення вправи із деталями.
- Перевірка перегляду доступного інвентаря.
- Перевірка відображення детальної інформації про інвентар.
- Перевірка перегляду м’язових груп.
- Перевірка відображення детальної інформації про м’язові групи.
- Перевірка перегляду розкладу тренувань.
- Перевірка початку виконання тренування.
- Перевірка переходу між вправами під час тренування.

- Перевірка завершення тренування з підсумком результатів.
- Перевірка збереження результатів виконаного тренування.

У таблиці Д.1 представлено перелік випробувань.

Таблиця Д.1 – Перелік випробувань основних функціональних можливостей

№	Дія	Очікуваний результат
1	2	3
1	Перевірка успішної реєстрація з валідними даними.	Користувач успішно зареєстрований та перенаправлений до головної сторінки
2	Перевірка помилки при введенні некоректного email	Виводиться повідомлення про помилку формату email
3	Перевірка помилки при введенні некоректного пароля	Виводиться повідомлення про недійсний пароль
4	Перевірка валідації незаповнених полів	Виводиться повідомлення про обов'язковість заповнення полів
5	Перевірка успішної авторизація з правильними обліковими даними	Користувач авторизований і перенаправлений до кабінету
6	Перевірка обробки порожніх полів при вході	Виводиться повідомлення про необхідність заповнення
7	Перевірка функції «Забув пароль»	Надсилається лист із посиланням на відновлення пароля
8	Перевірка відображення даних користувача після авторизації	Виводиться ім’я користувача, вік, зріст, вага, стать, ціль тренувань
9	Перевірка перегляду списку тренувань	Виводиться список доступних тренувань
10	Перевірка відображення детальної інформації про тренування	Відкривається сторінка з описом тренування
11	Перевірка перегляду списку вправ	Відображається перелік вправ
12	Перевірка відображення вправи із деталями	Відкривається інформація про вправу
13	Перевірка перегляду доступного інвентаря	Виводиться список інвентаря з назвами
14	Перевірка відображення детальної інформації про інвентар	Відображається опис інвентаря, фото, доступність
15	Перевірка перегляду м’язових груп	Виводиться перелік груп м’язів
16	Перевірка відображення детальної інформації про м’язові групи	Відображається опис м’язів, пов’язані вправи
17	Перевірка перегляду розкладу тренувань	Виводиться календар/таблиця з майбутніми тренуваннями
18	Перевірка початку виконання тренування	Починається режим тренування з таймером та вправами
19	Перевірка переходу між вправами під час тренування	Можна перемикатись між вправами, навігація працює

Кінець таблиці Д.1

1	2	3
20	Перевірка завершення тренування з підсумком результатів	Після завершення – виводиться звіт із результатами
21	Перевірка збереження результатів виконаного тренування	Результати зберігаються у профілі користувача

В результаті проведення випробувань мобільного застосунку під ОС Android для домашніх тренувань «TrainSphere» необхідно встановити відповідність функціональних характеристик розробленого мобільного застосунку та вимогам, які сформульовані в технічному завданні.