Національний університет «Полтавська політехніка імені Юрія Кондратюка»
(повне найменування вищого навчального закладу)
Навчально-науковий інститут інформаційних технологій та механотроніки
(повна назва факультету)
Кафедра комп'ютерних та інформаційних технологій і систем
(повна назва кафедри)

Пояснювальна записка
до дипломного проекту (роботи)
магістра
(освітньо-кваліфікаційний рівень)
на тему
Проектування мобільного фітнес-тренера з використаннямштучного інтелекту

Виконав: студент 2 курсу, групи 601-ТН
напряму
122 Комп'ютерні науки
(шифр і назва напряму)
Дацій Ю.А.
(прізвище та ініціали)
Керівник Лактіонов О.І.
(прізвище та ініціали)

Полтава – 2021 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА МЕХАНОТРОНІКИ

КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

напрям 122 «Комп'ютерні науки»

на тему

«ПРОЕКТУВАННЯ МОБІЛЬНОГО ФІТНЕС-ТРЕНЕРА З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ»

Студента групи 601-ТН Дація Юрія Андрійовича

Керівник роботи
кандидат технічних наук,
доцент Лактіонов О.І.

Завідувач кафедри
кандидат технічних наук,
доцент Головко Г.В.

Полтава – 2021

РЕФЕРАТ

Кваліфікаційна робота магістра: 99 с., 24рисунка, х додатки, 20 джерел.

Об’єкт дослідження:процеси розробки та функціонування мобільного додатку фітнес-тренеру.

Мета роботи: розробка мобільного додатку «Фітнес-тренер» що забезпечить створення ефективних тренувань у тренажерному залі та вдома. Додаток охоплює всі аспекти тренування, відпочинку та харчування.

Методи:проектування та розробкиефективної програми тренувань для додатку фітнес-тренера, створення макетів користувацького інтерфейсу, розробка програних модулів з застосуванням технології Agile та системи контролю версій GitHub.

Ключові слова:операційна система, мобільний додаток, мобільний телефон, додаток, користувач, Android, інформаційна система, модуль, база даних.

ABSTRACT

Masterthesis: 99pages, xxfigures, xappendices, 20sources.

Objectofstudy:processes of development and operation of a mobile application for a fitness trainer.

Objectiveofthework: development of a mobile application "Fitness Trainer" that will ensure the creation of effective workouts in the gym and at home. The application covers all aspects of training, recreation and nutrition.

Methods:design and development of an effective training program for the fitness trainer application, creation of user interface layouts, development of lost modules using Agile technology and GitHub version control system.

Keywords:operating system, mobile application, mobile phone, application, user, Android, information system, module, database.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

SQL – Structured Query Language.

ORM – Object-Relational Mapping.

ОС – Операційна система.

БД – База даних.

PROXY – Програмний комплекс, який виконує роль медіатора між користувачем та цільовим сервером.

API – Application Programming Interface

ПЗ – Програмне забезпечення

IDE – Інтегроване середовище розробки

НМ– Нейронна мережа

ADT–Android Development Tools

UI –User Interface

APK–Android Application Package

SDK –Software Development Kit

SVM–Support Vector Machine

ВСТУП

По мірі того, як люди розвивають свої можливості і бажання вести здоровий спосіб життя, виникає необхідність підтримувати ідею індивідуального підбору набору вправ і дієти, які дозволили б людині вести здоровий спосіб життя, відповідно до його можливостей і уподобань, витрачаючи при цьому мінімум зусиль і грошей. Для популяризації відстеження їх фізичних показників і підтримки інтересу до фізичної культури і спорту серед населення розробляються програмні продукти.

Одним з таких програмних продуктів є мобільні додатки. Мобільні пристрої сьогодні є значним інструментом в особистій розробці, допомагаючи не тільки використовувати стандартні функції дзвінків і повідомлень, але і виконувати завдання, без яких неможливо використовувати ці пристрої в повній мірі. Мобільні додатки є частиною цих завдань, вони дозволяють користувачеві відкрити для себе світ сучасних технологій, які знаходять своє застосування в цих додатках. Найдоступнішою і популярною операційною системою для мобільних пристроїв є Android. За даними Statcounter, частка цієї операційної системи на ринку мобільних операційних систем становить 73,52%, що говорить про те, що Android перевершує операційні системи, такі як iOS і Windows Phone

Актуальність теми полягає в тому, що мобільний додаток «Фітнес-тренер» – це готове рішення, що допоможе людині підготувати організм до загальних, різнопланових, всеосяжних фізичних навантажень, які використовуються не тільки в змагальній діяльності, але і в звичайному повсякденному житті. Це універсальна система фізичної підготовки, яка призначена для розвитку функціональних якостей користувачів будь-якої категорії, в тому числі і поліцейських, пожежних, військових.

У випускній кваліфікаційній роботі розглядаються мобільні додатки для заняття фізичною культурою та спортом, наводиться їх функціональна характеристика та порівняльний аналіз, пропонується нове рішення для

мобільних пристроїв – «Фітнес-тренер», яке на відміну від чотрьох аналогічних розробок враховує фізіологічну модель користувача, що дозволяє формувати індивідуальну траєкторію тренувань та розвитку фізичних показників користувача.

Метою роботи є оптимізація доступу користувачів до навчальних програм та програм харчування в додатку «Фітнес-тренер». Для досягнення цієї мети необхідно вирішити наступні завдання:

– провести аналіз вимог та предметної області виявлення, оглянути та проаналізувати існуючі рішення;
– здійснити проектування розділів мобільного додатку;
– здійснити розробку розділів харчування та тренувань;
– здійснити тестування мобільного додатку «Фітнес-тренер».

Очікувані результати: Готовий проект зможе зацікавити як професійних спортсменів, так і звичайних людей, у яких немає змоги займатися на спеціалізованих майданчиках та стадіонах.

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

Спорт став об’єктом державної уваги в XX столітті, а статус сфери державної політики отримав лише після II світової війни. Проте в сучасному світі не існує певних вироблених єдиних норм державної політики у цій сфері. Спорт для більшості країн залишається частиною суспільного життя, яка не є значущою в масштабах розвитку країни. У той же час в більшості розвинених країн створені механізми, що стимулюють розвиток суб’єктів спортивної галузі, засновані на податкові пільги, системі освіти, системі залучення муніципалітетів в розвиток і підтримку спортивної інфраструктури, підтримці громадських організацій, що займаються розвитком спорту, і т. п.

Дана тема не є достатньо вивченою Українськими економістами, оскільки сфера спорту в Україні в основному продовжує жити за законами і стандартам «радянського» спорту, на державному рівні не існує поняття «спортивна галузь» економіки, а роль недержавних організацій – національних спортивних федерацій, відповідальних з точки зору світових спортивних законів по розвиток видів спорту в країні, зведена до мінімуму.

Державна політика в сфері спорту в більшій мірі продовжує здійснюватися на основі принципів, закладених в радянські часи. Однак розвиток спорту в ринковому середовищі істотно відрізняється від його розвитку в плановій економіці. За радянських часів спорт (особливо дитячо-юнацький) не мав такого мізерного фінансування, процес його розвитку був значно більш централізований (відсутність повноважень місцевої влади на створення спортивних шкіл) і в більшій мірі спрямований на досягнення найвищих результатів (мета – відбір кращих вихованців для комплектування збірних команд, а не спорт як розвага, масовість занять спортом). На сьогоднішній день в Україні в сфері спорту (в першу чергу, дитячо-юнацького)

спостерігається та ж підміна понять, що і в сферах освіти і медицини: при декларованій безкоштовності – насправді фактична необхідність значних вкладень коштів (особливо батьківських) для досягнення результатів і спортивного росту. [1, с. 78-80]

Отже, зі склавшоїсяситуації ми можемо зробити наступний висновок. На даний час, спорт в незалежній Україні знаходиться в перманентному стані уповільненого руху вниз. Така динаміка не може продовжуватися надто довго. Саме тому з'явилася ідея створення додатку для мобільного телефону – «Персональний тренер». Інструментом можливого вирішення або хоча б поліпшення цього питання стане мобільний телефон.

На даний момент мобільні телефони грають важливу роль в сучасному світі. Практично у кожної людини є мобільний телефон, з його допомогою можна робити все: починаючи від читання звичайних кулінарних рецептів і закінчуючи укладанням угод у великих масштабах.

Усі функції в сучасних телефонах працюють за допомогою комплексу взаємопов'язаних програм, що називаються «операційною системою». На даний час існує багато різноманітних операційних систем. Але всі вони мають різноманітну популярність. Якщо додаток не багатоплатформовий, то розробка для вузького кола користувачів погано позначиться на його поширенні. Тому одним з головних завдань є вибір операційної системи, під яку буде розроблятися додаток. На рисунку 1 представлена діаграма, що показує долі на ринку, які займають різноманітні мобільні операційні системи. Отже, для вирішення поставленої задачі, закономірним рішенням буде обрати операційну систему Android.

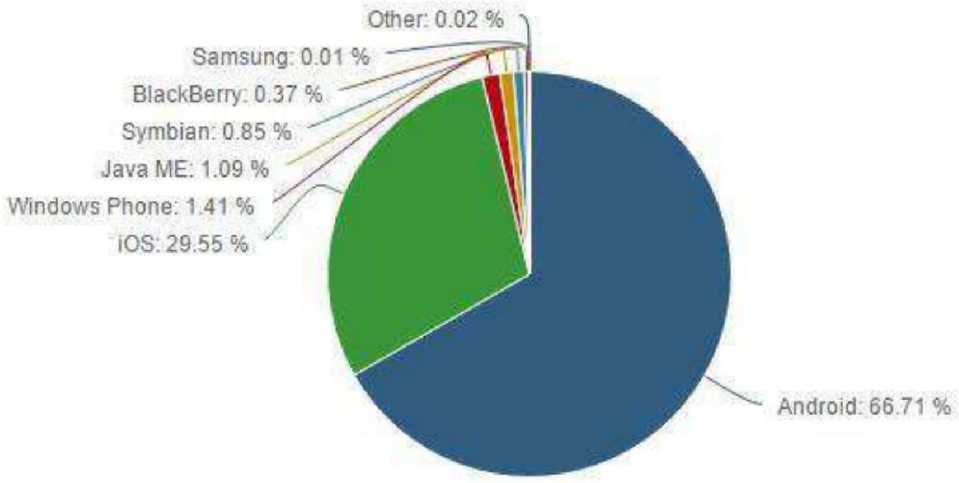


Рисунок 1.1 – Діаграма співвідношення мобільних операційних систем на ринку мобільних телефонів

Android – це операційна система для смартфонів, інтернет-планшетів, електронних книг, цифрових плеєрів, наручних годинників, ігрових консолей, нетбуків, смартбуків, окулярів Google, телевізорів та інших пристроїв. Він заснований на ядрі Linux і власній реалізації Google віртуальної машини Java.

Основні елементи Android:

– Applicationframework – забезпечує можливість гнучкої роботи з компонентами системи за допомогою API (ApplicationProgramInterface);

– Dalvikvirtualmachine – оптимізована для мобільних пристроїв віртуальна машина;

– Integratedbrowser – вбудований інтернет-браузер на основі відкритого рушія;

– Optimizedgraphics – підтримка користувацьких бібліотек 2D-графіки; OpenGL ES 1.0 використовується для реалізації 3D-графіки (апаратне прискорення необов'язкове).

– SQLite – використовується для зберігання структурованих даних.

– Mediasupport – підтримка аудіо, відео та форматів зображень (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).

– GSMTelephony, Bluetooth, EDGE, 3G, andWiFi, Camera, GPS, compass, andaccelerometer – (залежить від апаратного забезпечення конкретного мобільного пристрою).

– Richdevelopmentenvironment – включає в себе емулятор, інструменти для налагодження, профілювання пам'яті та продуктивності (Profiler), а також плагін дляEclipseIDE

Для створення власних додатків для Google Android потрібно встановити кілька компонентів – це Java Development Kit або JDK, який є пакетом інструментів розробника, а також Android Studio, яка містить інструменти SDK (Software Development Kit) – набір інструментів для розробки додатків для мобільної ОС. Це програмне забезпечення може бути використане як інструмент для отримання кореневих прав на пристрій і доступу до його системних компонентів.

Набір для розробки програмного забезпечення Android (SDK) містить повний набір інструментів розробки. До них відносяться налагоджувач, бібліотеки, емулятор на основі QEMU, документація, зразки коду та навчальні посібники. Покращення Android SDK йдуть «рука об руку» з загальним розвитком платформи Android. SDK також підтримує старі версії Android, якщо розробники хочуть зосередити свої програми на старих пристроях. Інструменти розробки є завантажуваними компонентами, тому після завантаження останньої версії та платформи для перевірки сумісності використовуються старі платформи та інструменти.

Існують API для взаємодії з веб-серверами та API Android.

API (ApplicationProgrammingInterface) – це програмний інтерфейс, інтерфейс для створення додатків або готовий код, що дозволяє отримувати інформацію з бази даних за допомогою запитів на спеціальний сервер. Синтаксис запитів і тип даних, які вони повертають, строго визначаються збоку самої служби.

Android API визначає функціональність, яку надає програма. Програмні компоненти спілкуються один з одним через API. При цьому компоненти

формують ієрархію – високорівневі компоненти використовують API низькорівневих компонентів, а вони, в свою чергу, використовують API ще компонентів нижчого рівня. Кожен шар використовує функціональність попереднього ("базового") шару передачі даних і, в свою чергу, забезпечує бажану функціональність наступному ("вищому") рівню.

Оскільки платформа Android і нові версії Android розвиваються, кожній версії Android присвоюється унікальний цілочисельний ідентифікатор, і називається рівень API. Тому кожній версії Android відповідає один рівень android API. Оскільки користувачі встановлюють більш ранні додатки, а також останню версію Android, реальні програми для Android повинні бути розроблені для роботи з декількома рівнями Android API.

Неможливо уявити сучасний телефон без мобільних додатків, адже саме вони роблять телефон багатофункціональним. Розглянемо поняття мобільного додатку.

Мобільний додаток (МД) – це спеціальний програмний продукт, який встановлюється користувачем на мобільний пристрій, як правило через спеціалізовані ринки (портали, магазини, маркетплейси). Найпопулярніші з них – App Store, Google Play, де на даний час за різним призначенням, мобільних застосунків вже налічується декілька мільярдів.

Будь-який мобільний додаток – це візитка, яка вручається постійним клієнтам, і він має вирішувати одне з трьох завдань користувача:

1. захоплююче проведення часу;
2. здійснення доступу в інтернет через брак інших способів;
3. можливість отримати потрібну інформацію в короткий проміжок часу.

Мобільні додатки поділяють по декількох категоріях, виходячи з того, для якої цільової аудиторії він розробляється, які цілі переслідує, як буде реалізований. Кожній категорії мобільних додатків властиві свої технічні характеристики і особливості реалізації. Нижче окреслено основні категорії мобільних додатків. Це лише найбільш поширені з них, насправді існує значно

більша кількість, а зовсім нові ідеї продовжують втілюватися розробниками з кожним днем.

Спорт. Спортивні новини, статистика, анонси, думки експертів, спілкування з фанатами і друзями. Електронні тренери. Купівля квитків. Покупки в режимі онлайн на будь-які заходи.

Розваги. До цієї категорії відносяться переважно ігри. Шутери, гонки, рішення головоломок і польоти на літаках; трансляція результатів на сторінку в соцмережі і брендування елементів гри.

Замовлення квитків. Простий і швидкий спосіб купівлі квитків в кіно, театр, на виставку. Є відгуки і оцінки і, відповідно, це підвищує продажі.

Бізнес. Додатки для фінансових організацій і банків: Включають цілий ряд професійних функцій: співвідношення валют, індекси, торговельні індекси та інше. Торгівля нерухомістю: Додатки містять карти з об’єктами продажу або оренди з докладною інформацією про кожного з них. Онлайн-продаж: Аукціони, розпродажі, колективні покупки абсолютно будь-яких предметів: від біжутерії до автомобілів.

Додатки для дітей. Все, що може зацікавити дитину: ігри, книги, мультфільми, музика, завдання та головоломки та інші розваги.

Подорожі. Бронювання готелю і не тільки. Оренда будинку або машини, замовлення номера в готелі та квитків на літак.

Туристичні гіди. Допоможуть знайти ресторан, магазин або заправку, розкажуть цікаві факти про пам’ятки і прокладуть зручний маршрут.

Додатки для міста. Допомагають зорієнтуватися в мегаполісі, знайти потрібний об’єкт, прокласти маршрут, припаркуватися і багато іншого.

Пошук роботи. Розмістити резюме, переглянути вакансії, відправити заявки і отримати повідомлення – зазвичай такі програми працюють в зв’язці з сайтом.

Соціальні програми. Соціальні мережі. Зручні для швидкого спілкування і обміну інформацією, перегляду новин і повідомлень. Існують

додатки для глобальних мереж, а також для вузьких і брендованих, наприклад, соцмережі BMW і Adidas.

Їжа. Замовлення і доставка їжі. Швидкі і зручні програми дозволяють замовляти їжу, ставити оцінки і залишати відгуки. За допомогою визначення геолокації закладу – додаток легко приведе вас до потрібного ресторану.

Рецепти. Додатки з покроковими фото і відеоінструкції страв, можливістю публікувати фото своїх кулінарних шедеврів, залишати коментарі та брати участь в конкурсах.

Ігри. Стати гравцем улюбленої футбольної команди, взяти участь в перегонах – все це доступно в ігрових додатках. Освіта. Навчання дітей. Вивчання будь-яких предметів і навичкам в ігровій формі.

Навчання навичкам. Правила дорожнього руху, управління яхтою, дресирування тварин або в'язання – можливості інтерактивних курсів нескінченні.

Новини. Газети, журнали та інші ЗМІ. Такі додатки зручні і значно розширюють аудиторію видань. Новини та коментарі можуть транслюватися в соцмережі або компілюватися в один RSS-потік. [2. с55-57]

Мобільні додатки захоплюють практично всі сфери побуту – від роботи до дозвілля і навіть особистого життя. Таким чином, мобільний телефон зі звичайного пристрою для зв'язку перетворився в інструмент з величезним потенціалом.

Основною метою дипломної роботи є розробка програмного модуля, що надасть можливість будь-якому користувачеві займатися спортом, незалежно від навколишніх умов, та допоможе змінити негативну динаміку розвитку спорту в Україні.

Розробка додатків для занять спортом поділяється на декілька етапів.

1. **Бриф.** Складання всіх побажань користувача з урахуванням специфіки обраного виду спорту.

2. **Визначення архітектури додатку.** Налаштування функцій, які забезпечать кінцевому користувачеві зручність при використанні.

3. **Вибір платформи.** Вибір найбільш відповідної по ряду найважливіших критеріїв платформи

4. **Розробка прототипу інтерфейсу.** Оформлення, дизайну і функціоналу в начерках.

5. **Написання програмного коду і процес доробок прототипу.**

6. **Тестування.**

7. **Публікація програми** на майданчиках, з яких кінцевий користувач зможе завантажити додаток.

1.2 Огляд існуючих програмних рішень

Популярність додатків для фітнесу та бігу останнім часом все зростає. Саме подібне рішення дозволяє максимально ефективно контролювати активність, вагу, спалювання калорій, внесення до тренувального процесу ігрового та змагального моменту.

Кожен додаток має свої характерні риси, власні переваги та недоліки. Тому, аби проектований нами мобільний додаток був якомога краще продуманий та реалізований, розглянемо існуючі аналоги:

- «Фитнес тренер FitProSport»;
- «Фитнес-план 30 дней»;
- «7 Минут Тренировка Pro».

Додатки буди обрані на основі рейтингу додатків, розміщених користувачами, кількості встановлень на мобільному пристрої від 10 000 і більше.

Критерії порівняння додатків базуються на потребах користувачів, які використовують мобільні додатки для занять спортом і фітнесом. Критерії, за якими оцінювалися додатки, наведені нижче:

- можливість індивідуального підбіру тренувань;
- наявність розділу харчування, відстеження сну;
- наявність персональних рекомендацій додатку;
- можливість підключення та синхронізації інших сервісів здоров'я.

"Фітнес тренер FitProSport" – це додаток для заняття спортом для чоловіків і жінок. Мобільний додаток має дві версії: платну та безкоштовну. Платна версія, на відміну від безкоштовної, не містить реклами, включає понад 400 готових планів тренувань для занять вдома та в тренажерному залі, графіки результатів тренувань та результатів вимірювання тіла.

У мобільному додатку **«Фитнес тренер FitProSport»** порівняно з «Фітнес-тренер» були виявлені такі особливості:

- відсутність підбору індивідуального плану тренувань;

– відсутність розділів харчування та контролю сну, рекомендацій додатку;

– немає можливості підключити додаткові послуги;

– відсутні програми тренувань для занять на вулиці.

Мобільний додаток **«Фитнес-план 30 дней»** розроблений для складання фітнес-плану на 30 днів. Воно також має дві версії: платну та безкоштовну.

У мобільному додатку «Фитнес-план 30 дней» порівняно з «Фітнес-тренер» були виявлені такі особливості:

– відсутність синхронізації із безкоштовною версією програми;

– відсутність заявленого функціоналу (календар, таймер відпочинку, харчування) у платній версії;

– обмеження 30 днів для складання програм тренувань.

Мобільний додаток **«7 Минут Тренировка Pro»** призначений для 7-хвилинної зарядки, заснованої на принципі «циклічних вправ високої інтенсивності». Тренування включає 12 вправ, кожна з яких виконується протягом 30 секунд, з перервами в 10 секунд між вправами.

У мобільному додатку «7 Минут Тренировка Pro» порівняно з «Фітнес-тренер» були виявлені такі особливості:

– додаток має тільки англійську моу;

– відсутність індивідуального підбору програм тренувань;

– відсутність розділів харчування та контролю сну;

– вузькоспеціалізована програма тренування.

Отже, огляд готових рішень показав, що існуючі програми мають ряд особливостей, відмінних від мобільного додатка «Фітнес-тренер», таких як: обмеження тривалості програми тренувань, відсутність розділу харчування та сну, відсутність підбору тренувань на основі способу життя користувачів. Також, не всі програми розроблені українською або російською мовою.

