МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2022 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

МЕТОДИ ЕФЕКТИВНОГО КОНТРОЛЮ ІНДИВІДУАЛЬНОГО ГРАФІКУ ЗАНЯТЬ СПОРТОМ З ІГРОВОЮ СИСТЕМОЮ МОТИВАЦІЇ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810313

Виконав студент 4-го курсу, групи 402
_____О. А. Костін
«13» червня 2022 р.

Керівник: старший викладач
_____І. С. Бурлаченко
«13» червня 2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____» _____20____р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

Видано студенту групи 402 факультету комп'ютерних наук Костіну Олександру Анатолійовичу

1. Тема кваліфікаційної роботи «Методи ефективного контролю індивідуального графіку занять спортом з ігровою системою мотивації».
Керівник роботи Бурлаченко Іван Сергійович, старший викладач.
Затв. наказом Ректора ЧНУ ім. Петра Могили від «07» грудня 2021 р. № 318
2. Строк представлення кваліфікаційної роботи студентом «__» _____ 20__ р.
3. Вхідні (початкові) дані до роботи: ім'я, вік та фізична підготовка користувача. Надання даних щодо виконання вправ користувачем.
Очікуваний результат роботи: система ефективного контролю індивідуального графіку занять спортом з ігровою системою мотивації
4. Перелік питань, що підлягають розробці (зміст пояснювальної записки)

– створити ефективну систему, яка буде відслідковувати виконання вправ з індивідуальним графіком;

– створювати ігрову систему мотивування спортсмена на продовження виконання тренувань;

– формалізувати методологію ефективного контролю індивідуального графіку занять спортом;

– розробити мобільний застосунок для тренувань спортсменів;

5. Перелік графічних матеріалів: сторінок 89, таблиць 1, рисунків 36, додатків 0, джерел 26.

6. Завдання до спеціальної частини «Вимоги до роботи із комп'ютерними пристроями»

7. Консультанти:

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алєксєєва Ганна Олександрівна, старший викладач	

Керівник роботи старший викладач, Бурлаченко Іван Сергійович

(наук. ступінь, вчене звання, прізвище та ініціали)

_____

(підпис)

Завдання прийнято до виконання Костін О.А.

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання «23» листопада 2021 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Методи ефективного контролю індивідуального графіку занять спортом з ігровою системою мотивації

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми БКР	05.11.2021	17.11.2021	Виконано
2	Отримання завдання на виконання БКР	25.11.2021	25.11.2021	Виконано
3	Складання календарного плану роботи	10.12.2021	21.12.2021	Виконано
4	Початок виконання БКР: збір та аналіз матеріалів до БКР	28.03.2022	25.04.2022	Виконано
5	Отримання завдання на переддипломну практику	23.05.2022	05.05.2022	Виконано
6	Розробка програмного забезпечення для БКР	25.05.2022	03.06.2022	Виконано
7	Попередній захист БКР на засіданні комісії кафедри	30.05.2022	30.05.2022	Виконано
8	Доробка та остаточне оформлення БКР	02.06.2022	20.06.2022	Виконано
9	Подання БКР рецензенту	16.06.2022	18.06.2022	Виконано
10	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	21.06.2022	23.06.2022	Виконано
11	Захист БКР перед екзаменаційною комісією	28.06.2022	28.06.2022	Виконано

Розробив студент Костін О. А._____

(прізвище та ініціали) (*підпис*)

Керівник роботи старший викладач, Бурлаченко Іван Сергійович_____

(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«____» _____ 2021 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи
студента групи 402 ЧНУ ім. Петра Могили
Костіна Олександра Анатолійовича
«Методи ефективного контролю індивідуального графіку занять спортом з ігровою системою мотивації»

Об’єктом дослідження даної роботи є контроль індивідуального графіку занять спортом.

Предметом дослідження є технології ефективного контролю тренувань з використанням нейронних мереж.

Метою роботи є розробка методів ефективного контролю індивідуального графіку занять спортом з ігровою системою мотивації.

Робота складається з фахового розділу і спеціальної частини з охорони праці. Пояснювальна записка складається зі вступу, п’яти розділів та висновків.

У першому розділі був проведений аналіз існуючих технологій тренувань та наявних аналогів застосунків для контролю спортивних занять.

У другому розділі було здійснено огляд математичних моделей спортивних занять. Були розглянуті методи фізіологічних вправ, відстеження позицій тіла людини та алгоритми створення індивідуального графіку занять.

У третьому розділі зроблено моделювання програми, розроблені прототипи і дизайн. Також зроблена архітектура програмного забезпечення.

У четвертому розділі були обрані технології для розробки, описано який був процес розробки та створено керівництво користувача.

Бакалаврська кваліфікаційна робота містить: сторінок 89, таблиць 1, рисунків 36, додатків 0, джерел 26.