1.3 Постановка задачі

В результаті структурного аналізу системи були виявлені основні бізнес-процеси:

– додаток повинен мати реєстрацію користувачів;

– повинна здійснюватися відправка SMS з кодом підтвердження на вказаний номер телефону;

– повинна працювати перевірка номеру телефону та коду підтвердження.

– у додатку повинен створюватися особистий кабінет користувача з можливістю заповнення інформації;

– користувач повинен мати змогу додавати і обирати тренування;

– користувач повинен мати змогу додавати раціон харчування;

– користувач повинен мати змогу обирати мову інтерфейсу;

– повинна бути реалізована прив'язка додаткових сервісів.

Для реалізації всіх необхідних функцій необхідно спроектувати і розробити розділи мобільного додатку на базіопераційної системиAndroid.

Огляд готових рішень показав, що існуючі програми мають ряд особливостей, які відрізняються від мобільного додатку «Фітнес-тренер», такі-як: обмеження в тривалості програми тренувань, відсутність розділу харчування та сну, відсутність підбору тренувань на основі способу життя користувачів.

РОЗДІЛ 2
ПРОЕКТУВАННЯМОБІЛЬНОГО ДОДАТКУ ФІТНЕС-ТРЕНЕРУ

У розділі розглядаються основні аспекти розробки програмного забезпечення, а також реалізація розділів мобільного додатку.

Основні аспекти проектування включають наступні етапи:

– аналіз вимог;

– опис сценаріїв поведінки;

– розробка об'єктної моделі програмного забезпечення;

– розробка архітектури та моделювання взаємодії програмних компонентів.

Діаграми прецедентів, розглянуті в цьому розділі, побудовані згідно ISO/IEC 19505-2:2012 «Информационные технологии. Унифицированный язык моделирования группы по управлению объектами (OMG UML). Часть 2. Сверхструктура».

Також у розділі розглянуті прототипи діалогового взаємодії користувача і програмного продукту, демонструють реалізацію основних бізнес-процесів.

1.1 Аналіз вимог і опис структури та функціоналу системи

Аналіз вимог та опис функціонального призначення системи ґрунтується на використанні діаграм варіантів використання (Use Case) та діяльності (Activity Diagram). Діаграма є концептуальним уявленням або концептуальною моделлю системи в процесі її проектування та розробки. Суть даної діаграми полягає в наступному: проектована система представляється у формі варіантів використання, з якими взаємодіє актор. Актором або дійовою особою називається будь-який об'єкт, суб'єкт або система, що взаємодіє з моделюється системою ззовні. Варіант використання служить описи сервісів, які система надає актору.

На рисунку 2.1 представлена схема варіантів використання мобільним додатком фітнес-тренер.

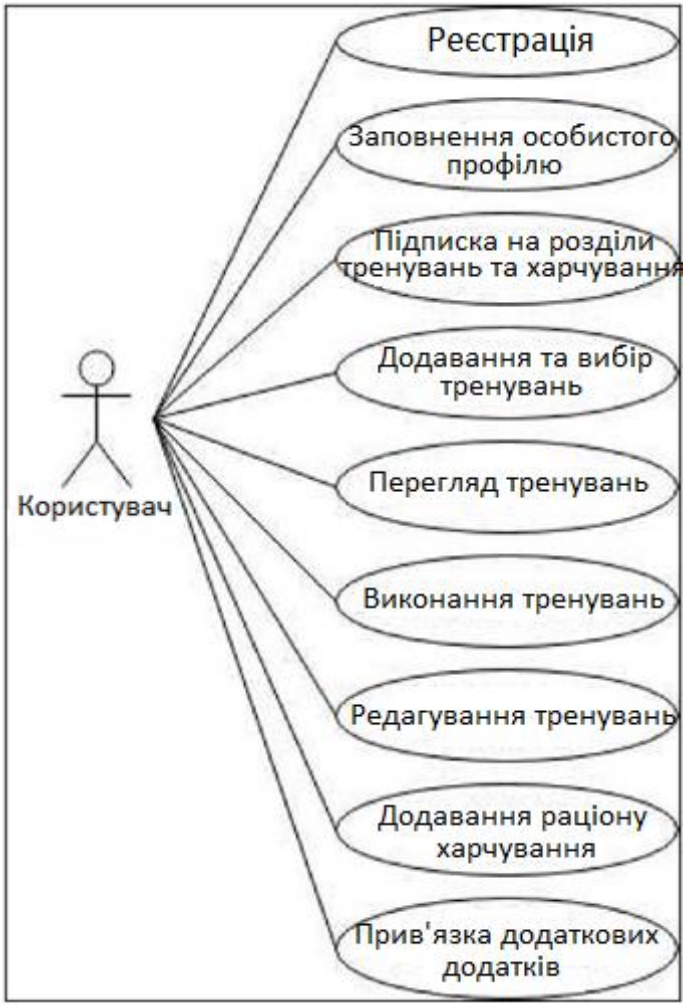


Рисунок 2.1 – Діаграма варіантів використання

Схема відображає функціональні вимоги мобільного додатку:

1) реєстрація користувача у мобільному додатку;
2) заповнення особистого профілю користувача;
3) підписка на розділи тренувань та харчування;
4) додавання та вибір тренувань;
5) перегляд тренувань;
6) виконання тренувань;
7) редагування тренувань;
8) додавання раціону харчування;
9) прив'язка додаткових додатків з фітнесу та спорту.

Розглянемо прецеденти діаграми окремо.

Прецедент 1: Реєстрація користувачів у мобільному додатку.

Короткий опис: Користувач реєструється у мобільному додатку.

Актор: Користувач.

Потік подій: Прецедент починається, коли користувач відкриває додаток вперше.

Базовий потік – Зареєструйтеся в мобільному додатку.

1) Користувач вводить логін (номер телефону) і пароль і натискає кнопку «Зареєструватися».

2) Система відправляє користувачеві SMS з кодом підтвердження.

3) Користувач вводить код підтвердження в додаток.

4) Система перевіряє введений користувачем код підтвердження з надісланим.

5) Якщо коди співрадають, користувач переходить на екран активності.

Альтернативний потік – введений код не відповідає коду, який було надіслано.

1) Користувач натискає на кнопку «Відправити код знову».

2) Перехід до пункту 2 основного потоку подій.

Постуслів'я: при успішному закінченні прецеденту користувач переходить на екран заповнення особистої інформації. При неуспішному – кнопка відправки коду стає неактивною протягом 30 с, потім протягом 60 с, потім протягом 120 с і потім стає не активною повністю.

Прецедент 2: Заповнення особистого профілю користувача.

Короткий опис: Користувач заповнює персональні дані у додатку.

Актор: Користувач.

Потік подій: прецедент починається, коли користувач заповнює особисті дані профілю.

Базовий потік – Введіть особисті дані у додаток.

1) Користувач вводить стать, вагу, зріст, дату народження, адресу електронної пошти.

2) Користувач вводить мету стеження за своєю фізичною активністю.

Передумови: Реєстрація у мобільному додатку.

Перед початком цього прецеденту користувач реєструється у додатку.

Післяуспішного завершення прецеденту користувач продовжує працювати з додатком і може зайти у будь-який розділ програми: «Активність», «Харчування», «Тренування», «Налаштування».

Прецедент 3: Додавання та вибір тренувань.

Короткий опис: Користувач проходить тестування, щоб отримати індивідуальний план тренувань. Або вибирає тренування із запропонованого списку і додає їх у свій план тренувань.

Актор: Користувач.

Потік подій: прецедент починається, коли користувач переходить у розділ "Тренування" і натискає на "Додати тренування"

Базовий потік – Пройти тестування та додати рекомендовану навчальну програму.

1) Користувач переходить в розділ «Тренування».
2) Система відображає тест для користувача.
3) Користувач проходить тест.
4) Користувач підписується на розділ «Тренування».
5) Система додає рекомендовану програму тренувань на основі результатів тестування.
6) Користувач вибирає рекомендовану навчальну програму.
7) Користувач вибирає час та дні виконання програми тренування.

Альтернативний потік – Додавання інших програм тренування.

1) Користувач переходить в розділ «Тренування»
2) Система перевіряє проходження тесту.
3) Якщо тест пройдено, користувач вибирає категорію програм тренувань.
4) Користувач вибирає програму тренування.
5) Користувач вибирає час і дні виконання обраної програми тренування.
6) Користувач додає обрану програму тренувань у план тренувань.

Передумови: Перед тим, як починається цей прецедент, користувач

зареєстрований у додатку.

При успішному закінченні прецеденту користувач продовжує роботу з додатком і може виконувати, переглядати і редагувати план тренувань.

На рисунку 2.2зображено діаграму діяльності прецеденту «Додавання та вибір тренувань».

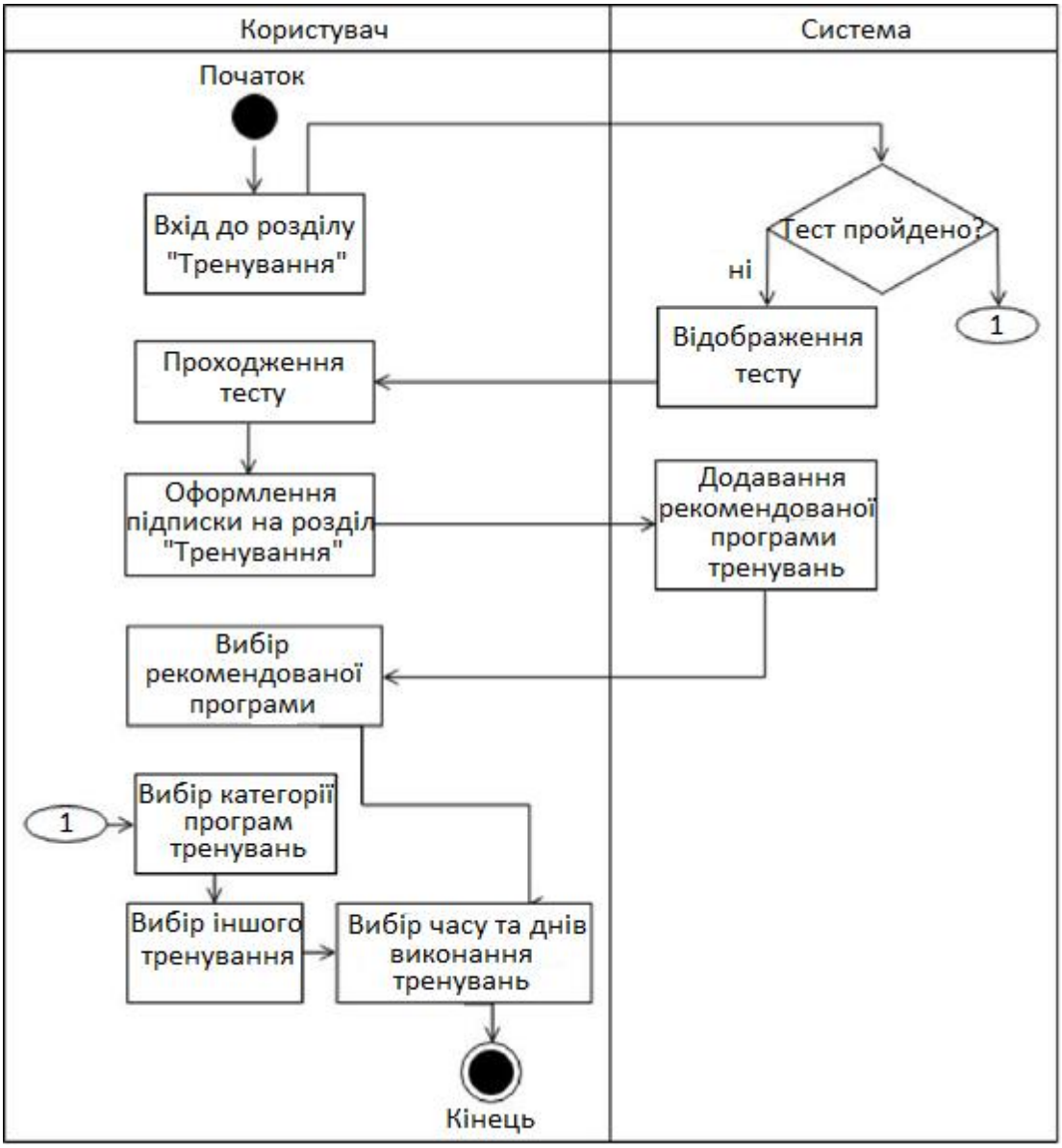


Рисунок 2.2 – Прецеден «Додавання та вибір тренувань»

Прецедент 4: Перегляд та редагування тренувань.

Короткий опис: Користувач переглядає, редагує рекомендовані або додані програми тренувань.

Актор: Користувач.

Потік подій: Прецедент починається, коли користувач заходить у розділ

«Тренування» та вибирає будь-яке з рекомендованих або доданих тренувань.

Базовий потік – Перегляд програми тренувань.

1) Система відображає усі прогрми тренувань.
2) Користувач переглядає прогрми тренувань по днях.

Альтернативний потік – Редагування прогрми тренувань.

1) Система відображає усі прогрми тренувань.
2) Користувач вносить зміни в тренування (дата початку, дні,виконання, скасування тренування)
3) Система вносить зміни у прогрму тренувань.
4) Система відображає оновлену прогрму тренувань.

Передумови: Перед тим, як починається цей прецедент, користувач зареєстрований у додатку, у користувача оформлена підписка на розділ «Тренування», у користувача є додане рекомендоване тренування.

При успішному закінченні прецеденту користувач продовжує роботу з додатком.

Прецедент 5: Виконання тренувань.

Короткий опис: Користувач виконує рекомендовані або додані програми тренувань.

Актор: Користувач.

Потік подій: Прецедент починається, коли користувач заходить у розділ «Тренування» та вибирає будь-яке з рекомендованих або доданих тренувань.

Базовий потік – Виконати програму тренувань.

1) Користувач заходить до розділу «Тренування».
2) Система відображає всі додані тренувальні програми.
3) Користувач вибирає програму тренувань.
4) Система відображає вправи, що включені в дане тренування.
5) Користувач включає відтворення звуку у тренуванні.
6) Користувач починає тренування за допомогою відтворення відео.
7) Користувач натискає кнопку "Завершити тренування".
8) Система робить відмітки про витрачений час на тренування та інші

дані, необхідні для складання звіту.

9) Система складає звіт про виконане тренування.

Передумови: Перед тим, як починається цей прецедент, користувач зареєстрований у додатку, у користувача оформлена підписка на розділ «Тренування», у користувача є додане рекомендоване тренування.

При успішному закінченні прецеденту користувач продовжує роботу з додатком і може переглянути звіт з виконаних тренувань.

На рисунку 2.3 зображено діаграму діяльності прецеденту «Виконання тренувань».

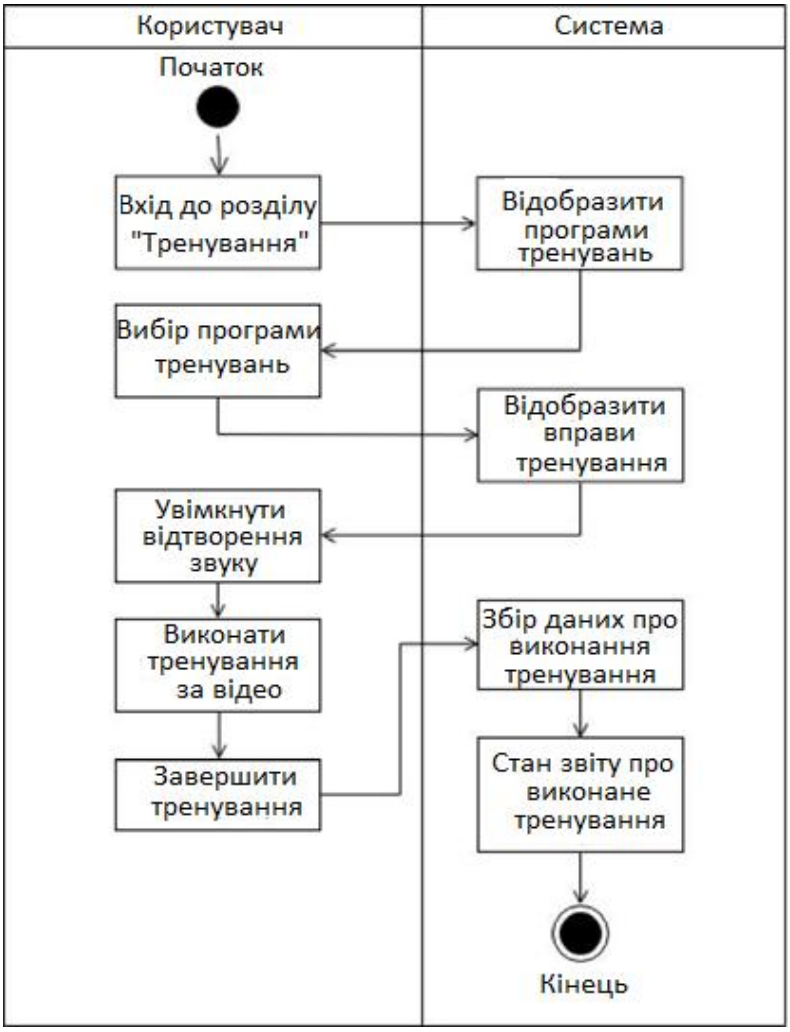


Рисунок 2.3 – Прецеден «Виконання тренувань»

Прецедент 6: Додавання раціону харчування.

Короткий опис: Користувач проходить тестування, щоб отримати індивідуальний план харчування або обирає раціон харчування із запропонованого списку і додає їх у свій план харчування.

Актор: Користувач.

Потік подій: Прецедент починається, коли користувач переходить до розділу "Харчування" і натискає "Додати раціон харчування ".

Базовий потік – Пройти тестування та додати рекомендовану програму тренувань.

1) Користувач переходить до розділу «Харчування».
2) Система відображає тест користувачеві.
3) Користувач проходить тест.
4) Користувач підписується на розділ «Харчування».
5) Система додає рекомендовану програму харчування на основірезультатів тестування.

Альтернативний потік – Додавання інших програм живлення.

1) Користувач заходить до розділу «Харчування»
2) Система перевіряє, проходження тесту.
3) Якщо тест пройдено, користувач вибирає інші програми харчування.
4) Користувач додає обрану програму харчування до свого плану харчування.
5) Користувач переглядає план харчування тарекомендації щодо здоров'я.

Передумови: Першніжцей прецедент починається, користувач реєструється в додатку, користувач підписався на розділ "Харчування".

При успішному закінченні прецеденту користувач продовжує роботу з додатком і може переглядати та редагувати план харчування.

Прецедент 7: Прив'язка додаткових додатків.

Короткий опис: Користувач має можливість прив'язати до застосування інші сервіси здоров'я та фітнесу.

Актор: Користувач.

Потік подій: Прецедент починається, коли користувач заходить у розділ «Параметри» та натискає «Прив'язати інші сервіси».

Базовий потік – Прив'язати інші сервіси.

1) Користувач переходить в "Налаштування".

2) Користувач вибирає "Прив'язати інші сервіси".

3) Система відображає доступні сервіси для прив'язки.

4) Користувач обирає необхіднйи сервіс.

5) Система відображає екран з авторизацією у вибраному сервісі.

6) Користувач проходить авторизацію.

7) Система синхронізує дані обраного сервісу з додатком.

Передумови: Перед тим, як починається цей прецедент, користувач зареєстрований у додатку.

При успішному закінченні прецеденту користувач продовжує роботу з додатком і може переглядати звіт про виконання програм тренувань та харчування з синхронізованою інформацією з інших сервісів.

1.2 Об'єктна модель програмного забезпечення

Діаграма класів призначена для відображення класів програми, що розробляється, і з взаємозв'язків. Діаграма класів дозволяє продемонструвати класи системи, їх атрибути, методи та взаємодію між класами.

На рисунку2.4 представлена діаграма класів прецеденту виконання тренування.

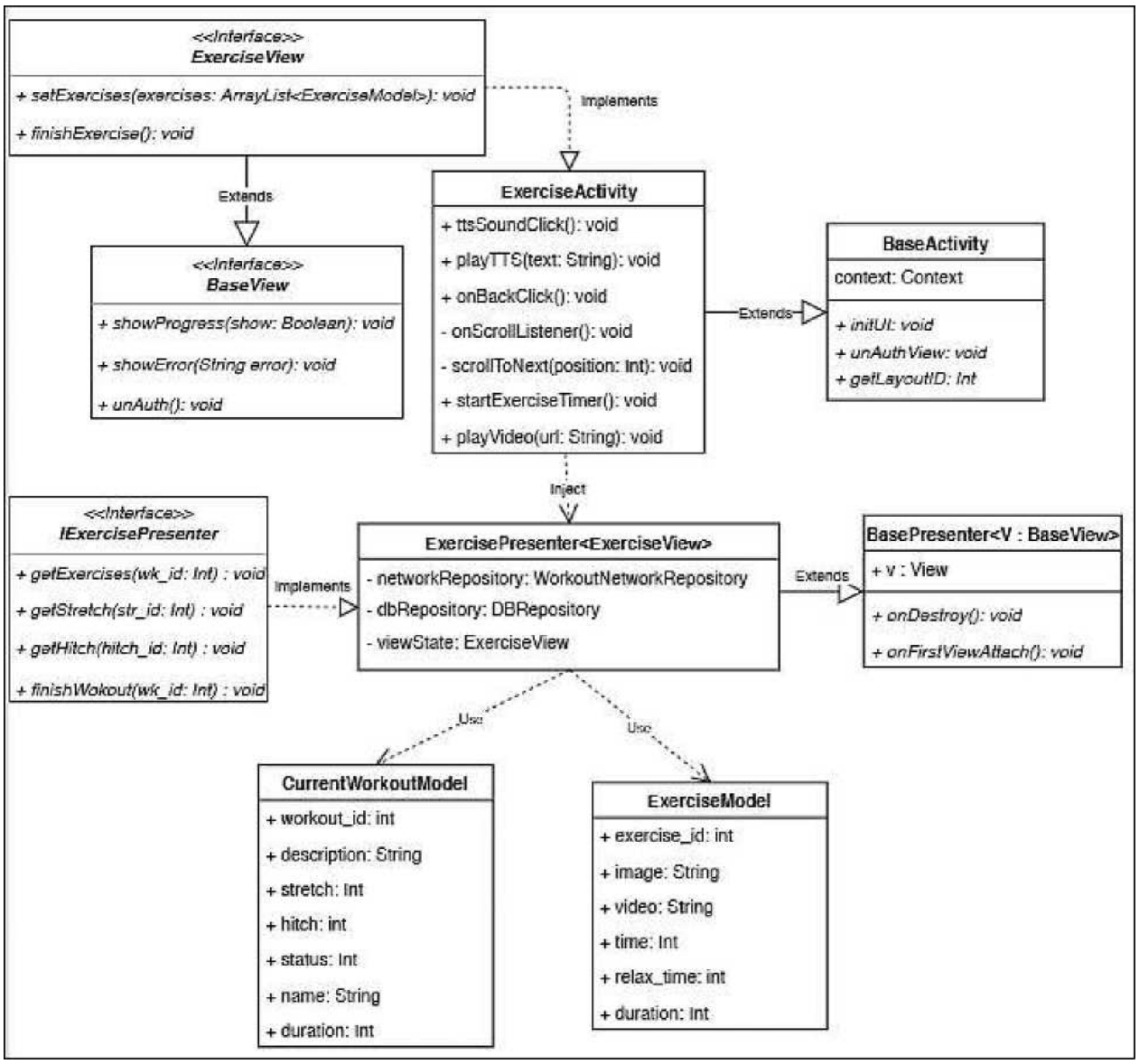


Рисунок 2.4 – Об'єктна модель прецеденту виконання тренування

Діаграма класів спроектована за допомогою патерну проектування MVP. Основна ідея будь-якого патерну проектування – це поділ логіки та UI-частини так, щоб їх можна було налагоджувати окремо [17]. На діаграмі класів представлено 3 шари:

Model – шар, який відображає дані з бази даних.

View – шар, який відображає інформацію на екрані та обробляє натискання на екрані.

Presenter – шар, який відповідає за логіку.

Шар Model представлений двома класами: CurrentWorkoutModel і Exer¬ciseModel. Перший відповідає за обране тренування, а другий за певні вправи у цьому тренуванні.

Клас CurrentWorkoutModel має такі атрибути: workout_id – ідентифікатор тренування, description – опис тренування, stretch – тривалості зарядки, hitch – тривалості затримки, status – статус виконання тренування, duration – тривалість тренування. Атрибути класу CurrentWorkoutModel є публічними (public).

Клас ExerciseModel має такі атрибути: exercise_id – ідентифікатор вправи, image – посилання на зображення вправи, video – посилання на відео виконання вправи, time – передбачений час виконання вправи, relax_time – час відпочинку між вправами, du¬ration – тривалість виконання вправи. Атрибути класу ExerciseModel є публічними (public).

Шар Presenter представлений трьома класами: IExercizePresenter, ExerciseP¬resenter, BasePresenter.

Клас IExercizePresenter – інтерфейс, який містить абстрактні операції: getExercise() – отримати вправу, getStretch() – отримати зарядку, getHitch() – отримати затримку, finishWorkout() – завершити тренування. Операції класу IExercizePresenter є публічними (public).

Клас ExercisePresenter реалізує інтерфейс IExercizePresenter і використовує класи моделей CurrentWorkoutModel та ExerciseModel. Успадковується від класу BasePresenter. Містить атрибути: networkRepository – екземпляр класу WorkoutNetworkRepository для роботи з мережею, dbRepository – екземпляр класу DBRepository для роботи з базою даних, viewState – екземпляр класу IExerciseView для виклику методів. Атрибути класу ExercisePresenter є приватними (private).

Клас BasePresenter має атрибут v – зберігає посилання на ExerciseView, операції: onDestroy() – знищити екран, onFirstViewAttach() – клас Eser-ciseActivity підключає клас ExercisePresenter. Атрибут та операції класу BasePresenter є публічними (public).

Шар View представлений чотирма класами: IExerciseView, BaseView, ExerciseActivity, BaseActivity.

Клас ExerciseView – інтерфейс, операціями якого є: setExercise() –

розпочати виконання вправ, finishExercise() – закінчити вправу. Клас IExerciseView успадковується від класу BaseView. Операції класу IExerciseView є абстрактними та публічними (public).

Клас BaseView – інтерфейс, операціями якого є: showProgress() – завантаження сторінки, showError() – показати помилку, un- Auth() – перехід на екран авторизації. Операції класу BaseView є абстрактними та публічними (public).

Клас ExerciseActivity – реалізує інтерфейс IExerciseView, успадковується від класу BaseActivity. Операціями класу ExerciseActivity є: ttsSoundClick() – натискання на озвучування вправ, playTTS() – озвучування виконання вправ, onBackClick() – повернутися на попередню вправу, onScrollListener() – перехід на наступну вправу, scroll- розминки, startExerciseTimer() – запуск таймера тренування, playVideo() – відтворення відео вправи. Операції класу ExerciseActivity є публічними (public).

Клас BaseActivity має атрибут context – доступ до операційної системи Android, абстрактні операції: initUI – ініціалізація елементів екрана, unAuthView – показати екран авторизації, getLayoutID – отримати ідентифікатор екрану з тренуванням.

Діаграма послідовності використовується для опису повного контексту взаємодій як своєрідного тимчасового графіка «життя» всієї сукупності об'єктів, що взаємодіють між собою для реалізації варіанта використання програмної системи, досягнення бізнес-мети або виконання будь-якої задачі, наприклад, для представлення динаміки поведінки об'єктів, відображаючи передачу повідомлень між відповідними класами [7,8].

На рисунку 2.5 зображено моделювання взаємодії користувача з програмним продуктом, що демонструє реалізацію основних бізнес-процесів.

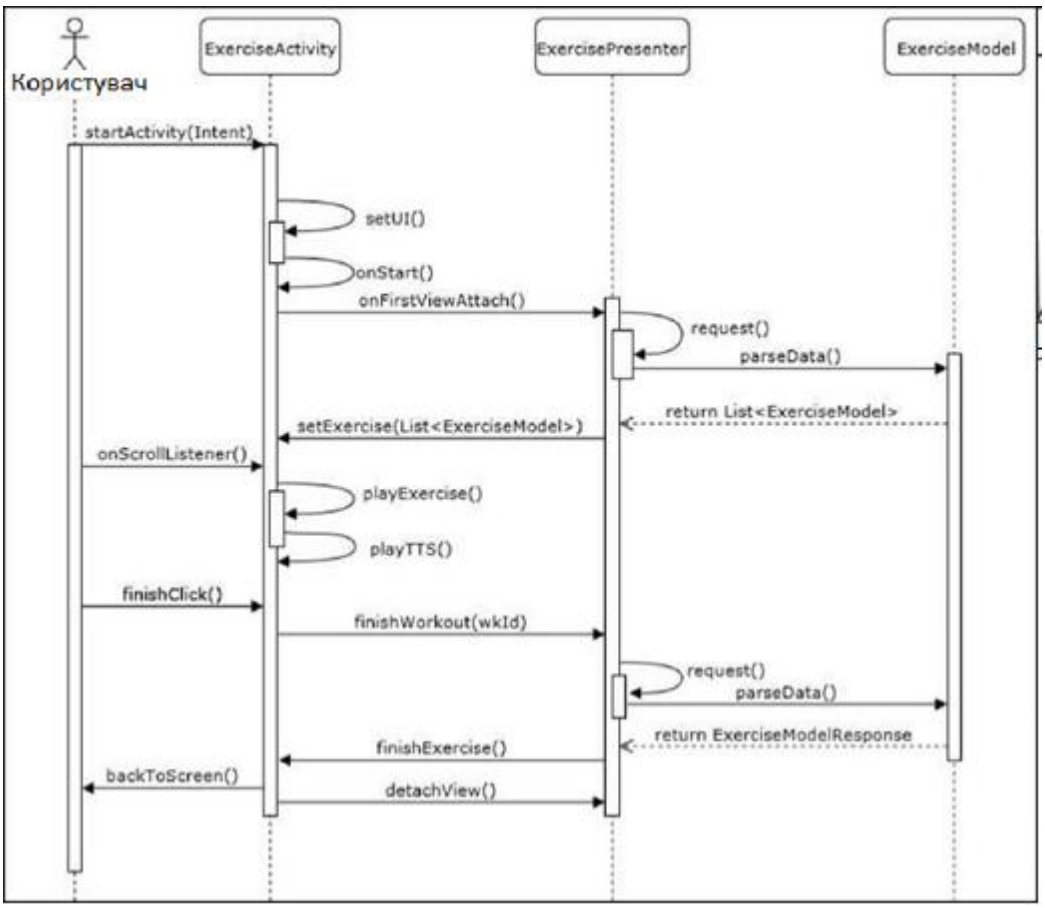


Рисунок 2.5 – Моделювання взаємодії користувача з програмним продуктом

Прецедент починається, коли користувач починає тренування.

Об'єкт «Користувач» переходить на екран виконання вибраного тренування та надсилає об'єкту «ExerciseActivity» повідомлення «startActivity», яке запускає екран виконання тренування.

Об'єкт «ExerciseActivity» ініціалізує елементи екрана (setUI) і екран показується користувачеві (onStart).

Об'єкт ExerciseActivity посилає об'єкту ExercisePresenter повідомлення onFirstViewAttach і підключає його (ExerciseActivity знає про існування ExerciseActivity).

Об'єкт «ExercisePresenter» запитує список вправ у тренування («reste») і отримані дані перетворюються на об'єкт «ExerciseModel» («parseData») і повертаються в «ExercisePresenter» («return List»).

Об'єкт ExercisePresenter посилає об'єкту ExerciseActivity повідомлення setExercise і передає в ExerciseActivity список вправ.

Об'єкт «Користувач» прокручує вправи тренування та надсилає об'єкту

«ExerciseActivity» повідомлення «onScrollListener».

Об'єкт ExerciseActivity починає програвати відео з вправами (playExercise) і відтворювати озвучування вправ (playTTS).

Об'єкт «Користувач» натискає на завершення тренування та надсилає об'єкту «ExerciseActivity» повідомлення «finishClick».

Об'єкт ExerciseActivity посилає об'єкту ExercisePresenter повідомлення finishWorkout про те, що тренування завершено і передає в ExercisePresenter ідентифікатор (wkid) тренування.

Об'єкт «ExercisePresenter» виконує запит на завершення тренування («reste») і отримані дані перетворюються на об'єкт «ExerciseModel» («parseData») і повертаються в «ExercisePresenter» («return ExerciseModelResponse»).

Об'єкт ExercisePresenter посилає об'єкту ExerciseActivity повідомлення finishExercise про те, що вправи успішно завершені.

Об'єкт ExerciseActivity повідомляє об'єкту ExercisePresenter про те, що завершив свою роботу (detachView)

Об'єкт "ExerciseActivity" надсилає об'єкту "Користувач" ExerciseActivity "backToScreen" і об'єкт "Користувач" повертається на екран вибору тренувань.

1.3 Моделювання взаємодії програмних компонентів

У мові моделювання UML для фізичного представлення моделей систем використовуються діаграми реалізації, які включають дві діаграми: компонентів і розгортання.

Діаграма компонентів дозволяє визначити архітектуру системи, що розробляється. Пунктирні стрілки, що з'єднують модулі, демонструють відносини взаємопов'язаності. Основними графічними елементами діаграми компонентів є компоненти, інтерфейси та залежності між ними [8].

Для реалізації проекту використовуватиметься патерн Dependency Injection, що дозволяє спростити процес надання залежностей програмному

компоненту, також спрощує тестування окремих елементів програми зручної реалізації заміни залежностей.

На рисунку 2.6 представлено моделювання компонентів розроблюваних розділів додатку «Фітнес-тренер».

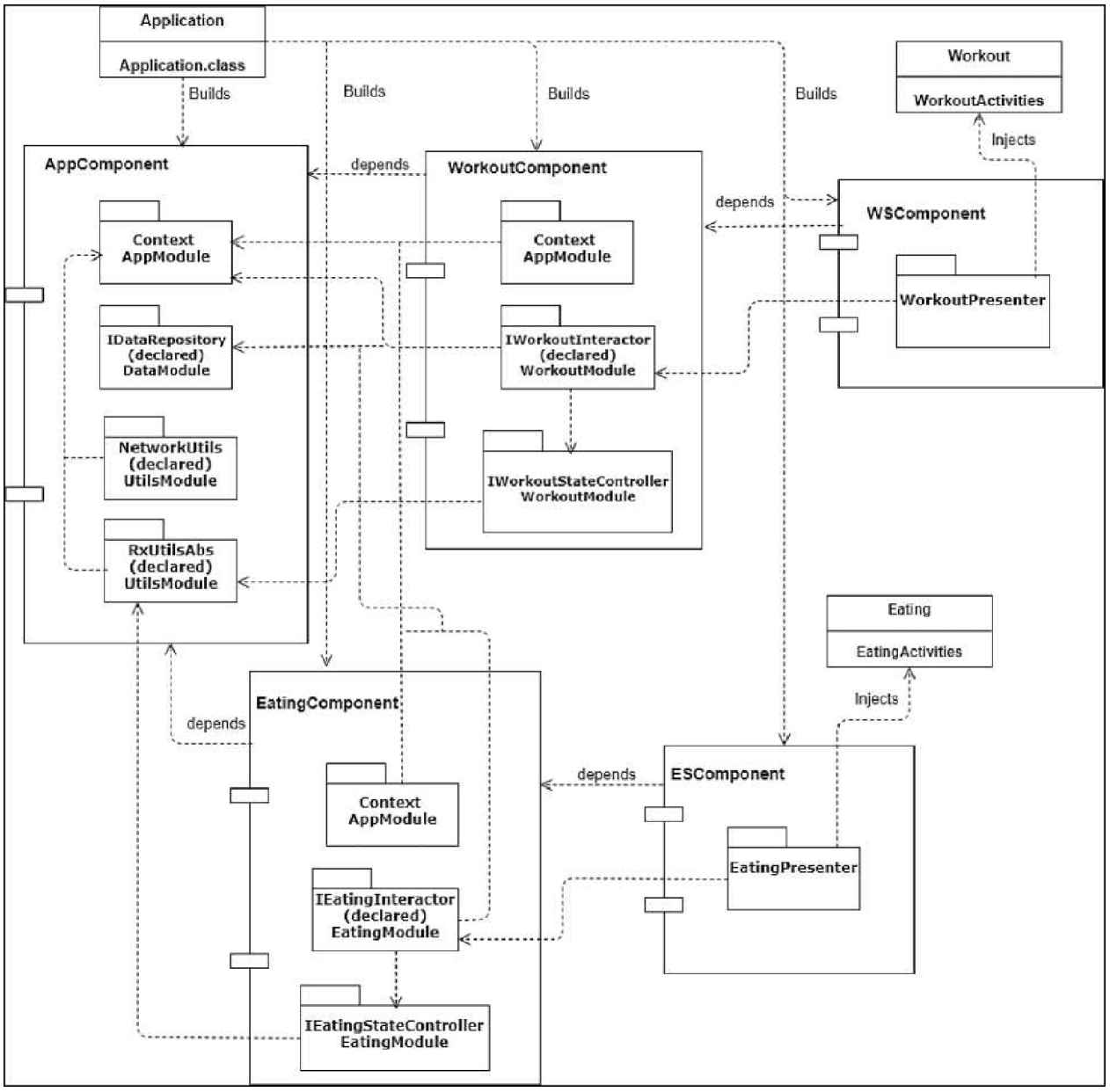


Рисунок 2.6 – Моделювання компонентів розроблюваних розділів додатку «Фітнес-тренер».

На рисунку 16 представлено 5 компонентів:

1) AppComponent – це компонент, який є точкою входу, через яку система або користувач може увійти в програму. Тут зберігаються об'єкти, необхідні для життєвого циклу програми, тобто глобальні синглтони: Context (контекст програми), IDataRepository (клас, що відповідає за запити до сервера),

NetworkUtils (клас-утиліта для роботи з мережею) та RXUtilsAbs (клас - Утиліта).

2) WorkoutComponent – компонент, що містить об'єкти, які потрібні для всіх екранів Тренувань: IWorkoutInteractorта IWorkoutStateController. Надає локальні синглтони для всіх екранів тренувань.

3) WSComponent – компонент, що надає "локальні сингл¬тони" для конкретного екрана Тренувань. Кожен екран має свій Presenter, стійкий до переорієнтації, тобто чий життєвий цикл відрізнятиметься від життєвого циклу фрагмента або активіті.

4) EatingComponent – компонент, що містить об'єкти, які потрібні для всіх екранів Живлення: lEatingInteractor і IEatingStateController. Надає локальні синглтони для всіх екранів Харчування.

5) ESComponent – компонент, що надає "локальні сингл-тони" для конкретного екрана Живлення. Кожен екран має свій Presenter, стійкий до переорієнтації, тобто чий життєвий цикл відрізнятиметься від життєвого циклу фрагмента або активіті.

Другий формою фізичного уявлення програмної системи є діаграма розгортання. Вона використовується для представлення загальної конфігурації та топології розподіленої програмної системи та містить зображення розміщення компонентів за окремими вузлами системи. Діаграма розгортання містить графічне зображення процесорів пристроїв, процесів та зв'язків між ними [8].

Нарисунку 2.7 зображена архітектура розділів, що розробляються додатки «Фітнес-тренер».

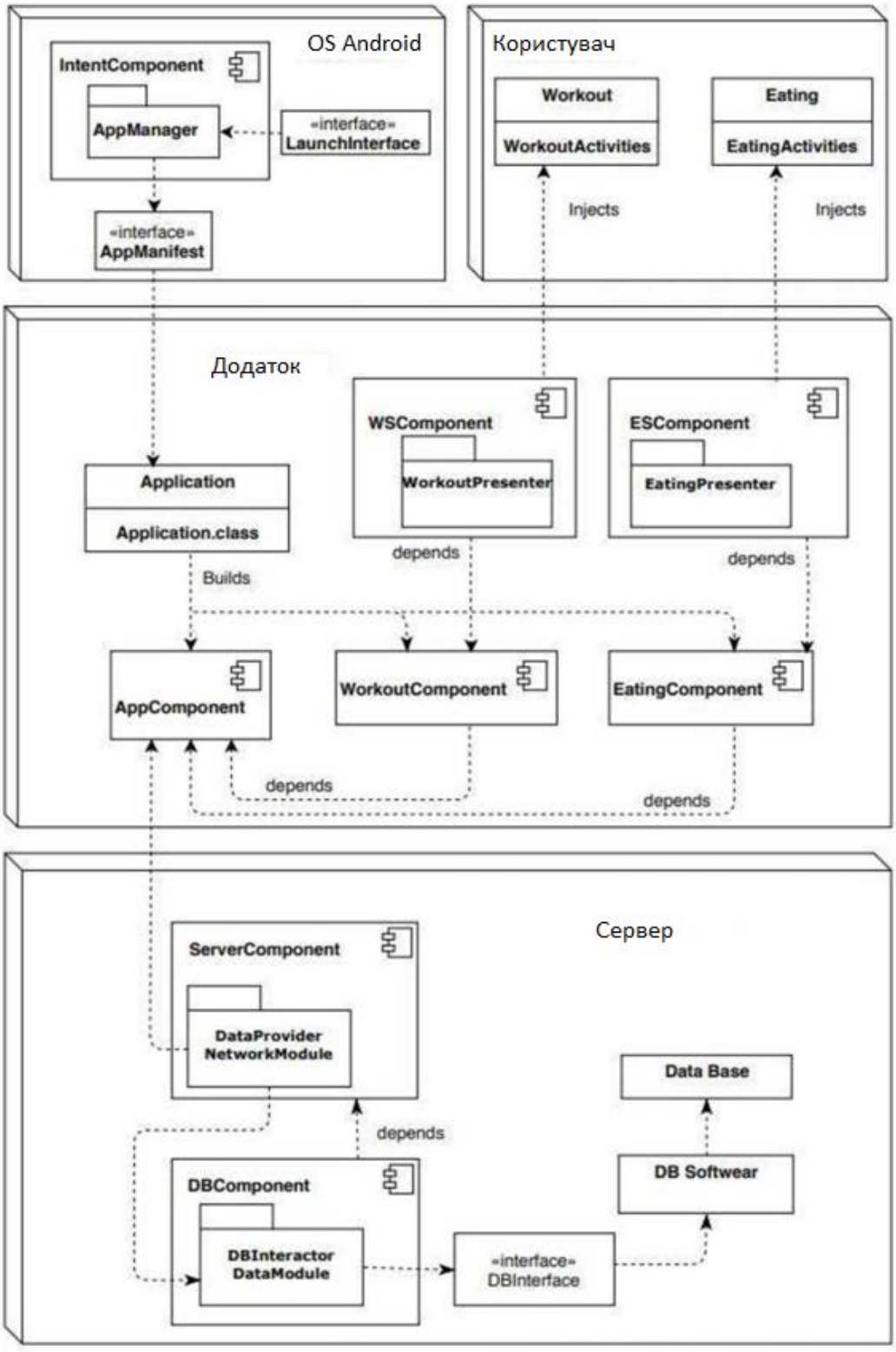


Рисунок 2.7 – Архітектура розроблюваних розділів додатку «Фітнес-тренер».

Діаграма розгортання представлена чотирма вузлами:

1) OSAndroid – операційна система Android.

2) Користувач – користувач, виконуючий взаємодію з екранами Тренувань і Харчувння.

3) Програма – програма «Фітнес-тренер», за допомогою якої Користувачеві підбираються індивідуальні плани тренувань та харчування. Здійснює взаємодію між користувачем, операційною системою та сервером.

4) Сервер – використовує систему управління базами даних, що надає користувачеві дані про тренування та харчування.

Щоб здійснювати зв'язок між програмою та сервером, необхідно мати загальний формат представлення даних. Передача даних між програмою та сервером здійснюється за допомогою JSON (JavaScript Object Notation).

Програма, реагуючи на дії користувача, відправляє дані на сервер, використовуючи протокол передачі гіпертексту https. Дані передаються у форматі JSON, після чого сервер обробивши дані повертає необхідну відповідь додатку також у форматі JSON.

Розглянемо взаємодію клієнта з сервером на основі прецеденту відправки даних про проходження тестування: після проходження тесту всі зібрані відповіді користувача відправляються на сервер, далі обробляються і за допомогою фільтрації сервер повертає дані в залежності від тесту – програма харчування або тренувань.