Ключові слова: *Нейронні мережі, Відстеження рухів, Класифікація поз, Фітнес застосунок, Тренер AI, Індивідуальний графік занять, Ігрова система мотивації, Геймифікація, TensorFlow, MoveNet, Kotlin, Ефективний контроль занять спортом*

ABSTRACT

of the bachelor's degree

student of group 402 in Petro Mohyla Black Sea National University

Kostin Oleksandr Anatoliyovych

«Methods of effective control of the individual schedule of sports with a game system of motivation»

The object of study of this work is to control the individual schedule of sports.

The subject of research is the technology of effective control of training using neural networks.

The aim of the work is to develop methods of effective control of the individual schedule of sports with a game system of motivation.

The work consists of a professional section and a special section on labor protection. The explanatory note consists of an introduction, five chapters and conclusions.

In the first section, an analysis of existing training technologies and existing analogues of applications for the control of sports activities was conducted.

The second section reviews mathematical models of sports. Methods of physiological exercises, tracking the positions of the human body and algorithms for creating an individual schedule were considered.

In the third section the modeling of the program is made, prototypes and design are developed. Software architecture is also made.

The fourth section selected development technologies, described the development process, and created a user guide.

The bachelor's thesis contains: pages 89, tables 1, figures 36, appendices 0, sources 26.

Keywords: *Neural Networks, Movement Tracking, Pose Classification, Fitness Application, AI Trainer, Individual Training Schedule, Motivation Game System, Gamification, TensorFlow, MoveNet, Kotlin, Effective Sports Control*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 3
ВСТУП 4
1 АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ ТЕХНОЛОГІЙ ТРЕНУВАНЬ. ОСНОВНІ ПРОБЛЕМИ ТА ЇХ ВИРІШЕННЯ 7
1.1 Існуючі технології тренувань 7
1.2 Огляд та аналіз існуючих систем контролю занять спортом. Висвітлення проблем та їх рішення 12
1.3 Вимоги до програмного забезпечення, та постановка задачі 17
Висновки до розділу 1 17
2 МАТЕМАТИЧНІ МОДЕЛІ СПОРТИВНИХ ЗАНЯТЬ 19
2.1 Методи фізіологічних вправ 19
2.2 Відстеження позицій тіла людини 23
2.3 Алгоритми створення індивідуального графіку занять 31
Висновки до розділу 2 34
3 МОДЕЛЮВАННЯ ТА ТЕХНІЧНЕ ПРОЄКТУВАННЯ ЗАСТОСУНКУ 36
3.1 Прототипування та дизайн програмного забезпечення 36
3.2 Архітектура програмного забезпечення 43
Висновки до розділу 3 48
4 ПРОГРАМНА РЕАЛІЗАЦІЯ 49
4.1 Обґрунтування та вибір технологій розробки ПЗ 49
4.2 Опис програмної реалізації 52
4.3 Керівництво користувача 59
Висновки до розділу 4 64
5 ОХОРОНА ПРАЦІ 66
5.1 Шкідливі фактори роботи за комп’ютером 66
5.2 Вимоги до приміщення з використанням екранних пристроїв 68
5.3 Вимоги до роботи із комп’ютерними пристроями 69
5.4 Виробничий шум та вібрації 71
5.5 Режим праці та відпочинку на підприємствах з комп’ютерними пристроями 74
Висновки до розділу 5 76
ВИСНОВКИ 78
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 79
ДОДАТОК А Код Програмного Застосунку 82

ПЕРЕЛІК СКОРОЧЕНЬ

БД	–	База даних
БЖ	–	Блок живлення
ВДТ	–	Візуальний дисплейний термінал
ДК	–	Державний класифікатор
ДСанПіН	–	Державні Санітарні Правила і Норми
ДСН	–	Державні Санітарні Норми
ЕОМ	–	Електронна обчислювальна машина
ЗІЗ	–	Засоби індивідуального захисту
ІТ	–	Інформаційні Технології
КПО	–	Коефіцієнт природної освітленості
НМ	–	Нейронна мережа
ОС	–	Операційна Система
ПЕОМ	–	Персональні електронні обчислювальні машини
ПЗ	–	Програмне забезпечення
ПК	–	Персональний комп’ютер
ПК	–	Персональний комп’ютер
ССБП	–	Система стандартів безпеки праці
COCO	–	Common Objects in Context
FPS	–	Frame Per Second
UML	–	Unified Modeling Language

Пояснювальна записка

до кваліфікаційної роботи

на тему:

МЕТОДИ ЕФЕКТИВНОГО КОНТРОЛЮ ІНДИВІДУАЛЬНОГО ГРАФІКУ ЗАНЯТЬ СПОРТОМ З ІГРОВОЮ СИСТЕМОЮ МОТИВАЦІЇ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810313

Виконав студент 4-го курсу, групи 402
_____*О. А. Костін*____
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Керівник: ___*старший викладач*_____
(наук. ступінь, вчене звання)
_____*І. С. Бурлаченко*_
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Миколаїв – 2022

ВСТУП

На початку 2020 року всіх зустріло тяжке випробування у вигляді пандемії COVID-19. Це сприяло тому, що більшість людей сиділи вдома, і це погано відображалось на їх самопочутті та фізичному стані.