1.4 Проектування плану тренувань з допомогою машинного навчання

Планування тренувань вирішується шляхом використання задачі багатокласової класифікації, де обрані властивості особистості відносяться до того чи іншого класу.

Для реалізації вирішення задачі класифікації будемо використовувати мову програмування Python, так як це мінімалістична та інтуїтивно зрозуміла мова з повнофункціональною бібліотечною лінією, яка значно скорочує час, необхідний для отриманнярезультатів.Для реалізування алгоритмів будемо використовувати бібліотеки: scikit-learn (sklearn), NumPy та Pandas.Зробимо короткий огляд бібліотек.

Scikit-learn – один з найбільш широко використовуваних пакетів Python. Він дозволяє виконувати безліч операцій та надає безліч алгоритмів. Scikit-learn також пропонує чудову документацію про свої класи, методи та функції, а також опис алгоритмів, що використовуються.

NumPy - це бібліотека мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом із великою бібліотекою високорівневих (і дуже швидких) математичних функцій для операцій із цими масивами.

Pandas це високорівнева бібліотека Python для аналізу даних, вона називається високорівневою, тому що побудована вона поверх нижчого рівня бібліотеки NumPy (написана на Сі), що є великим плюсом у продуктивності. В екосистемі Python, pandas є найбільш просунутою бібліотекою, що швидко розвивається, для обробки та аналізу даних.

Отже, обравши необхідний інструментарій для реалізації вирішення задачи, перейдемо до наступного кроку.

Відповідно доCRISP-DM методології, планування тренувань передбачає здійснення наступних етапів:

Етап 1. Бізнес-аналіз, де визначається мета створення додатку, ресурс фінанси тощо, завдання, планування.

Етап 2. Первинний аналіз даних передбачає збір даних за допомогою соціологічного тесту, де використовують критерії Кі, зміст яких залежить від програми тренування та узгоджується із державними стандартами. За необхідноті здійснюють детальний аналіз вхідних даних на предмет якості тощо.

Етап 3. Підготовка даних передбачає формування навчальної та тестової вибірок за допомогою функцій бібліотети Scikit Learn, зокрема train_test_split. За наявності підготовлених даних тренування додатково реалізовано перевірки на предмет існування помилок. Підібрані дані стосовно тренування за необхідності шкалюють.

Етап 4. Моделювання, передбачає навчання моделей тренування та вибір алгоритма інформаційної технології. За обраним алгоритмом проводять навчання досліджуваної моделі та визначають її якість. У вказаному дослідженні розглядається алгоритм класифікації логістичної регресії. Алгоритм логістичної регресії було обрано, бо він має тикі переваги, як: висока швидкість, що підходить для двох завдань класифікації; просто та зрозуміло, ви можете безпосередньо побачити вагу кожної функції; можна легко оновити модель для поглинання нових даних.

Етап 5. Рекомендації стосовно плану тренувань.

1.5 Розробка макету

Однією з функціональних вимог є зручний та інтуїтивно зрозумілий інтерфейс. Розробимо макети користувацького інтерфейсу мобільного додатку за допомогою векторного графічного редактора FIGMA та наведемо їх нижче (Рис. 2.9 – 2.12).

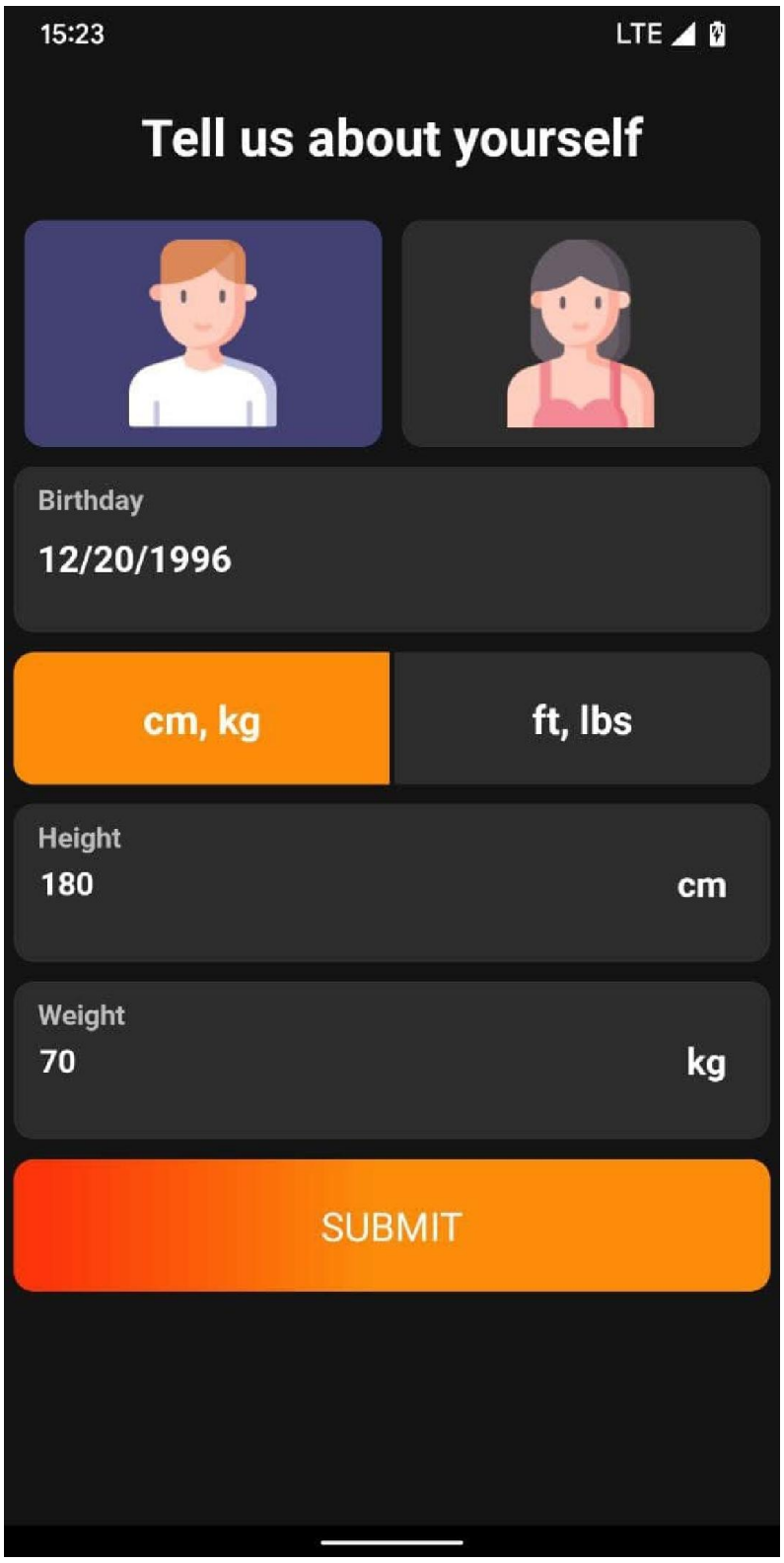


Рисунок 2.9 – Вікно заповнення профілю

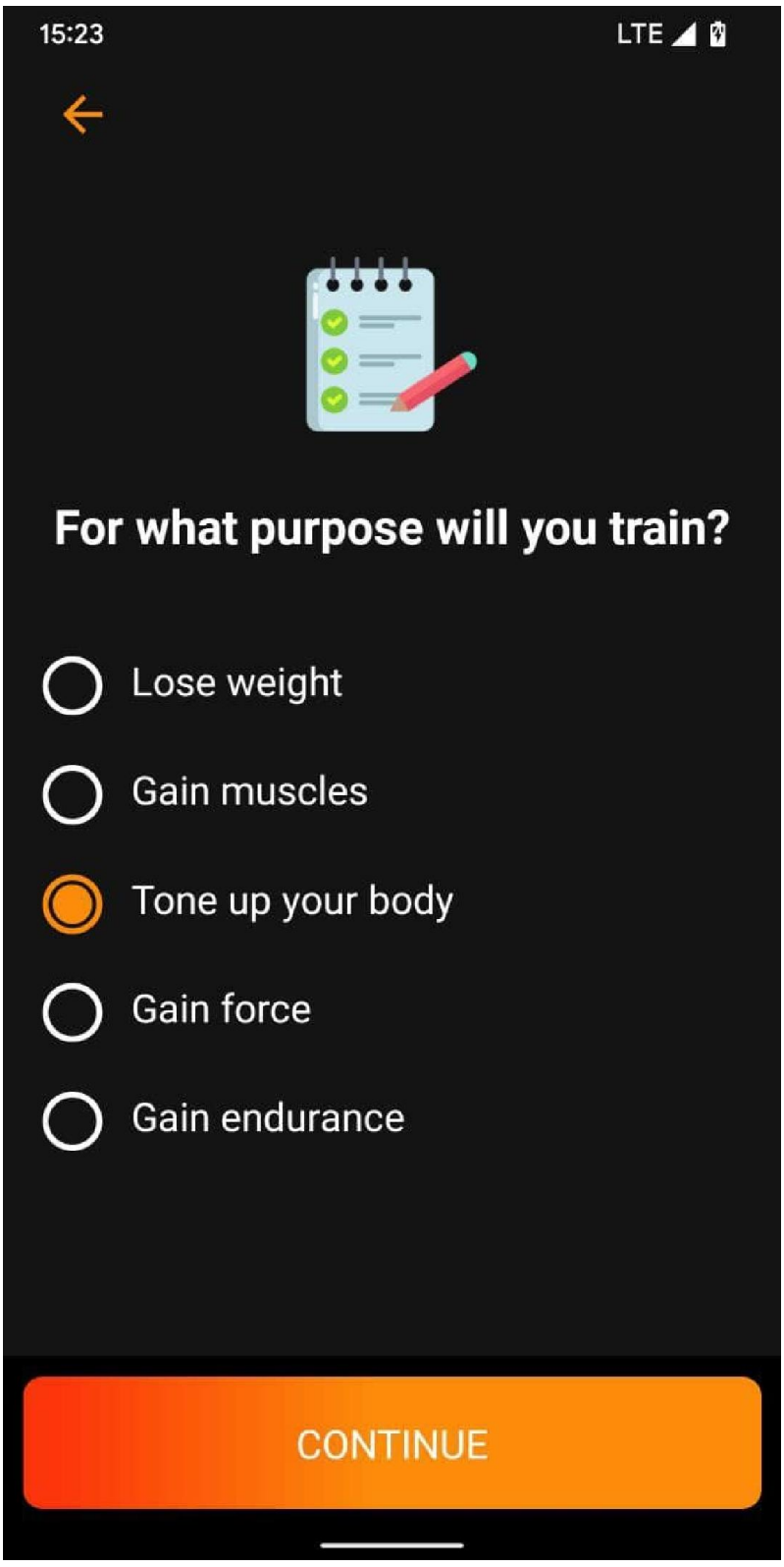


Рисунок 2.10 – Вікно проходження тесту

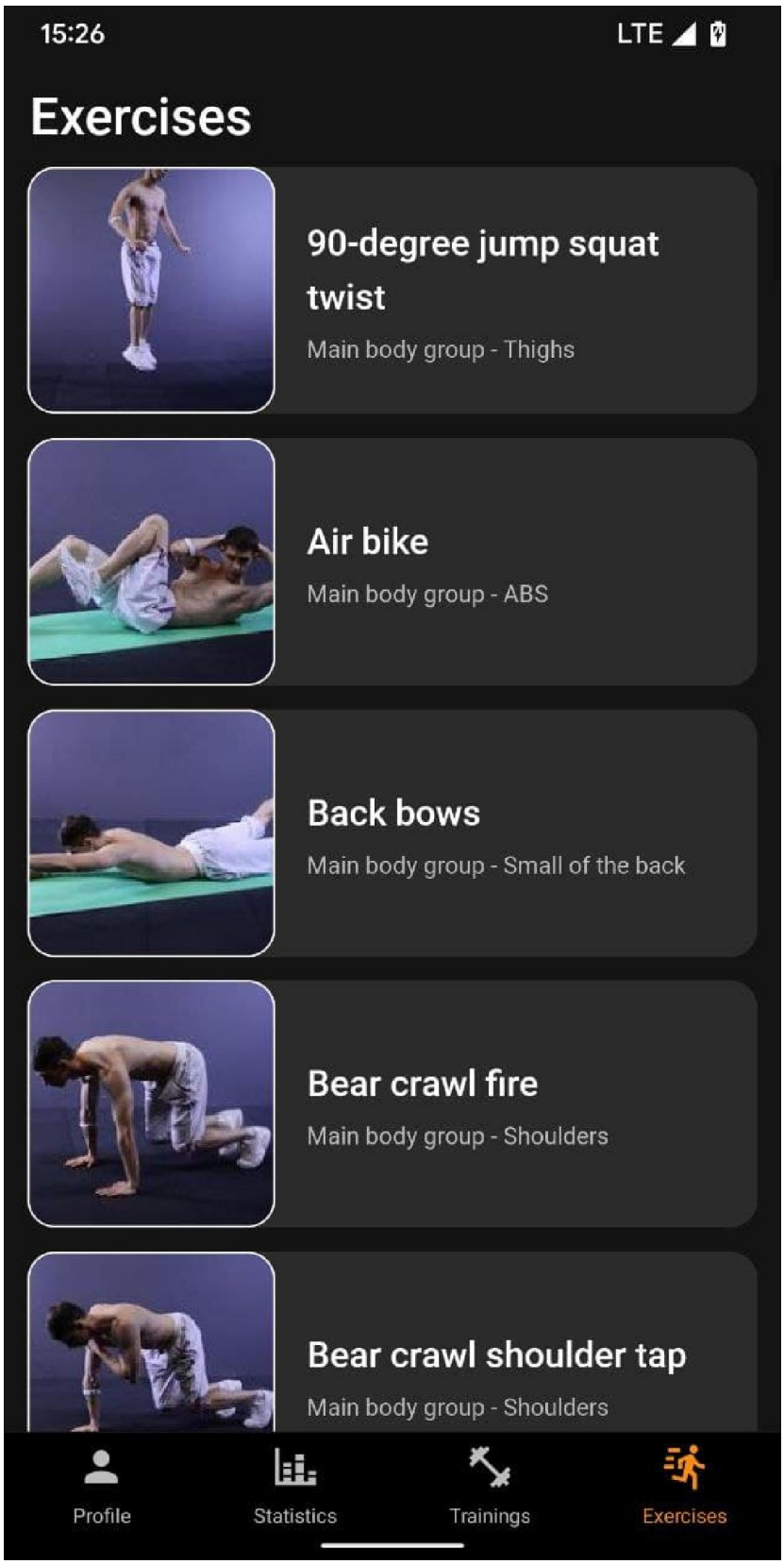


Рисунок 2.11 – Вікно з переліком вправ

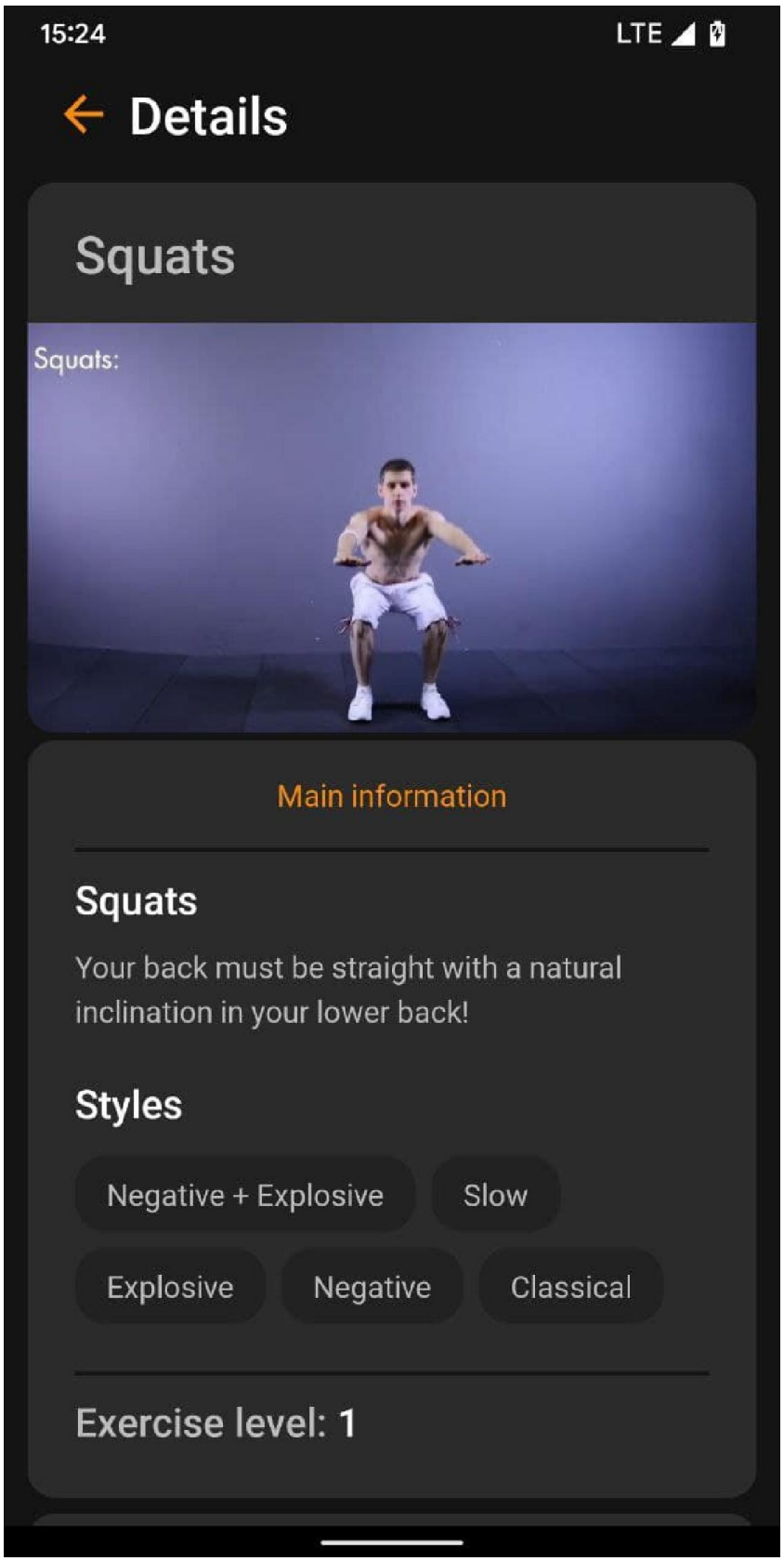


Рисунок 2.12 – Вікно з виконанням вправи

РОЗДІЛ 3
РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ПЕРСОНАЛЬНОГО ТРЕНЕРА

1.6 Вибір платформи

Android – операційна система для смартфонів і безлічі інших пристроїв. Спочатку система розроблялася каліфорнійською компанією Android Inc., яку потім купив американський пошуковий гігант Google. Частка Android на ринку ОС складає 66,71%. Кількість додатків для Андроїд в магазині додатків Google Play перевищує 1,43 млн. Ця операційна система має найбільшу аудиторію користувачів, а, отже, як було сказано вище, приверне більше користувачів. Акаунт розробника для публікації додатка в Play Маркет теж платний, проте коштує набагато менше ніж в App Store і платіж здійснюється одноразово (в App Store щорічний платіж). Проаналізувавши найпопулярніші мобільні операційні системи було вирішено, що розробка буде виконуватися під пристрої з ОС Android, так як вона найбільш відповідає заявленим вимогам.

Найпопулярнішими середовищами розробки під операційну систему Android на сьогоднішній час є:

- Eclipse;
- Microsoft Xamarin;
- Android Studio.

Eclipse є безкоштовною програмною платформою з відкритим вихідним кодом, контролюється організацією Eclipse Foundation. Написана на мові програмування Java і основною метою її створення є підвищення продуктивності процесу розробки програмного забезпечення. Eclipse сама по собі не є середовищем розробки додатків для мобільних пристроїв, але до неї можна підключити окремий плагін ADT (Android Developtent Tools).

Тим не менш, дане середовище розробки не орієнтоване саме на розробку мобільних додатків, тому можливі різні проблеми на етапі розробки, яких можна уникнути, вибравши інше середовище розробки.

Microsoft Xamarin – це платформа розробки мобільних додатків для створення нативних додатків iOS, Android і Windows із загального коду C# або NET, яка дозволяє багаторазово використовувати між платформами від 75% до майже 100% коду. Програми, написані за допомогою Xamarin і C#, мають повний доступ до інтерфейсів API базової платформи можливість створювати нативні призначені для користувача інтерфейси, а також компілювати код в машинний, тому вплив на продуктивність під час виконання є незначним [4].

Розробники, знайомі з C#, .NET і Visual Studio, можуть розраховувати на такі ж можливості і продуктивність при роботі з Xamarin для мобільних додатків, включаючи віддалене налагодження на пристроях Android, iOS і Windows, без необхідності вивчати нативні мови, наприклад, Objective-C або Java. Дивно, але багато високопродуктивних додатків з красивими користувацькими інтерфейсами – наприклад, NASCAR, Aviva та Mix Radio створені за допомогою Xamarin.

Однак через невисоку популярність даного середовища розробки та передбачуваними проблемами з пошуком інформації– віднеї довелося відмовитися

Android Studio – це інтегроване середовище розробки (IDE) для роботи з платформою Android, анонсована 16 травня 2013 року на конференції Google I/O [6]. Це молоде середовище розробки, але вже дуже популярне серед розробників під ОС Android, Середовище має ряд позитивних особливостей. Ось деякі з них:

– розширений редактор макетів: WYSIWYG, здатний працювати з UI компонентами за допомогою Drag-and-Drop, функція попереднього перегляду Макстіл на декількох конфігураціях екрану;

– збірка програм, заснована на Gradle;

– різні види збірок і генерація кількох apk файлів;

– рефакторинг коду;

– статичний аналізатор коду (Lint), що дозволяє знаходити проблеми продуктивності, несумісності версій і інше;

– шаблони основних макетів і компонентів Android;

– підтримка розробки додатків для Android Wear і Android TV;

– Android Studio 2.1 підтримує Android N Preview SDK, а це значить, що розробники зможуть почати роботу зі створення програми для нової програмної платформи.

Також, дане середовище розробки, як випливає з назви, орієнтоване на розробку додатків саме під ОС Android, що є істотною перевагою перед іншими IDE. У підсумку, після аналізу існуючих середовищ розробки, було обране середовище розробки Android Studio, так як воно ідеально підходить під потреби розробляємого програмного продукту, а також у вільному доступі є багато інформації по його налаштуванню і роботі з ним.

1.7 Реалізація розділів тренувань та харчування у мобільному додатку

В результаті розробки програмних модулів деякі частини дизайну зазнали змін, але загалом кінцевий продукт відповідає розробленим макетам (Рис. –).

Для того, щоб отримати доступ до функціоналу мобільного додатка, користувачеві необхідно зареєструватися в мобільному додатку «Фітнес-тренер», або якщо у користувача вже є обліковий запис, то авторизуватися.

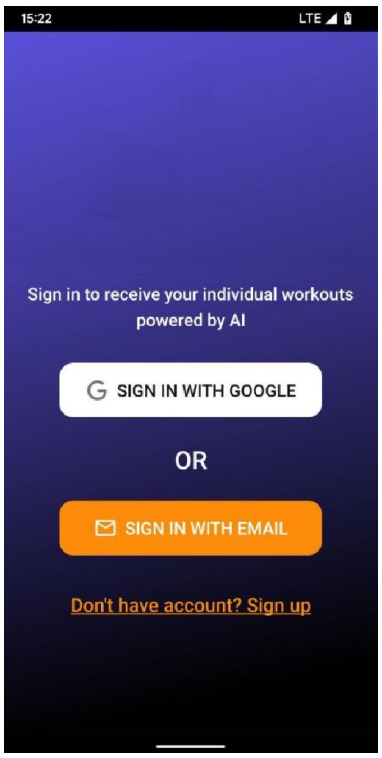


Рисунок 3.1 – Вікно авторизації

Після авторизації нам показується вікно, з можливістю заповнення особистого профілю. Для заповнення знадоюляться такі дані, як: стать, дата народження, зріст, вага. Введені дані можна змінити у налаштуваннях профілю.

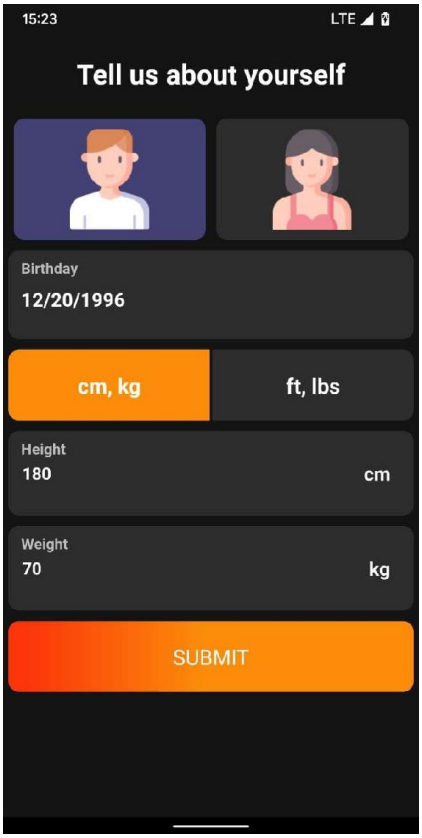


Рисунок 3.2 – Заповнення профілю

Після заповнення даних профілю відкривається вікно тренувань, де перед початком тренування нам пропонується пройти тест для формування програми тренувань(Рис. N). Також нам доступний перехід до розділів «Профіль», «Статистика», «Вправи».

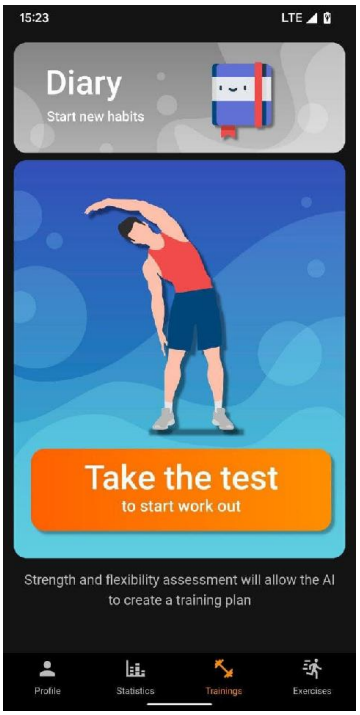


Рисунок 3.3 – Пропозиція пройти тест

При натисканні на кнопку «Пройти тест» розпочинається проходження тесту(Рис. –) Після проходження тесту складається план тренувань.

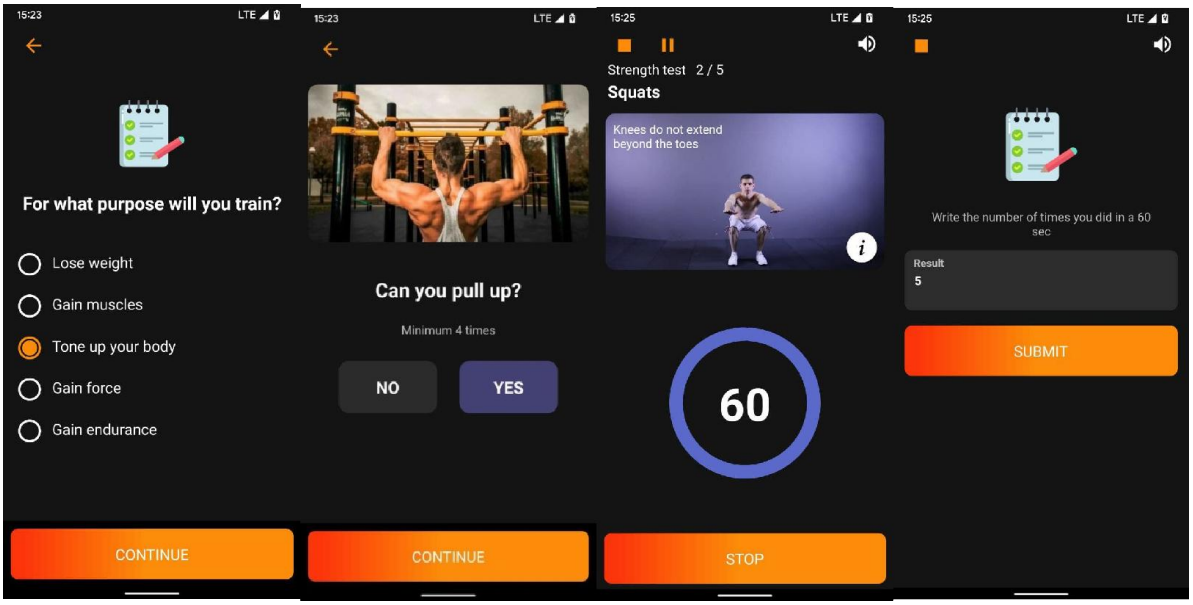
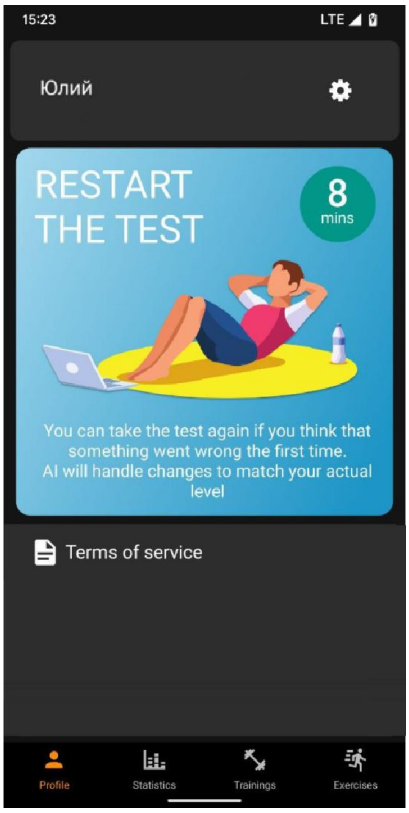


Рисунок 3.4– 3.7 – Проходження тесту

При вереході на екран «Профіль», нам дається змога пройти тестування наново та змінити мову, пароль, або вийти з облікового запису.



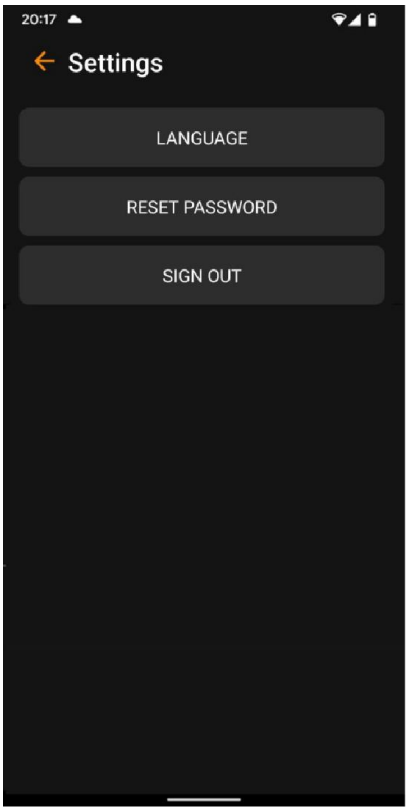


Рисунок 3.8 – 3.9 – Налаштування профілю

При переході до екрану «Статистика» ми можемо обрати ступінь відновлення певної групи м'яз.

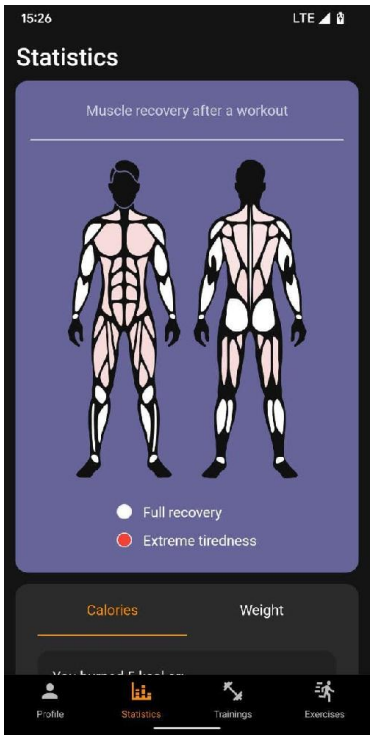


Рисунок 3.10 – Екран статистики

При вереході на екран «Вправи», нам дається змога обрати одну з вправ да виконати її.

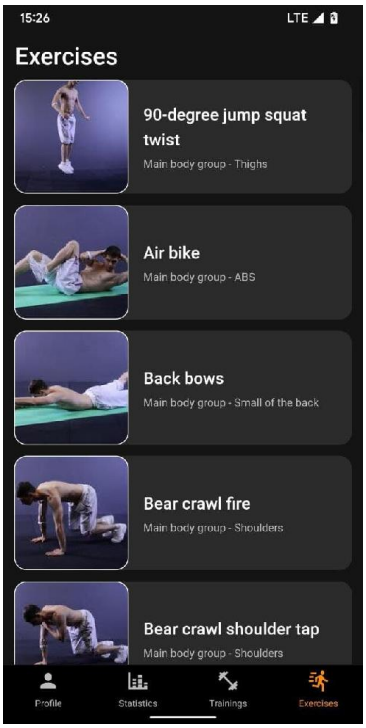


Рисунок 3.11 – Екран вправ

РОЗДІЛ 4
ТЕСТУВАННЯ МОБІЛЬНОГО ДОДАТКУ «ФІТНЕС-ТРЕНЕР»

4.1 Вибір виду тестування

Тестування – заключний етап розробки програмного продукту. В ході тестування передбачається перевірка відповідності між реальною і очікуваною поведінкою програми, що здійснюється на кінцевому наборі тестів, обраному певним чином. У більш широкому сенсі, тестування - це одна з технік контролю якості, що включає в себе активності з планування робіт (Test Management), проектування тестів (Test Design), виконання тестування (Test Execution) і аналізу отриманих результатів (Test Analysis). [7]

Метою тестування є підвищення вірогідності того, що розробляємий продукт буде функціонувати коректно при будь-яких обставинах. Також метою тестування є перевірка того, що розробляємий продукт відповідає всім описаним вимогам.

Для тестування розроблених модулів скористаємося ручним тестуванням, застосовуючи як негативний, так й позитивний підхід.

4.2 Тест план

Тестування розробляємого продукту здійснювалось на різних версіях операційної системі Android, нижче наведений список:

- Android 4.2 (API level 17);
- Android 5.0 (API level 21);
- Android 7.1 (API level 25);
- Android 10 (API level 29).

Тестування проводилось за допомогою емулятору Android пристроїв, що поставляється у складі Android SDK. Також тестування проводилось на пристроях Google Pixel та BlackBerry Classic(Q20). Останній пристрій працює на операційній системі Blackberry OS 10.3.3, але має змогу емулювати ОС

Android версії 4.2, що дає можливість використовувати додаток на телефонах Блекбері.

Для того, щоб провести тестування, складемо тест-плани та визначимо функції, що будуть протестовані.

- коректна реєстрація;
- некоректна реєстрація;
- коректна авторизація;
- некоректна авторизація;
- навігація між вправами;
- навігація між тренуваннями;
- вибір тренування;
- проходження тестування;
- редагування профілю;

Розробимо тест-кейси, які необхідно перевірити. Враховуючи те, що різні типи даних мають однаковий базовий функціонал вони будуть мати однакові тест-кейси (Табл. 4.1).

Таблиця 4.1 – Тест-кейси перевірки коректної реєстрації

Опис	Перевірка коректності роботи реєстрації	
Передумови	Відкрито вікно реєстрації	
1	2	3
№	Дія	Очікуваний результат
1	Ввести в відповідне поле коректну електронну пошту.	Введено коректну електронну пошту.
2	Ввести в відповідне поле пароль.	Введено пароль.
3	Натиснути кнопку «Зареєструватися».	Виконана реєстрація та відкрито вікно з заповненням профілю.

Таблиця 4.2 – Тест-кейс перевірки введення некоректної електронної пошти при реєстрації

Опис	Перевірка некоректного введення електронної пошти при реєстрації	
Передумови	Відкрито вікно реєстрації	
№	Дія	Очікуваний результат
1	Ввести в відповідне поле некоректну електронну пошту.	Введено некоректну електронну пошту.
2	Ввести в відповідне поле пароль.	Введено пароль.
3	Натиснути кнопку «Зареєструватися».	Реєстрація невиконана, спливає повідомлення: «Введіть існуючу пошту».

Таблиця 4.3 – Тест-кейси перевірки коректної авторизації

Опис	Перевірка коректності роботи авторизації	
Передумови	Відкрито вікно авторизації	
1	2	3
№	Дія	Очікуваний результат
1	Ввести в відповідне поле коректну електронну пошту.	Введено коректну електронну пошту.
2	Ввести в відповідне поле коректний пароль.	Введено пароль.
3	Натиснути кнопку «Увійти».	Виконана авторизація та відкрито вікно зі списком тренувань

Таблиця 4.4 – Тест-кейс перевірки введення некоректної електронної пошти при авторизації

Опис	Перевірка некоректного введення електронної пошти при авторизації	
Передумови	Відкрито вікно авторизації	
№	Дія	Очікуваний результат
1	Ввести в відповідне поле некоректну електронну пошту.	Введено некоректну електронну пошту.
2	Ввести в відповідне поле коректний пароль.	Введено коректний пароль.
3	Натиснути кнопку «Увійти».	Авторизація невиконана, спливає повідомлення: «Введіть коректну пошту».

Таблиця 4.5 – Тест-кейс перевірки введення некоректного пароля при авторизації

Опис	Перевірка некоректного введення пароля при авторизації	
Передумови	Відкрито вікно авторизації	
№	Дія	Очікуваний результат
1	Ввести в відповідне поле коректну електронну пошту.	Введено коректну електронну пошту.
2	Ввести в відповідне поле некоректний пароль.	Введено некоректний пароль.
3	Натиснути кнопку «Увійти».	Авторизація невиконана, спливає повідомлення: «Введіть коректний пароль».

Таблиця 4.6 – Тест-кейси для перевірки навігації фітнес-тренера

Опис	Перевірка навігації фітнес-тренера при першому запуску	
Передумови	Відкрито вікно з введенням параметрів користувача	
1	2	3
№	Дія	Очікуваний результат
1	Заповнити форму з параметрами користувача.	Переходимо на сторінку тестування.
2	Розпочати тестування для складання плану тернувань.	Пройти тестування, отримати план тренувань.
3	Натиснути кнопку «Виконти тренування» за складеним планом тренувань.	Розпочинається тренування. Виконуємо разминку. Виконуємо запропоновані вправи. Закінчуємо тренування.
4	Обрати вкладку «Статистика».	Переходимо на сторінку статистики, відмічаємо рівень виснаженості після тренування.
5	Обрати один з двох рівнів складності підготовки.	Обираємо рівень складності.
Опис	Перевірка навігації фітнес-тренера при подальших запусках	
Передумови	Відкрито вікно з планом тренувань	
№	Дія	Очікуваний результат
1	Обрати тренування.	Відкривається вікно з обраним тренуванням.

Таблиця 4.7 – Тест-кейси для довільного виконання вправ

Опис	Довільне виконання вправ	
Передумови	Відкрито вікно з вправами	
1	2	3
№	Дія	Очікуваний результат

Продовження таблиці 4.7

1	2	3
1	Обрати вправу.	Відкривається вікно з вправою
2	Розпочати виконання вправи.	При натисканні на відповідну кнопку відкривається вікно з описом та технікою виконання вправи.

4.3Введення в експлуатацію

Після того, як мобільний додаток буде готовий до введення в експлуатацію, в Android Studio додаток компілюється в APK файл, який ми можемо розповсюджувати та встановлювати на Android пристрої.

Android - відкрита платформа, яка пропонує широкі можливості. Користувач сам обирає спосіб розповсюдження програми, відповідний його потребам, від публікації в магазині додатків до розміщення на сайті або відправки електронною поштою.

Процедура розробки та підготовки файлу APK до випуску однакова для всіх додатків і не залежить від моделі поширення. Це дозволяє заощадити час і при необхідності автоматизувати ті чи інші процеси.

Як правило, публікація в магазині додатків (наприклад, в Google Play) дозволяє охопити найбільш широку аудиторію.

Google Play - це найбільший магазин додатків Android, який надає доступ до всесвітньої аудиторії, але використовувати його не обов’язково: ми можемо опублікувати додаток на іншому майданчику або на декількох одночасно. [8]

На відміну від інших моделей поширення, Google Play дозволяє використовувати сервіс продажу контенту через додатки та сервіс ліцензування. Сервіс продажу контенту через додаток спрощує розповсюдження товарів, що продаються в додатку, наприклад ігрової валюти або розширених функцій. Сервіс ліцензування допомагає запобігти незаконне

встановлення і використання додатків. Проте треба зауважити, що публікація тут не безкоштовна, за акаунт розробника доведеться заплатити 25$.

Якщо розробник не хоче розміщувати свій продукт в магазині додатків, наприклад в Google Play, він можете опублікувати його на власному сайті або сервері (приватному чи корпоративному). Для цього спочатку готується додаток до публікації, а потім розміщуєте готовий файл APK на сайті і додається посилання для його скачування. При цьому користувачам необхідно дозволити встановлення невідомих додатків.

ВИСНОВОК

Мета дипломної роботи полягала в створенні мобільного додатку фітнес-тренера. Дана розробка надає користувачеві можливість пройти тестування, результат якого дозволить обрати план тренувань, який було складено за допомогою нейронної мережі та тренуватися без участі тренера.

В ході написання дипломної роботи було сформовано предметну область та функціональні вимоги до проекту. Також було розглянуто існуючі програмні рішення. В процесі розгляду були встановлені висновки: додаток повинен мати зручний та інтуїтивно зрозумілий інтерфейс та у додатку повинна бути реалізована автономна робота, без підключення до інтернету.

Також були описані фунціонал та структура мобільного додатку. Основними функціями додатку є: сладення плану тернувань на основі результатів, отриманих з проходження користувачем тесту. Також користувач може створювати влсне тренування, обираючи вправи з бібліотеки. У додатку користувач має можливість обрати мову інтерфейсу.

Було розроблено макети дизайну всіх сторінок додатку: сторінки з реєстрацією та авторизацією, сторінки особистого кабінету, сторінки з планом тренувань, сторінки з окремими вправами.

Далі була обрана платформа, яка використовувалася в процесі створення мобільного додатку, а саме – Android Studio.

Далі було розроблено план тренувань та користувацький інтерфейс додатку.

Також було створено сім тест-кейсів для тестування всіх основних функцій мобільного додатку. Для тестування додатку було обрано ручне тестування з використанням як негативного, так і позитивного підходу.

Для введення в експлуатацію, мобільний додаток може поширюватися у виді інсталяційного APK файлу або бути розміщеним в магазині додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кочубей Р.В. Організаційно-економічні проблеми трансформації спортивної галузі України.Механізм регулювання економіки. 2016. № 3. С. 78-87.

2. Рабимов Н.Р., Туракулов І. Мобильные приложения и их роль в жизни современного человека. Достижения науки и образования. 2019. С. 1-3.

3. 7 лучшихфитнес-приложений для Android [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://rskrf.ru/tips/obzory-i-topy/7-luchshikh-fitnes-prilozheniy-dlya-android (дата звернення 09.10.2021). – Назва з екрана.

4. Visual Studioand Xamarin [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://msdn.microsoft.com/ru-ru/library/mt299001.aspx (дата звернення 23.10.2021). – Назва з екрана.

5. Android Studio [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу : https://ru.wikipedia.org/wiki/Android_Studio (дата звернення 19.11.2021). – Назва з екрана.

6. OneSignal [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу : https://documentation.onesignal.com/docs (дата звернення 19.10.2021). – Назва з екрана.

7. Легка атлетика. Підручник для тренерів / [Оэолин Н. Г., Хоменков Л. С., Петровский В. В и др.]; под ред. Л. С. Хоменкова – [2-е изд.]. – М. : Физкультура и спорт, 1982. – 481с.

8. Тестирование. Фундаментальная теория [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу : https://habr.com/ru/post/279535 (дата звернення 22.09.2021). – Назва з екрана.

9. Android Developers Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу : https://developer.android.com/distribute/marketing-tools/alternative-distribution (дата звернення 23.09.2021). – Назва з екрана.

10. Легкая атлетика : Учебник для студентов педагогических институтов : учебник / [Макаров А. Н., Сирис П. З., Теннов В. Пи др.]; под ред. А. Н. Макарова. [2-е изд.]. М. : Просвещение, 1987.– 304 с.

11. Бондарчук А. П. Управление тренировочным процессом спортсменов высокого класса / Анатолий Павлович Бондарчук. – М. Олимпия Пресс, 2007. – 197 с.

12. Питание спрортсмена: [сост. Луиз Бурк, Рон Моган]. – М. : Человек, 2012. 54 с.

MVCArchitecture [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://tutorialsteacher.com/(дата звернення 23.09.2021). – Назва з екрана.

13. ASP NET Core documentation [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.microsoft.com/ (дата звернення 23.09.2021). – Назва з екрана.

14. Microservice Architecture [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://microservices.io/ (дата звернення 23.09.2021). – Назва з екрана.

15. Johnson B. Professional Visual Studio 2017 / Bruce Johnson. – 1st ed. – Birmingham: Wrox, 2017. – 864 с.

16. Sells C. Programming WPF / C. Sells, I. Griffiths. – 2nd ed. – Sebastopol: O'Reilly Media, 2007. – 874 с.

17. Hilyard J. C# 6.0 Cookbook: Solutions for C# Developers / J. Hilyard, S. Teilhet. – 4th ed. – Sebastopol: O'Reilly Media, 2015. – 704 с.

18. Goyvaerts J. Regular Expressions Cookbook: Detailed Solutions in Eight Programming Languages / J. Goyvaerts, S. Levithan. – 2nd ed. – Sebastopol: O'Reilly Media, 2012. – 612 с.

19. Nathan A. WPF 4.5 Unleashed / Adam Nathan. – 1st ed. – Indianapolis: Sams Publishing, 2013. – 864 с.

ДОДАТОК А
ВИХІДНИЙ КОД ОСНОВНИХ КОМПОНЕНТІВ

1. Реалізація машинного навчання:

```
#coding=utf-8
import numpy as np
# Данные прочитаны
def loadDataSet(fileName):
    dataMat = []; labelMat = []
    fr = open(fileName)
    for line in fr.readlines():
        lineArr = line.strip().split('\t')
        dataMat.append([float(lineArr[0]), float(lineArr[1])])
        labelMat.append(float(lineArr[2]))
    return dataMat,labelMat

 # Этот процесс означает, что один альфа определен, а другой альфа выбран из остальных, чтобы сформировать процесс выбора двух в алгоритме SMO для оптимизации
 #: Индекс выбранных альфа; номера всех альфа (требуется -1, в конце концов индекс начинается с 0)
 #r: выбранный индекс
def selectJrand(i,m):
    j=i
    while (j==i):
        j = int(np.random.uniform(0,m))
    return j

 # Клип aj, то есть полученный aj находится в диапазоне [L, H], для конкретного метода, пожалуйста, обратитесь к странице метода статистического обучения. 127 7.108
 #: редактируемый a; верхний предел H; нижний предел L
 #r: aj после редактирования
def clipAlpha(aj,H,L):
    if aj > H:
        aj = H
    if L > aj:
        aj = L
    return aj
```

```
 # Используйте алгоритм SMO для SVM
 #: список данных; список меток; коэффициент компромисса (увеличьте коэффициент релаксации и введите штрафной член в функции целевой оптимизации); отказоустойчивость; максимальное количество итераций
 #r: вернуть окончательное значение b и альфа-вектор
def smoSimple(dataMatIn, classLabels, C, toler, maxIter):
    dataMatrix = np.mat(dataMatIn)
    labelMat = np.mat(classLabels).transpose()

    b = 0
    m,n = np.shape(dataMatrix)
    alphas = np.mat(np.zeros((m,1)))
    iter = 0

    while (iter < maxIter):
        alphaPairsChanged = 0

        for i in range(m):
                # Рассчитать прогнозируемое значение, используется формула альфа * y * (x * x) + b, пожалуйста, обратитесь к 7.56 в статистическом методе Ли Хана
            fXi = float(np.multiply(alphas,labelMat).T*(dataMatrix*dataMatrix[i,:].T)) + b
                # Рассчитать ошибку между прогнозируемым значением и фактическим значением
            Ei = fXi - float(labelMat[i])

                # Если условие KKT не выполняется, labelMat [i] * fXi <1 (labelMat [i] * fXi-1 <-toler)
                # и alpha <C или labelMat [i] * fXi> 1 (labelMat [i] * fXi-1> toler) и alpha> 0
                # Затем оптимизируй
            if ((labelMat[i]*Ei < -toler) and (alphas[i] < C)) \
                or ((labelMat[i]*Ei > toler) and (alphas[i] > 0)):
                    # Следующий процесс заключается в выборе другого alphasj и вычислении соответствующих обязательных параметров.
                j = selectJrand(i,m)
                fXj = float(np.multiply(alphas,labelMat).T*(dataMatrix*dataMatrix[j,:].T)) + b
                Ej = fXj - float(labelMat[j])

                alphaIold = alphas[i].copy()
```

```
            alphaJold = alphas[j].copy()
                    # Понимание здесь может относиться к странице 126 статистических методов обучения, которые должны найти верхнюю и нижнюю границы альфа
            if (labelMat[i] != labelMat[j]):
               L = max(0, alphas[j] - alphas[i])
               H = min(C, C + alphas[j] - alphas[i])
            else:
               L = max(0, alphas[j] + alphas[i] - C)
               H = min(C, alphas[j] + alphas[i])

                    если L == H: вывести «L == H»; продолжить # Если верхняя и нижняя границы не изменились, это означает, что нет необходимости обновлять
                    # Рассчитать неотредактированный alphaj по формуле
            eta = 2.0 * dataMatrix[i,:]*dataMatrix[j,:].T \
                  - dataMatrix[i,:]*dataMatrix[i,:].T \
                  - dataMatrix[j,:]*dataMatrix[j,:].T

                    если eta> = 0: вывести «eta> = 0»; продолжить # Если eta> = 0, выпрыгнуть из этого цикла

                    # Формула может ссылаться на статистический метод обучения стр. 127 7.106
            alphas[j] -= labelMat[j]*(Ei - Ej)/eta
                    # Клип альфа
            alphas[j] = clipAlpha(alphas[j],H,L)
                    # Если измененное значение alphaj не сильно меняется, выпрыгните из этого цикла
            if (abs(alphas[j] - alphaJold) < 0.00001): print "j not moving enough"; continue
                    # В противном случае вычислим соответствующее значение альфа
            alphas[i] += labelMat[j]*labelMat[i]*(alphaJold - alphas[j])#update i by the same amount as j
                    # Рассчитать значение b для двух альфа-случаев отдельно
            b1 = b - Ei- labelMat[i]*(alphas[i]-alphaIold)*dataMatrix[i,:]*dataMatrix[i,:].T - labelMat[j]*(alphas[j]-alphaJold)*dataMatrix[i,:]*dataMatrix[j,:].T
            b2 = b - Ej- labelMat[i]*(alphas[i]-alphaIold)*dataMatrix[i,:]*dataMatrix[j,:].T - labelMat[j]*(alphas[j]-alphaJold)*dataMatrix[j,:]*dataMatrix[j,:].T
                    # Если 0 <alphai <C, то b = b1
            if (0 < alphas[i]) and (C > alphas[i]): b = b1
                    # В противном случае, если 0 <alphai <C, то b = b1
            elif (0 < alphas[j]) and (C > alphas[j]): b = b2
```

```
                    # В противном случае, alphai, alphaj = 0 или C
            else: b = (b1 + b2)/2.0
                    # Если вы перейдете к этому шагу, поверхность изменит пару альфа-значений
            alphaPairsChanged += 1
            print "iter: %d i:%d, pairs changed %d" % (iter,i,alphaPairsChanged)
             # Наконец определите, есть ли измененная альфа-пара, и если нет, переходите к
следующей итерации
      if (alphaPairsChanged == 0): iter += 1
      else: iter = 0
      print "iteration number: %d" % iter


       # Вернуть окончательное значение b и альфа-вектор
   return b,alphas
```

2. Реалізація мобільного додатку:

Fitness/.idea/vcs.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<project version="4">
<component name="VcsDirectoryMappings">
<mapping directory="" vcs="" />
</component>
</project>
```

Fitness/ Fitness.iml

```
<?xml version="1.0" encoding="UTF-8"?>
<module external.linked.project.id="Mr.Fitness" external.linked.project.path="$MODULE_DIR$"
external.root.project.path="$MODULE_DIR$" external.system.id="GRADLE"
external.system.module.group="" external.system.module.version="unspecified"
type="JAVA_MODULE" version="4">
<component name="FacetManager">
<facet type="java-gradle" name="Java-Gradle">
<configuration>
<option name="BUILD_FOLDER_PATH" value="$MODULE_DIR$/build" />
<option name="BUILDABLE" value="false" />
</configuration>
</facet>
</component>
<component name="NewModuleRootManager" LANGUAGE_LEVEL="JDK_1_7" inherit-compiler-
output="true">
<exclude-output />
<content url="file://$MODULE_DIR$">
<excludeFolder url="file://$MODULE_DIR$/.gradle" />
</content>
<orderEntry type="inheritedJdk" />
<orderEntry type="sourceFolder" forTests="false" />
</component>
</module>
```

Fitness/app/app.iml

```
<?xml version="1.0" encoding="UTF-8"?>
<module external.linked.project.id=":app" external.linked.project.path="$MODULE_DIR$"
external.root.project.path="$MODULE_DIR$/.." external.system.id="GRADLE"
external.system.module.group="Mr.Fitness" external.system.module.version="unspecified"
type="JAVA_MODULE" version="4">
<component name="FacetManager">
<facet type="android-gradle" name="Android-Gradle">
<configuration>
<option name="GRADLE_PROJECT_PATH" value=":app" />
</configuration>
</facet>
<facet type="android" name="Android">
<configuration>
<option name="SELECTED_BUILD_VARIANT" value="debug" />
<option name="SELECTED_TEST_ARTIFACT" value="_android_test_" />
<option name="ASSEMBLE_TASK_NAME" value="assembleDebug" />
<option name="COMPILE_JAVA_TASK_NAME" value="compileDebugSources" />
<option name="ASSEMBLE_TEST_TASK_NAME" value="assembleDebugAndroidTest" />
<option name="COMPILE_JAVA_TEST_TASK_NAME" value="compileDebugAndroidTestSources"
/>
<afterSyncTasks>
<task>generateDebugAndroidTestSources</task>
<task>generateDebugSources</task>
</afterSyncTasks>
<option name="ALLOW_USER_CONFIGURATION" value="false" />
<option name="MANIFEST_FILE_RELATIVE_PATH" value="/src/main/AndroidManifest.xml" />
<option name="RES_FOLDER_RELATIVE_PATH" value="/src/main/res" />
<option name="RES_FOLDERS_RELATIVE_PATH" value="file://$MODULE_DIR$/src/main/res" />
<option name="ASSETS_FOLDER_RELATIVE_PATH" value="/src/main/assets" />
</configuration>
</facet>
</component>
<component name="NewModuleRootManager" LANGUAGE_LEVEL="JDK_1_7" inherit-compiler-
output="false">
<output url="file://$MODULE_DIR$/build/intermediates/classes/debug" />
<output-test url="file://$MODULE_DIR$/build/intermediates/classes/androidTest/debug" />
<exclude-output />
<content url="file://$MODULE_DIR$">
<sourceFolder url="file://$MODULE_DIR$/build/generated/source/r/debug" isTestSource="false"
generated="true" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/source/aidl/debug" isTestSource="false"
generated="true" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/source/buildConfig/debug"
isTestSource="false" generated="true" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/source/rs/debug" isTestSource="false"
generated="true" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/res/rs/debug" type="java-resource" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/res/resValues/debug" type="java-resource"
/>
```

```
<sourceFolder url="file://$MODULE_DIR$/build/generated/source/r/androidTest/debug"
isTestSource="true" generated="true" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/source/aidl/androidTest/debug"
isTestSource="true" generated="true" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/source/buildConfig/androidTest/debug"
isTestSource="true" generated="true" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/source/rs/androidTest/debug"
isTestSource="true" generated="true" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/res/rs/androidTest/debug" type="java-test-
resource" />
<sourceFolder url="file://$MODULE_DIR$/build/generated/res/resValues/androidTest/debug"
type="java-test-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/debug/res" type="java-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/debug/resources" type="java-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/debug/assets" type="java-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/debug/aidl" isTestSource="false" />
<sourceFolder url="file://$MODULE_DIR$/src/debug/java" isTestSource="false" />
<sourceFolder url="file://$MODULE_DIR$/src/debug/jni" isTestSource="false" />
<sourceFolder url="file://$MODULE_DIR$/src/debug/rs" isTestSource="false" />
<sourceFolder url="file://$MODULE_DIR$/src/main/res" type="java-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/main/resources" type="java-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/main/assets" type="java-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/main/aidl" isTestSource="false" />
<sourceFolder url="file://$MODULE_DIR$/src/main/java" isTestSource="false" />
<sourceFolder url="file://$MODULE_DIR$/src/main/jni" isTestSource="false" />
<sourceFolder url="file://$MODULE_DIR$/src/main/rs" isTestSource="false" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTest/res" type="java-test-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTest/resources" type="java-test-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTest/assets" type="java-test-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTest/aidl" isTestSource="true" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTest/java" isTestSource="true" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTest/jni" isTestSource="true" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTest/rs" isTestSource="true" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/assets" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/blame" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/classes" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/debug" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/dependency-cache" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/dex" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/exploded-
aar/com.android.support/appcompat-v7/23.1.0/jars" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/exploded-
aar/com.android.support/design/23.0.1/jars" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/exploded-
aar/com.android.support/support-v4/23.1.0/jars" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/incremental" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/jniLibs" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/manifests" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/pre-dexed" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/res" />
```

```
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/rs" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/symbols" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/tmp" />
<excludeFolder url="file://$MODULE_DIR$/build/intermediates/transforms" />
<excludeFolder url="file://$MODULE_DIR$/build/outputs" />
<excludeFolder url="file://$MODULE_DIR$/build/tmp" />
</content>
<orderEntry type="jdk" jdkName="Android API 23 Platform" jdkType="Android SDK" />
<orderEntry type="sourceFolder" forTests="false" />
<orderEntry type="library" exported="" name="gson-2.2.4" level="project" />
<orderEntry type="library" exported="" name="support-v4-23.1.0" level="project" />
<orderEntry type="library" exported="" name="design-23.0.1" level="project" />
<orderEntry type="library" exported="" name="appcompat-v7-23.1.0" level="project" />
<orderEntry type="library" exported="" name="support-annotations-23.1.0" level="project" />
<orderEntry type="library" exported="" name="org.apache.http.legacy-android-23" level="project" />
</component>
</module>
```

Fitness/app/build.gradle

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 23
    buildToolsVersion "23.0.1"
    useLibrary 'org.apache.http.legacy'
    defaultConfig {
        applicationId "projectomicron.mrfitness"
        minSdkVersion 15
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    productFlavors {
    }
}
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    testCompile 'junit:junit:4.12'
    // Set this dependency if you want to use Mockito
    testCompile 'org.mockito:mockito-core:1.10.19'
    // Set this dependency if you want to use Hamcrest matching
    testCompile 'org.hamcrest:hamcrest-library:1.1'
    compile 'com.android.support:appcompat-v7:23.1.0'
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile 'com.android.support:design:23.0.1'
```

```
    compile 'com.google.code.gson:gson:2.2.4'
}
```

Fitness/app/proguard-rules.pro

```
# Add project specific ProGuard rules here.
# By default, the flags in this file are appended to flags specified
# in C:\Users\egue7\AppData\Local\Android\sdk/tools/proguard/proguard-android.txt
# You can edit the include path and order by changing the proguardFiles
# directive in build.gradle.
#
# For more details, see
#   http://developer.android.com/guide/developing/tools/proguard.html
# Add any project specific keep options here:
# If your project uses WebView with JS, uncomment the following
# and specify the fully qualified class name to the JavaScript interface
# class:
#-keepclassmembers class fqcn.of.javascript.interface.for.webview {
#   public *;
#}
```

Fitness/app/src/androidTest/java/projectomicron/mrfitness/ApplicationTest.java

```
package projectomicron.mrfitness;
import android.app.Application;
import android.test.ApplicationTestCase;
/**
 * <a href="http://d.android.com/tools/testing/testing_android.html">Testing Fundamentals</a>
 */
public class ApplicationTest extends ApplicationTestCase<Application> {
    public ApplicationTest() {
        super(Application.class);
    }
}
```

Fitness/app/src/main/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="projectomicron.mrfitness" >
<uses-permission
        android:name="android.permission.INTERNET">
</uses-permission>
<uses-permission
        android:name="android.permission.ACCESS_NETWORK_STATE">
</uses-permission>
<application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
<activity android:name=".LoginActivity" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />
```

```
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name=".CreateAccountActivity" >
</activity>
<activity
        android:name=".TabMenuManagerActivity"
        android:label="@string/title_activity_menu"
        android:theme="@style/AppTheme.NoActionBar" >
</activity>
<activity android:name=".ViewFitnessStatsActivity"></activity>
<activity android:name=".ViewWorkOutActivity"></activity>
<activity android:name=".CreateWorkOutActivity"></activity>
</application>
</manifest>
```

Fitness/app/src/main/java/projectomicron/mrfitness/AccountManager.java

```
package projectomicron.mrfitness;
import android.util.Log;
import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.ArrayList;
import java.util.List;
/**
 * This class manages the creation, loading, updating, and deleting of a user's account.
 * Created by Emanuel Guerrero on 11/23/2015.
 */
public class AccountManager {
    /**
     * Instance variables only visible in the AccountManager class.
     */
    private String userName;
    private String passWord;
    private String firstName;
    private String lastName;
    private int userID;
    private int age;
    private int reasonForUse;
    private int trainerID;
    private int certificationNumber;
   private int insuranceNumber;
    /**
     * Instance variables that are only visible in the AccountManager class. Contains the specific
     * URL that contains the PHP script of the webservice.
     */
    private final String CREATEACCOUNT_URL = "http://lamp.cse.fau.edu/~eguerre4/webservice/" +
                                             "createaccount.php";
    private final String LOADACCOUNT_URL = "http://lamp.cse.fau.edu/~eguerre4/webservice/" +
                                           "loadaccount.php";
```

```
private final String UPDATEACCOUNT_URL = "http://lamp.cse.fau.edu/~eguerre4/webservice/" +
                                         "updateaccount.php";
private final String DELETEACCOUNT_URL = "http://lamp.cse.fau.edu/~eguerre4/webservice/" +
                                         "deleteaccount.php";
private final String CHECKUSERNAME_URL = "http://lamp.cse.fau.edu/~eguerre4/webservice/" +
     "checkusername.php";
/**
 * Instance variables that are only visible in the AccountManager class. Contains the JSON
 * element ids from the response of the PHP script.
 */
private final String TAG_SUCCESS = "success";
private final String TAG_MESSAGE = "message";
private final String TAG_USERNAME = "username";
private final String TAG_PASSWORD = "password";
private final String TAG_FIRSTNAME = "firstname";
private final String TAG_LASTNAME = "lastname";
private final String TAG_AGE = "age";
private final String TAG_TRAINERID = "trainerid";
private final String TAG_CERTIFICATIONNUMBER = "certificationnumber";
private final String TAG_INSURANCENUMBER = "insurancenumber";
/**
 * Constructs an empty AccountManager object.
 */
public AccountManager() {
}
/**
 * Constructs an AccountManager object with the information of the user when creating an account
 * and loading a user's account when logged in.
 * @param aUserID the user id of the user who is logged in. This field is not required when
 *               the user creates a account
 * @param aUserName the user name of the user who is logged in or creating an account
 * @param aPassWord the password that user enters when logging in or when creating an account
 * @param aFirstName the first name of the user that is logged in or when creating an account
 * @param aLastName the last name of the user that is logged in or when creating an account
 * @param aReasonForUse the role of a user that can be integer zero for client or integer one
 *                      for trainer
 * @param aTrainerID the id of the trainer that the client is associated with. This is an
 *                   optional field if the user is an trainer
 * @param aCertificationNumber the certification number of the user that has the reasonForUse
 *                             field with an integer of one. This field is required for trainers
 * @param aInsuranceNumber the insurance number of the user that has the reasonForUSe field with
 *                         an integer of one. This field is required for trainers
 * @precondition aUserID > 0 || aUsername == 0
 * @precondition aUserName.size() > 0
 * @precondition aPassWord.size() > 0
 * @precondition aFirstName.size() > 0
 * @precondition aLastName.size() > 0
 * @precondition aAge > 0
 * @precondition aReasonForUse >= 0
 * @precondition aTrainerID > 0 || aTrainerID == 0
```

```
 * @precondition aCertificationNumber > 0 || aCertification == 0
 * @precondition aInsuranceNumber > 0 || aInsuranceNumber ==0
 */
public AccountManager(int aUserID, String aUserName, String aPassWord, String aFirstName,
                String aLastName,int aAge, int aReasonForUse, int aTrainerID,
                int aCertificationNumber, int aInsuranceNumber) {
   this.userID = aUserID;
   this.userName = aUserName;
   this.passWord = aPassWord;
   this.firstName = aFirstName;
   this.lastName = aLastName;
   this.age = aAge;
   this.reasonForUse = aReasonForUse;
   this.trainerID = aTrainerID;
   this.certificationNumber = aCertificationNumber;
   this.insuranceNumber = aInsuranceNumber;
}
/**
 * Gets the user id of the user.
 * @precondition userID > 0
 * @return the user id of the user
 */
public int getUserID() {
   return userID;
}
/**
 * Gets the username of the user.
 * @precondition userName.size() > 0
 * @return the username of the user
 */
public String getUserName() {
   return userName;
}
/**
 * Sets the username of the user.
 * @param aUserName the username the user entered to create a account or change
 * @precondition aUserName.size() > 0
 */
public void setUserName(String aUserName) {
   this.userName = aUserName;
}
/**
 * Gets the password of the user.
 * @precondition passWord.size() > 0
 * @return the password of the user
 */
public String getPassWord() {
   return passWord;
}
/**
```

```
 * Sets the password of the user.
 * @param aPassWord the password the user entered to create a account or change
 * @precondition aPassWord.size() > 0
 */
public void setPassWord(String aPassWord) {
   this.passWord = aPassWord;
}
/**
 * Gets the first name of the user.
 * @precondition firstName.size() > 0
 */
public String getFirstName() {
   return firstName;
}
/**
 * Sets the first name of the user.
 * @precondition aFirstName.size() > 0
 * @return the first name of the user
 */
public void setFirstName(String aFirstName) {
   this.firstName = aFirstName;
}
/**
 * Gets the last name of the user.
 * @precondition lastName.size() > 0
 * @return the last name of the user
 */
public String getLastName() {
   return lastName;
}
/**
 * Sets the last name of the user.
 * @param aLastName the last name the user entered to create a account or change
 * @precondition aLastName.size() > 0
 */
public void setLastName(String aLastName) {
   this.lastName = aLastName;
}
/**
 * Gets the age of the user.
 * @precondition age > 0
 * @return the age of the user
 */
public int getAge() {
   return age;
}
/**
 * Sets the age of the user.
 * @precondition aAge > 0
 */
```

```
public void setAge(int aAge) {
   this.age = aAge;
}
/**
 * Gets the reason for use of the user.
 * @precondition reasonForUse == 1 || reasonForUse == 2
 * @return the reason for use of the user
 */
public int getReasonForUse() {
   return reasonForUse;
}
/**
 * Gets the trainer id of the user.
 * @precondition trainerID > 0
 * @return the trainer id of the user who is a client of the trainer.
 */
public int getTrainerID() {
   return trainerID;
}
/**
 * Gets the certification number of the user who is a trainer.
 * @precondition certificationNumber > 0
 * @return the certification number of the user
 */
public int getCertificationNumber() {
   return certificationNumber;
}
/**
 * Sets the certification number of the user who is a trainer.
 * @param aCertificationNumber the certification number of the user entered to create a account
 *                             or change
 * @precondition aCertificationNumber > 0
 */
public void setCertificationNumber(int aCertificationNumber) {
   this.certificationNumber = aCertificationNumber;
}
/**
 * Gets the insurance number of the user who is a trainer.
 * @precondition insuranceNumber > 0
 * @return the insurance number of the user
 */
public int getInsuranceNumber() {
   return insuranceNumber;
}
/**
 * Sets the insurance number of the user who is a trainer.
 * @param aInsuranceNumber the insurance number of the user entered to create a account or change
 * @precondition aInsuranceNumber > 0
 */
public void setInsuranceNumber(int aInsuranceNumber) {
```

```
        this.insuranceNumber = aInsuranceNumber;
    }
    /**
     * Creates a new account for the user. Note that we do not need to pass a user id because the
     * database will take care of creating one for the user.
     * @precondition userName.size() > 0
     * @precondition passWord.size() > 0
     * @precondition firstName.size() > 0
     * @precondition lastName.size() > 0
     * @precondition reasonForUse == 1 || reasonForUse == 2
     * @precondition trainerID > 0 || aTrainerID == 0
     * @precondition certificationNumber > 0 || aCertification == 0
     * @precondition insuranceNumber > 0 || aInsuranceNumber ==0
     * @return a String value to indicate if the account has been created successfully
     */
    public String createAccount() {
        //Declare a variable to store the JSON message
        String jsonMessage;
        //Build the POST parameters that need to be passed in the request for checking if the
        //username that was entered already exists
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        params.add(new BasicNameValuePair("username", userName));
        //Make the HTTP request to check if the username entered already exists in the database
        Log.d("request!", "starting");
        JSONObject json1 =
JSONWebservice.getInstance().makeHttpRequest(CHECKUSERNAME_URL, params);
        //Check the log for the json response
        Log.d("Checking username", json1.toString());
        //Get the json success tag
        try {
            int success = json1.getInt(TAG_SUCCESS);
            if (success == 0 && json1.getString(TAG_MESSAGE).equals
                                        ("The username is already taken!")) {
                Log.d("Creating Account failed", json1.getString(TAG_MESSAGE));
                jsonMessage = json1.getString(TAG_MESSAGE);
                return jsonMessage;
            }
            else if (success == 0 && json1.getString(TAG_MESSAGE).equals
                                            ("Database query error#1!")) {
                Log.d("Creating Account failed", json1.getString(TAG_MESSAGE));
                jsonMessage = json1.getString(TAG_MESSAGE);
                return jsonMessage;
            }
        }
        catch (JSONException e) {
            e.printStackTrace();
            jsonMessage = "Database query error#1!";
            return jsonMessage;
        }
```

```
        //Add additionally POST parameters that need to be passed in the request for creating the
        //user an account
        params.add(new BasicNameValuePair("password", passWord));
        params.add(new BasicNameValuePair("firstname", firstName));
        params.add(new BasicNameValuePair("lastname", lastName));
        params.add(new BasicNameValuePair("age", Integer.toString(age)));
        params.add(new BasicNameValuePair("reasonforuse", Integer.toString(reasonForUse)));
        params.add(new BasicNameValuePair("trainerid", Integer.toString(trainerID)));
        params.add(new BasicNameValuePair("certificationnumber", Integer.
                                    toString(certificationNumber)));
        params.add(new BasicNameValuePair("insurancenumber", Integer.toString(insuranceNumber)));
        //Make the HTTP request to create the account for the user
        JSONObject json2 =
JSONWebservice.getInstance().makeHttpRequest(CREATEACCOUNT_URL, params);
        //Check the log for the json response
        Log.d("Checking username", json1.toString());
        //Get the json success tag
        try {
            int success = json2.getInt(TAG_SUCCESS);
            if (success == 1) {
                Log.d("Account Created!", json2.getString(TAG_MESSAGE));
                jsonMessage = json2.getString(TAG_MESSAGE);
            }
            else if (success == 0 && json2.getString(TAG_MESSAGE).equals
                                            ("The username is already taken!")) {
                Log.d("Creating Account failed", json2.getString(TAG_MESSAGE));
                jsonMessage = json2.getString(TAG_MESSAGE);
            }
            else if (success == 0 && json2.getString(TAG_MESSAGE).equals
                                              ("Not all fields were entered!")) {
                Log.d("Creating Account failed", json2.getString(TAG_MESSAGE));
                jsonMessage = json2.getString(TAG_MESSAGE);
            }
            else {
                Log.d("Creating Account failed", json2.getString(TAG_MESSAGE));
                jsonMessage = json2.getString(TAG_MESSAGE);
            }
        }
        catch (JSONException e) {
            e.printStackTrace();
            jsonMessage = "Database query error#1!";
            return jsonMessage;
        }
        return jsonMessage;
    }

    /**
     * Loads an account of the user that is logged in.
     * @precondition userid > 0
     * @return a String value to indicate that the user's account has been successfully loaded
```

```
     */
    public String loadAccount() {
        //Declare a variable to store the JSON message
        String jsonMessage;
        //Build the POST parameters that need to be passed in the request for loading the user's
        //account
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        params.add(new BasicNameValuePair("userid", Integer.toString(userID)));
        //Make the HTTP request to load the user's account
        Log.d("request!", "starting");
        JSONObject json = JSONWebservice.getInstance().makeHttpRequest(LOADACCOUNT_URL,
params);
        //Check the log for the json response
        Log.d("Loading user", json.toString());
        //Get the json success tag
        try {
            int success = json.getInt(TAG_SUCCESS);
            if (success == 1 && json.getString(TAG_MESSAGE).equals("Account loaded!")) {
                Log.d("Account loaded", json.getString(TAG_MESSAGE));
                jsonMessage = json.getString(TAG_MESSAGE);
                //Set the fields of the AccountManager object with the loaded data
                this.userName = json.getString(TAG_USERNAME);
                this.passWord = json.getString(TAG_PASSWORD);
                this.firstName = json.getString(TAG_FIRSTNAME);
                this.lastName = json.getString(TAG_LASTNAME);
                this.age = json.getInt(TAG_AGE);
                this.trainerID = json.getInt(TAG_TRAINERID);
                this.certificationNumber = json.getInt(TAG_CERTIFICATIONNUMBER);
                this.insuranceNumber = json.getInt(TAG_INSURANCENUMBER);
            }
            else if (success == 0 && json.getString(TAG_MESSAGE).equals
                                                ("Account does not exist!")) {
                Log.d("Loading account failed", json.getString(TAG_MESSAGE));
                jsonMessage = json.getString(TAG_MESSAGE);
            }
            else {
                Log.d("Connection Error!", json.getString(TAG_MESSAGE));
                jsonMessage = json.getString(TAG_MESSAGE);
            }
        }
        catch (JSONException e) {
            e.printStackTrace();
            jsonMessage = "Database query error#1!";
            return jsonMessage;
        }

        return jsonMessage;
    }

    /**
```

```
    * Updates the account info of the user that is logged in.
    * @precondition userID > 0
    * @precondition userName.size() > 0
    * @precondition passWord.size() > 0
    * @precondition firstName.size() > 0
    * @precondition lastName.size() > 0
    * @precondition age > 0
    * @precondition reasonForUse == 1 || reasonForUse == 2
    * @precondition trainerID > 0 || aTrainerID == 0
    * @precondition certificationNumber > 0 || aCertification == 0
    * @precondition insuranceNumber > 0 || aInsuranceNumber ==0
    * @return a String value to indicate if the account has been updated successfully
    */
   public String updateAccount() {
      //Declare a variable to store the JSON message
      String jsonMessage;
      //Build the POST parameters that need to be passed in the request for checking if the
      //username that was entered already exists
      List<NameValuePair> params = new ArrayList<NameValuePair>();
      params.add(new BasicNameValuePair("username", userName));
      //Make the HTTP request to check if the username entered already exists in the database
      Log.d("request!", "starting");
      JSONObject json1 =
JSONWebservice.getInstance().makeHttpRequest(CHECKUSERNAME_URL, params);
      //Check the log for the json response
      Log.d("Checking username", json1.toString());
      //Get the json success tag
      try {
         int success = json1.getInt(TAG_SUCCESS);
         if (success == 0 && json1.getString(TAG_MESSAGE).equals("The username is already
taken!")) {
            Log.d("Updating Account failed", json1.getString(TAG_MESSAGE));
            jsonMessage = json1.getString(TAG_MESSAGE);
            return jsonMessage;
         }
         else if (success == 0 && json1.getString(TAG_MESSAGE).equals("Database query error#1!"))
{
            Log.d("Updating Account failed", json1.getString(TAG_MESSAGE));
            jsonMessage = json1.getString(TAG_MESSAGE);
            return jsonMessage;
         }
      }
      catch (JSONException e) {
         e.printStackTrace();
         jsonMessage = "Database query error#1!";
         return jsonMessage;
      }

      //Add additionally POST parameters that need to be passed in the request for updating the
      //user an account
```

```
        params.add(new BasicNameValuePair("password", passWord));
        params.add(new BasicNameValuePair("firstname", firstName));
        params.add(new BasicNameValuePair("lastname", lastName));
        params.add(new BasicNameValuePair("age", Integer.toString(age)));
        params.add(new BasicNameValuePair("reasonforuse", Integer.toString(reasonForUse)));
        params.add(new BasicNameValuePair("trainerid", Integer.toString(trainerID)));
        params.add(new BasicNameValuePair("certificationnumber",
Integer.toString(certificationNumber)));
        params.add(new BasicNameValuePair("insurancenumber", Integer.toString(insuranceNumber)));
        params.add(new BasicNameValuePair("userid", Integer.toString(userID)));
        //Make the HTTP request to create the account for the user
        JSONObject json2 =
JSONWebservice.getInstance().makeHttpRequest(UPDATEACCOUNT_URL, params);
        //Check the log for the json response
        Log.d("Updating account", json1.toString());
        //Get the json success tag
        try {
            int success = json2.getInt(TAG_SUCCESS);
            if (success == 1 && json2.getString(TAG_MESSAGE).equals("Account updated
successfully!")) {
                Log.d("Account updated!", json2.getString(TAG_MESSAGE));
                jsonMessage = json2.getString(TAG_MESSAGE);
            }
            else if (success == 0 && json2.getString(TAG_MESSAGE).equals("Not all fields were
entered!")) {
                Log.d("Updating Account failed", json2.getString(TAG_MESSAGE));
                jsonMessage = json2.getString(TAG_MESSAGE);
            }
            else {
                Log.d("Updating Account failed", json2.getString(TAG_MESSAGE));
                jsonMessage = json2.getString(TAG_MESSAGE);
            }
        }
        catch (JSONException e) {
            e.printStackTrace();
            jsonMessage = "Database query error#1!";
            return jsonMessage;
        }
        return jsonMessage;
    }
    /**
     * Deletes the account of the user.
     * @precondition userid > 0
     * @return a String value to indicate if the user's account has been removed
     */
    public String deleteAccount() {
        //Declare a variable to store the JSON message
        String jsonMessage;

        //Build the POST parameters that need to be passed in the request for deleting the user's
```

```
        //account
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        params.add(new BasicNameValuePair("userid", Integer.toString(userID)));

        //Make the HTTP request to delete the user's account
        JSONObject json = JSONWebservice.getInstance().makeHttpRequest(DELETEACCOUNT_URL,
params);
        //Check the log for the json response
        Log.d("Deleting account", json.toString());

        //Get the json success tag
        try {
            int success = json.getInt(TAG_SUCCESS);
            if (success == 1 && json.getString(TAG_MESSAGE).equals("Account deleted successfully!"))
{
                Log.d("Account deleted!", json.getString(TAG_MESSAGE));
                jsonMessage = json.getString(TAG_MESSAGE);
            }
            else if (success == 0 && json.getString(TAG_MESSAGE).equals("User's account does not
exist!")) {
                Log.d("Deleting account failed", json.getString(TAG_MESSAGE));
                jsonMessage = json.getString(TAG_MESSAGE);
            }
            else {
                Log.d("Deleting account failed", json.getString(TAG_MESSAGE));
                jsonMessage = json.getString(TAG_MESSAGE);
            }
        }
        catch (JSONException e) {
            e.printStackTrace();
            jsonMessage = "Database query error#1!";
            return jsonMessage;
        }
        return jsonMessage;
    }
}
```

Fitness/app/src/main/java/projectomicron/mrfitness/CreateWorkOutActivity.java

```
package projectomicron.mrfitness;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

/**
```

```
 * This class is a subclass of AppCompatActivity that facilitates the creattion of a workout.
 * Created by Emanuel Guerrero on 12/2/2015.
 */
public class CreateWorkOutActivity extends AppCompatActivity implements View.OnClickListener {
    /**
     * Instance variables only visible in the CreateWorkOutActivity class. These are the components
     * of the view
     */
    private TextView clientTextBox;
    private TextView errorMessageLabel;
    private EditText treadMillMilesTextBox;
    private EditText treadMillMinutesTextBox;
    private EditText pushUpsTextBox;
    private EditText sitUpsTextBox;
    private EditText squatsTextBox;
    private Button createWorkOutButton;
    private Button goBackButton;
  private ProgressDialog dialog;
    /**
     * Instance variables only visible in the CreateWorkOutActivity class. These are the extra
     * that are access from intents
     */
    private int userID;
    private int reasonForUse;
    private int trainerID;
    private int clientID;
    private String clientName;
    /**
     * Instance variables only visisble in the CreateWorkOutActivity class. These are the values
     * that the user will input.
     */
    private int treadMillMiles;
    private int treadMillMinutes;
    private int pushUps;
    private int sitUps;
    private int squats;
    /**
     * Overrides the onCreate method inherited from AppCompatActivity to get the references for the
     * components in the view.
     * @param savedInstanceState
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //Get the user id, reason for use, and trainer id from the intent
        Bundle extras = getIntent().getExtras();
        userID = extras.getInt("userID");
        reasonForUse = extras.getInt("userRole");
        trainerID = extras.getInt("trainerID");
        clientID = extras.getInt("clientID");
```

```
    clientName = extras.getString("clientName");
    //Get the view of the AppCompatActivity
    setContentView(R.layout.activity_create_workout);
    //Get a reference to the client text box
    clientTextBox = (TextView) findViewById(R.id.clientTextBox);
    //Set the name of the client text box
    clientTextBox.setText(clientName);
    //Get a reference to the error message label
    errorMessageLabel = (TextView) findViewById(R.id.errorMessageLabel);
    //Get references for the text boxes
    treadMillMilesTextBox = (EditText) findViewById(R.id.treadMillMilesTextBox);
    treadMillMinutesTextBox = (EditText) findViewById(R.id.treadMillMinutesTextBox);
    pushUpsTextBox = (EditText) findViewById(R.id.pushUpsTextBox);
    sitUpsTextBox = (EditText) findViewById(R.id.sitUpsTextBox);
    squatsTextBox = (EditText) findViewById(R.id.squatsTextBox);

    //Get references for the buttons
    createWorkOutButton = (Button) findViewById(R.id.editButton);
    goBackButton = (Button) findViewById(R.id.goBackButton);
    //Set onClickListeners for the buttons
    createWorkOutButton.setOnClickListener(this);
    goBackButton.setOnClickListener(this);
}
/**
 *Implements the onClick method in the View.OnClickListener Interface to add the event
 * listeners to the buttons.
 * @param v the view of the activity class
 */
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.editButton:
            //Get the input from the text boxes
            try {
                treadMillMiles = Integer.parseInt(treadMillMilesTextBox.getText().toString());
            }
            catch (NumberFormatException e) {
                treadMillMiles = 0;
            }
            try {
                treadMillMinutes = Integer.parseInt(treadMillMinutesTextBox.getText().toString());
            }
            catch (NumberFormatException e) {
                treadMillMinutes = 0;
            }
            try {
                pushUps = Integer.parseInt(pushUpsTextBox.getText().toString());
            }
            catch (NumberFormatException e) {
                pushUps = 0;
```

```
            }
            try {
                sitUps = Integer.parseInt(sitUpsTextBox.getText().toString());
            }
            catch (NumberFormatException e) {
                sitUps = 0;
            }
            try {
                squats = Integer.parseInt(squatsTextBox.getText().toString());
            }
            catch (NumberFormatException e) {
                squats = 0;
            }
            //Check if any of the input was zero
            if (treadMillMiles < 0 || treadMillMinutes < 0 || pushUps < 0 || sitUps < 0 ||
                    squats < 0) {
                //Let the user know that some of the fields are missing
                errorMessageLabel.setText(R.string.errorMessageTextLabelNegativeValues);
                errorMessageLabel.setVisibility(View.VISIBLE);
            }
            else {
                //Start the background thread to creating the workout for the client
                new CreateWorkOut().execute();
            }
            break;
        case R.id.goBackButton:
            new Thread(new Runnable() {
                @Override
                public void run() {
                    //Set an intent to redirect the user to the TabMenuManagerActivity activity
                    Intent intent = new Intent(getApplicationContext(), TabMenuManagerActivity.class);
                    //Store the userId and user role of the user to be used throughout the application
                    intent.putExtra("userID", userID);
                    intent.putExtra("userRole", reasonForUse);
                    intent.putExtra("trainerID", trainerID);
                    intent.putExtra("clientID", clientID);
                    intent.putExtra("clientName", clientName);
                    //Start the next activity
                    onSwitch(intent);
                }
            }).start();
            break;
        default:
            break;
    }
}
/**
 * Redirects a user to a new activity.
 * @param intent a new activity to start
 */
```

```
public void onSwitch(Intent intent) {
   startActivity(intent);
}
/**
 * This is a subclass of AsyncTask that creates the workout for the client
 */
class CreateWorkOut extends AsyncTask<String, String, String> {
   /**
    * Overrides the onPreExecute method inherited from AsyncTask to show a dialog.
    */
   @Override
   protected void onPreExecute() {
      dialog = new ProgressDialog(CreateWorkOutActivity.this);
      dialog.setMessage("Creating WorkOut...");
      dialog.setIndeterminate(false);
      dialog.setCancelable(true);
      dialog.show();
   }
   /**
    * Implements the doInBackground method in the Interface that AsyncTask implements.
    * @param params auto-generated from the compiler
    * @return a String that indicates if the process is complete
    */
   @Override
   protected String doInBackground(String... params) {
      //Declare a FitnessManager object
      FitnessManager fitnessManager = new FitnessManager();
      //Create the workout for the client and check if it was successfully
      final int workOutID = fitnessManager.createWorkOut(clientID, treadMillMiles,
                              treadMillMinutes, pushUps, sitUps, squats);
      if (workOutID == -1) {
         //Let the user know that not all of the fields were filled
         CreateWorkOutActivity.this.runOnUiThread(new Runnable() {
            @Override
            public void run() {
               //Set the error message label letting the user know that they did not fill
               //all of the fields
               errorMessageLabel.setText(R.string.errorMessageTextLabelMissingField);
               errorMessageLabel.setVisibility(View.VISIBLE);
            }
         });
      }
      else if (workOutID == -2) {
         //Let the user know that there was a connection error
         CreateWorkOutActivity.this.runOnUiThread(new Runnable() {
            @Override
            public void run() {
               //Set the error message label letting the user know that they did not fill
               //all of the fields
               errorMessageLabel.setText(R.string.errorMessageTextLabelConnectionError);
```

```
                    errorMessageLabel.setVisibility(View.VISIBLE);
                }
            });
        }
        else {
            //Let the user know that the workout is created. GIve them the workout ID
            CreateWorkOutActivity.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    //Set the error message label letting the user know that they did not fill
                    //all of the fields
                    errorMessageLabel.setText("WorkOut " + String.valueOf(workOutID) + " was
created.\n" +
                                              "Please click on the go back button");
                    errorMessageLabel.setVisibility(View.VISIBLE);
                }
            });
        }
        return null;
    }
    /**
     * Overrides the onPostExecute method inherited from AsyncTask to close the dialog box.
     * @param s auto-generated from the compiler
     */
    @Override
    protected void onPostExecute(String s) {
        dialog.dismiss();
    }
  }
}
```

Fitness/app/src/main/java/projectomicron/mrfitness/FitnessStats.java

```
package projectomicron.mrfitness;

/**
 * This class represents the contents of a user's fitness stats.
 * Created by Emanuel Guerrero on 12/2/2015.
 */
public class FitnessStats {
    /**
     * Instance variables that are only visible in the FitnessStats class. These represent the
     * contents of a user's fitness stats.
     */
    private int weight;
    private int heightFeet;
    private int heightInches;
    private int BMI;
    /**
     * Constructs a FitnessStats object.
     * @precondition aWeigth > 0
     * @precondition aHeightFeet > 0
```

```
     * @precondition aHeightInches > 0
     * @precondition aBMI > 0
     * @param aWeight the weight of the user
     * @param aHeightFeet the feet part of the height of the user
     * @param aHeightInches the inches part of the height of user
     */
    public FitnessStats(int aWeight, int aHeightFeet, int aHeightInches, int aBMI) {
        this.weight = aWeight;
        this.heightFeet = aHeightFeet;
        this.heightInches = aHeightInches;
        this.BMI = aBMI;
    }
    /**
     * Gets the weight of the user.
     * @precondition weight > 0
     * @return the weight of the user
     */
    public int getWeight() {
        return weight;
    }
    /**
     * Gets the feet part of the user's height.
     * @precondition heightFeet > 0
     * @return the feet component of the user's height
     */
    public int getHeightFeet() {
        return heightFeet;
    }
    /**
     * Gets the inches part of the user's height.
     * @precondition heightInches > 0
     * @return the inches component of the user's height
     */
    public int getHeightInches() {
        return heightInches;
    }
    /**
     * Gets the BMI of the user.
     * @precondition BMI > 0
     * @return the BMI of the user
     */
    public int getBMI() {
        return BMI;
    }
}
```

Fitness/app/src/main/java/projectomicron/mrfitness/JSONWebservice.java

```
package projectomicron.mrfitness;

import android.util.Log;
import org.apache.http.HttpEntity;
```

```
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.List;
/**
 * This class executes http requests to the webservice for loading and saving data to the database.
 * The webservice will a send a response with a JSON string that will be used to create a JSON
 * object. All classes can only have one instance of this class as a singleton.
 * Created by Emanuel Guerrero on 11/11/2015.
 */
public class JSONWebservice {
    /**
     * Instance variable only visible in the JSONWebservice class.
     */
    private static JSONWebservice instance = new JSONWebservice();
    /**
     * Instance fields only visible in the JSONWebservice class.
     */
    private static InputStream is;
    private static JSONObject jsonObject;
    private static String json = "";
    /**
     * Constructs an empty JSONWebservice object.
     */
    private JSONWebservice() {
    }
    /**
     * Gets an instance of the JSONWebservice class.
     * @return a instance of the JSONWebservice class to call methods to access the database
     */
    public static JSONWebservice getInstance() {
        return instance;
    }
    /**
     * Gets a JSON object from the requested URL.
     * @param url the url to run specific PHP script
     * @param  params a set of POST parameters to send in the request
```

```
 * @precondition params.size() > 0
 * @return a JSON object that the object that requested can use
 */
public JSONObject makeHttpRequest(String url, List<NameValuePair> params) {
   //Make HTTP request
   try {
      // Set the timeout in milliseconds until a connection is established.
      HttpParams httpParameters = new BasicHttpParams();
      int timeoutConnection = 3000;
      HttpConnectionParams.setConnectionTimeout(httpParameters, timeoutConnection);
      // Set the default socket timeout (SO_TIMEOUT) in milliseconds which is the timeout for
      // waiting for data.
      int timeoutSocket = 5000;
      HttpConnectionParams.setSoTimeout(httpParameters, timeoutSocket);

      //Request a POST method
      DefaultHttpClient httpClient = new DefaultHttpClient(httpParameters);
      HttpPost httpPost = new HttpPost(url);

      //Set the header to define requiring a JSON String object
      httpPost.setHeader("Accept", "json");
      httpPost.setEntity(new UrlEncodedFormEntity(params));

      //Get a response from the server
      HttpResponse httpResponse = httpClient.execute(httpPost);
      HttpEntity httpEntity = httpResponse.getEntity();
      is = httpEntity.getContent();
   }
   catch (UnsupportedEncodingException e) {
      e.printStackTrace();
   }
   catch (ClientProtocolException e) {
      e.printStackTrace();
   }
   catch (IOException e) {
      e.printStackTrace();
   }
   //Convert the result to a JSON String
   try {
      BufferedReader reader = new BufferedReader(
            new InputStreamReader(is, "utf8"), 8);
      StringBuilder stringBuilder = new StringBuilder();
      String line;
      while ((line = reader.readLine()) != null) {
         stringBuilder.append(line + "\n");
      }
      is.close();
      json = stringBuilder.toString();
   }
   catch (UnsupportedEncodingException e) {
```

```
            e.printStackTrace();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        catch (Exception e) {
            Log.e("Buffer Error", "Error converting result " + e.toString());
        }
        //Try parsing the string to a JSON object
        try {
            jsonObject = new JSONObject(json);
        }
        catch (JSONException e) {
            Log.e("JSON Parser", "Error parsing data " + e.toString());
        }
        //Return the JSON String
        return jsonObject;
    }
}
```

3. Реалізація веб сервісів:

checkusername.php (перевірка даних входу)

```
<?php
//Require the connection to the database to happen
require('connection.php');

if (!empty($_POST)) {
  //Check if the username the user entered already exists in the database
  $query = "SELECT * FROM useraccount WHERE username = :username";
  //Update what :username should be
  $query_params = array(':username' => $_POST['username']);
  //Run the query
  try {
    $statement = $db->prepare($query);
    $result = $statement->execute($query_params);
  }
  catch (PDOException $ex) {
    //Create the data that will be the JSON response
    $response['success'] = 0;
    $response['message'] = "Database query error#1!";
    //Kill the script and send the JSON response to the client
    header('Content-Type: application/json; charset=utf8');
    die(json_encode($response));
  }

  //Fetch the array of returned data and check if any rows were returned
  $row = $statement->fetch();
  if ($row) {
    //Create the data that will be the JSON response
```

```
    $response['success'] = 0;
    $response['message'] = "The username is already taken!";
    //Send the JSON response to the client
    header('Content-Type: application/json charset=utf8');
    echo json_encode($response);
  }
  else {
    //Create the data that will be the JSON response
    $response['success'] = 1;
    $response['message'] = "The username is avaliable!";
    //Send the JSON response to the client
    header('Content-Type: application/json charset=utf8');
    echo json_encode($response);
  }
}
?>
```

webservicescripts/connection.php

```
<?php
//Declare the variables needed to connect to the database
$host = "localhost";
$dbname = "eguerre4";
$username = "eguerre4";
$password = "pizza6576";

//Tell the MySQL server that we will communicate with UTF-8
$options = array(PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8');

//Open a connection to the database using the PDO library
try {
  $db = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8;",
            $username, $password, $options);
}
catch (PDOException $ex) {
  //If an error occurs opening a connection to the database
  die("Connection to database failed: " . $ex->getMessage());
}

//Configures PDO to throw exceptions when an error occurs
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

//Configures PDO to return dayabase rows from the database using an associative
//array
$db->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);

//This is used to undo magic quotes
if (function_exists('get_magic_quotes_gpc') && get_magic_quotes_gpc()) {
  function undo_magic_quotes_gpc(&$array) {
    foreach ($array as &$value) {
      if (is_array($value)) {
        undo_magic_quotes_gpc($value);
```

```
      }
      else {
        $value = stripslashes($value);
      }
    }
  }

  undo_magic_quotes_gpc($_POST);
  undo_magic_quotes_gpc($_GET);
  undo_magic_quotes_gpc($_COOKIE);
}
//session_start();
 ?>
```

webservicescripts/createaccount.php

```
<?php
//Require the connection to the database to happen
require('connection.php');

//Check if the posted data is not empty
if (!empty($_POST)) {
    //Check if any of the required fields are empty
    if (empty($_POST['username']) || empty($_POST['password']) ||
        empty($_POST['firstname']) || empty($_POST['lastname']) ||
        empty($_POST['age']) || empty($_POST['reasonforuse'])) {
      //Create the data that will be the JSON response
      $response["success"] = 0;
      $response["message"] = "Not all fields were entered!";
      //Kill the script and send the JSON response to the client
      header('Content-Type: application/json; charset=utf8');
      die(json_encode($response));
    }
    else {
      //Check if the username already exists in the database
      $query = "SELECT * FROM useraccount WHERE username = :username";
      //Update what :username should be
      $query_params = array(':username' => $_POST['username']);
      //Run the query
      try {
        $statement = $db->prepare($query);
        $result = $statement->execute($query_params);
      }
      catch (PDOException $ex) {
        //Create the data that will be the JSON response
        $response["success"] = 0;
        $response["message"] = "Database query error#1!";
        //Kill the script and send the JSON response to the client
        header('Content-Type: application/json; charset=utf8');
        die(json_encode($response));
      }
```

```
//Fetch the array of returned data and check if any rows were returned
$row = $statement->fetch();
if ($row) {
  //Create the data that will be the JSON response
  $response["success"] = 0;
  $response["message"] = "The username is already taken!";
  //Kill the script and send the JSON response to the client
  header('Content-Type: application/json; charset=utf8');
  die(json_encode($response));
}
else {
  //We can insert the new user into the database
  $query = "INSERT INTO useraccount (username, password, firstname, lastname,
          age, reasonforuse, trainerid, certificationnumber, insurancenumber)
          VALUES (:username, :password, :firstname, :lastname, :age, :reasonforuse,
          :trainerid, :certificationnumber, :insurancenumber)";
  //Update the tokens with the actual data
  $query_params = array(':username' => $_POST['username'],
                  ':password' => $_POST['password'],
                  ':firstname' => $_POST['firstname'],
                  ':lastname' => $_POST['lastname'],
                  ':age' => $_POST['age'],
                  ':reasonforuse' => $_POST['reasonforuse'],
                  ':trainerid' => $_POST['trainerid'],
                  ':certificationnumber' => $_POST['certificationnumber'],
                  ':insurancenumber' => $_POST['insurancenumber']);
  //Run the query
  try {
    $statement = $db->prepare($query);
    $result = $statement->execute($query_params);
  }
  catch (PDOException $ex) {
    //Create the data that will be the JSON response
    $response["success"] = 0;
    $response["message"] = "The username is already taken!";
    //Kill the script and send the JSON response to the client
    header('Content-Type: application/json; charset=utf8');
    die(json_encode($response));
  }
  //Get the user id of the workout that was just inserted
  $query = "SELECT MAX(userid) as userid FROM useraccount";
  //Run the query
  try {
    $statement = $db->prepare($query);
    $result = $statement->execute();
  }
  catch (PDOException $ex) {
    //Create the data that will be the JSON response
    $response["success"] = 0;
    $response["message"] = "Database query error#1!";
```

```
        //Kill the script and send the JSON response to the client
        header('Content-Type: application/json; charset=utf8');
        die(json_encode($response));
      }
      //Fetch the array of returned data and check if any rows were returned
      $row = $statement->fetch();
      if($row) {
        //Insert into the userfitnessstatus table
        $query = "INSERT INTO userfitnessstatus (userid, weight, heightfeet,
                heightinches) VALUES (:userid, :weight,
                :heightfeet, :heightinches)";
        $weight = 0;
        $heightfeet = 0;
        $heightinches = 0;
        //Update the tokens with the actual data
        $query_params = array(':userid' => $row['userid'],
                        ':weight' => $weight,
                        ':heightfeet' => $heightfeet,
                        ':heightinches' => $heightinches);
        //Run the query
        try {
          $statement = $db->prepare($query);
          $result = $statement->execute($query_params);
        }
        catch (PDOException $ex) {
          //Create the data that will be the JSON response
          $response["success"] = 0;
          $response["message"] = "Database query error#1!";
          //Kill the script and send the JSON response to the client
          header('Content-Type: application/json; charset=utf8');
          die(json_encode($response));
        }

      //Create the data that will be the JSON response
      $response["success"] = 1;
      $response["message"] = "Account has been created successfully!";
      //Send the JSON response to the client
      header('Content-Type: application/json; charset=utf8');
      echo json_encode($response);
    }
   }
  }
}
?>
```

webservicescripts/createworkout.php

```
    <?php
    //Require the connection to the database to happen
    require('connection.php');

    //Check if the posted data is not empty
```

```
if (!empty($_POST)) {
  //Check if any of the required fields are empty
  if (empty($_POST['userid'])) {
      //Create the data that will be the JSON response
      $response["success"] = 0;
      $response["message"] = "Not all fields were entered!";
      //Kill the script and send the JSON response to the client
      header('Content-Type: application/json; charset=utf8');
      die(json_encode($response));
  }
  else {
    //Insert the workout into the database
    $query = "INSERT INTO userworkout (userid, treadmillmiles, treadmillminutes,
          pushups, situps, squats) VALUES (:userid, :treadmillmiles,
          :treadmillminutes, :pushups, :situps, :squats)";
    //Update the tokens with the actually values
    $query_params = array(':userid' => $_POST['userid'],
                  ':treadmillmiles' => $_POST['treadmillmiles'],
                  ':treadmillminutes' => $_POST['treadmillminutes'],
                  ':pushups' => $_POST['pushups'],
                  ':situps' => $_POST['situps'],
                  ':squats' => $_POST['squats']);
    //Run the query
    try {
      $statement = $db->prepare($query);
      $result = $statement->execute($query_params);
    }
    catch (PDOException $ex) {
      //Create the data that will be the JSON response
      $response["success"] = 0;
      $response["message"] = "Database query error#1!";
      //Kill the script and send the JSON response to the client
      header('Content-Type: application/json; charset=utf8');
      die(json_encode($response));
    }
    //Get the workout id of the workout that was just inserted
    $query = "SELECT MAX(workoutid) as workoutid FROM userworkout";
    //Run the query
    try {
      $statement = $db->prepare($query);
      $result = $statement->execute();
    }
    catch (PDOException $ex) {
      //Create the data that will be the JSON response
      $response["success"] = 0;
      $response["message"] = "Database query error#1!";
      //Kill the script and send the JSON response to the client
      header('Content-Type: application/json; charset=utf8');
      die(json_encode($response));
    }
```

```
    //Fetch the array of returned data and check if any rows were returned
    $row = $statement->fetch();
    if ($row) {
      //Create the data that will be the JSON response
      $response["success"] = 1;
      $response["message"] = "Workout has been added successfully!";
      //Get the work out id from the returned rows
      $response['workoutid'] = $row['workoutid'];
      //Send the JSON response to the client
      header('Content-Type: application/json; charset=utf8');
      echo json_encode($response);
    }
  }
}
?>
```

webservicescripts/deleteaccount.php

```
<?php
//Require the connection to the database to happen
require('connection.php');
if (!empty($_POST)) {
  //Check to see if the user's account exists
  $query = "SELECT * FROM useraccount WHERE userid = :userid";
  //Update what :userid should be
  $query_params = array(':userid' => $_POST['userid']);
  //Run the query
  try {
    $statement = $db->prepare($query);
    $result = $statement->execute($query_params);
  }
  catch (PDOException $ex) {
    //Create the data that will be the JSON response
    $response['success'] = 0;
    $response['message'] = "Database query error#1!";
    //Kill the script and send the JSON response to the client
    header('Content-Type: application/json; charset=utf8');
    die(json_encode($response));
  }
  //Fetch the array of returned data and check if any rows were returned
  $row = $statement->fetch();
  if ($row) {
    //Check if the user has any messages in the usertextmessage table
    $query = "SELECT * FROM usertextmessage WHERE userid = :userid";
    //Update what :userid should be
    $query_params = array(':userid' => $_POST['userid']);
    //Run the query
    try {
      $statement = $db->prepare($query);
      $result = $statement->execute($query_params);
    }
    catch (PDOException $ex) {
```

```
  //Create the data that will be the JSON response
  $response['success'] = 0;
  $response['message'] = "Database query error#1!";
  //Kill the script and send the JSON response to the client
  header('Content-Type: application/json; charset=utf8');
  die(json_encode($response));
}
//Fetch the array of returned data and check if any rows were returned
$row = $statement->fetch();
if ($row) {
  //Delete all the messages in the textmessagetable based on the userid
  $query = "DELETE FROM usertextmessage WHERE userid = :userid";
  //Update what :userid should be
  $query_params = array(':userid' => $_POST['userid']);
  //Run the query
  try {
    $statement = $db->prepare($query);
    $result = $statement->execute($query_params);
  }
  catch (PDOException $ex) {
    //Create the data that will be the JSON response
    $response['success'] = 0;
    $response['message'] = "Database query error#1!";
    //Kill the script and send the JSON response to the client
    header('Content-Type: application/json; charset=utf8');
    die(json_encode($response));
  }
}
//Check if the user has any workouts in the userworkout table
$query = "SELECT * FROM userworkout WHERE userid = :userid";
//Update what :userid should be
$query_params = array(':userid' => $_POST['userid']);
//Run the query
try {
  $statement = $db->prepare($query);
  $result = $statement->execute($query_params);
}
catch (PDOException $ex) {
  //Create the data that will be the JSON response
  $response['success'] = 0;
  $response['message'] = "Database query error#1!";
  //Kill the script and send the JSON response to the client
  header('Content-Type: application/json; charset=utf8');
  die(json_encode($response));
}
//Fetch the array of returned data and check if any rows were returned
$row = $statement->fetch();
if ($row) {
  //Delete all the workouts in the userworkout based on the userid
  $query = "DELETE FROM userworkout WHERE userid = :userid";
```

```
  //Update what :userid should be
  $query_params = array(':userid' => $_POST['userid']);
  //Run the query
  try {
    $statement = $db->prepare($query);
    $result = $statement->execute($query_params);
  }
  catch (PDOException $ex) {
    //Create the data that will be the JSON response
    $response['success'] = 0;
    $response['message'] = "Database query error#1!";
    //Kill the script and send the JSON response to the client
    header('Content-Type: application/json; charset=utf8');
    die(json_encode($response));
  }
}
//Check if the user has any fitness stats in the userfitnessstatus table
$query = "SELECT * FROM userfitnessstatus WHERE userid = :userid";
//Update what :userid should be
$query_params = array(':userid' => $_POST['userid']);
//Run the query
try {
  $statement = $db->prepare($query);
  $result = $statement->execute($query_params);
}
catch (PDOException $ex) {
  //Create the data that will be the JSON response
  $response['success'] = 0;
  $response['message'] = "Database query error#1!";
  //Kill the script and send the JSON response to the client
  header('Content-Type: application/json; charset=utf8');
  die(json_encode($response));
}
//Fetch the array of returned data and check if any rows were returned
$row = $statement->fetch();
if ($row) {
  //Delete all the fitness stats in the userfitnessstatus based on the userid
  $query = "DELETE FROM userfitnessstatus WHERE userid = :userid";
  //Update what :userid should be
  $query_params = array(':userid' => $_POST['userid']);
  //Run the query
  try {
    $statement = $db->prepare($query);
    $result = $statement->execute($query_params);
  }
  catch (PDOException $ex) {
    //Create the data that will be the JSON response
    $response['success'] = 0;
    $response['message'] = "Database query error#1!";
    //Kill the script and send the JSON response to the client
```

```
      header('Content-Type: application/json; charset=utf8');
      die(json_encode($response));
    }
  }
  //Delete the user's account from the database
  $query = "DELETE FROM useraccount WHERE userid = :userid";
  //Update what :userid should be
  $query_params = array(':userid' => $_POST['userid']);
  //Run the query
  try {
    $statement = $db->prepare($query);
    $result = $statement->execute($query_params);
  }
  catch (PDOException $ex) {
    //Create the data that will be the JSON response
    $response['success'] = 0;
    $response['message'] = "Database query error#1!";
    //Kill the script and send the JSON response to the client
    header('Content-Type: application/json; charset=utf8');
    die(json_encode($response));
  }
  //Create the data that will be the JSON response
  $response["success"] = 1;
  $response["message"] = "Account deleted successfully!";
  //Send the JSON response to the client
  header('Content-Type: application/json; charset=utf8');
  echo json_encode($response);
 }
 else {
    //Create the data that will be the JSON response
    $response["success"] = 0;
    $response["message"] = "User's account does not exist!";
    //Send the JSON response to the client
    header('Content-Type: application/json; charset=utf8');
    echo json_encode($response);
 }
}
?>
```

webservicescripts/deletemessage.php

```
<?php
//Require the connection to the database to happen
require('connection.php');
//Check if the posted data is not empty
if (!empty($_POST)) {
 //Prepare the query to remove the speciic message from the database based on
 //the messageid
 $query = "DELETE FROM usertextmessage WHERE messageid = :messageid";
 //Update what :messageid should be
 $query_params = array(':messageid' => $_POST['messageid']);
 //Run the query
```

```
  try {
    $statement = $db->prepare($query);
    $result = $statement->execute($query_params);
  }
  catch (PDOException $ex) {
    //Create the data that will be the JSON response
    $response['success'] = 0;
    $response['message'] = "Database query error#1!";
    //Kill the script and send the JSON response to the client
    header('Content-Type: application/json; charset=utf8');
    die(json_encode($response));
  }
  //Create the data that will be the JSON response
  $response["success"] = 1;
  $response["message"] = "Message deleted successfully!";
  //Send the JSON response to the client
  header('Content-Type: application/json; charset=utf8');
  echo json_encode($response);
}
?>
```

webservicescripts/getuserid.php

```
<?php
//Require the connection to the database to happen
require('connection.php');

//Check if the posted data is not empty
if (!empty($_POST)) {
  //Get the user's info based on their username
  $query = "SELECT * FROM useraccount WHERE username = :username";
  //Update what :username should be
  $query_params = array(':username' => $_POST['username']);
  //Run the query
  try {
    $statement = $db->prepare($query);
    $result = $statement->execute($query_params);
  }
  catch (PDOException $ex) {
    //Create the data that will be the JSON response
    $response = array('success' => 0,
                      'message' => "Database query error#1");
    //Kill the script and send the JSON response to the client
    header('Content-Type: application/json; charset=utf8');
    die(json_encode($response));
  }
  //Fetch the array of returned data and check if any rows were returned
  $row = $statement->fetch();
  if ($row) {
    //Create the data that will be the JSON response
    $response["success"] = 1;
    $response["message"] = "UserID found!";
```

```
    $response["userid"] = $row["userid"];
    $response["userrole"] = $row["reasonforuse"];
    $response["trainerid"] = $row['trainerid'];
    //Send the JSON response to the client
    header('Content-Type: application/json; charset=utf8');
    echo json_encode($response);
  }
}
 ?>
```