Тому назріла необхідність пошуку нових ефективних шляхів та засобів удосконалення та контролю тренувань та фізичного навантаження людей. Потрібні технології, які б сприяли розвитку позитивної мотивації при заняттях фізичним вихованням, формували основи самостійної оздоровчої діяльності та підтримували у людей мотивацію під час занять навіть у складні часи. Система фізичної культури повинна сприяти утвердженню здорового способу життя та прививати любов до спорту.

На початку 2022 року в країні трапилась війна. По всій території впроваджене воєнне положення і кожного дня існує небезпека життю. У зв’язку зі стресовою ситуацією організм виділяє адреналін та кортизол. Ці гормони негативно впливають на здоров’я у довгостроковій перспективі. І зараз кожен українець повинен піклуватись про своє здоров’я, щоб бути сильним як духом так і тілом, щоб мати можливість захищати свою країну та відновити її після перемоги.

Також відомо, що для того щоб перебороти стрес та депресію під час таких незрозумілих часів, потрібно щоб в житті біло щось стабільне, та мати успіхи і контроль хоча б над чимось маленьким у житті.

Для цього, в рамках даної кваліфікаційної роботи було вирішено розробити технологію та методи ефективного контролю індивідуального графіку занять спортом. Та щоб покращити мотивацію додати до неї ігрову систему мотивації. Таким чином люди будуть піклуватись про своє здоров’я, мати успіхи та чітко їх відслідковувати. Досягати цілей і покращувати своє самопочуття.

Актуальність – система контролю виконання фізичних вправ та створення індивідуального графіку тренувань необхідна людям для підтримки свого здоров’я та морального стану.

Завдання які мають бути вирішеними:

– створити ефективну систему, яка буде відслідковувати виконання вправ з індивідуальним графіком;

– створювати ігрову систему мотивування спортсмена на продовження виконання тренувань;

– формалізувати методологію ефективного контролю індивідуального графіку занять спортом;

– розробити мобільний застосунок для тренувань спортсменів;

Об’єкт дослідження – контроль індивідуального графіку занять спортом

Предмет дослідження – технології ефективного контролю тренувань з використанням нейронних мереж

Мета роботи – Розробка методів ефективного контролю індивідуального графіку занять спортом з ігровою системою мотивації

Методи що мають бути використані – мова програмування Kotlin для розробки основного застосунку, та TensorFlow для роботи з нейронними мережами.

1 АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ ТЕХНОЛОГІЙ ТРЕНУВАНЬ. ОСНОВНІ ПРОБЛЕМИ ТА ЇХ ВИРІШЕННЯ

1.1 Існуючі технології тренувань

Спортивне тренування – детальний і структурований процес виховання, тренування та підвищення фізичної підготовки спортсменів в умовах відповідного гігієнічного режиму, на основі педагогічного та лікарського контролю, а також самоконтролю [1].

Засоби та технології спортивного тренування поділяються на такі категорії [1]:

– загально-підготовчі;

– спеціально-підготовчі;

– спеціальні вправи відносно обраного виду спорту.

Загально-підготовчі вправи. Методи, що служать комплексному функціональному розвитку організму людини. Засобами і методами загальної фізичної підготовки людини повинні бути [1]:

– універсальними, що у поєднанні зі спеціальними вправами дозволяють повноцінно розвинути фізичні навички;

– відражаючими особливості спортивної дисципліни та забезпечувати позитивну передачу тренувальних і рухових навичок.

Спеціально-підготовчі вправи включають в собі елементи змагальних дій, їх варіанти, а також ті дії, які мають значні схожості з обраним видом спорту у формі та способі демонстрації майстерності. Спеціальні підготовчі вправи, у свою чергу, поділяються на *підвідні*, *імітаційні* та *підготовчі* вправи [1].

– *Підвідні* вправи допомагають оволодіти формою і технікою рухів.

– *Імітаційні* вправи відповідають координаційно-кінематичній структурі обраного виду спорту.

– *Підготовчі* вправи направлені на розвиток спеціальних рухових навичок.

Спеціальні вправи відносно обраного виду спорту. Цілісні рухові дії або їх комбінації, Які здійснюються за правилами обраного виду спортивного бою.

Спортивні методи тренування. Методи спортивних тренувань поділяються на *загально-педагогічні*, *специфічні* та *додаткові*, які спеціально пристосовані до потреб спортивної практики.

Загально-педагогічні методи спрямовані на здобуття знань. Вони поділяються на методи, які забезпечують:

– сенсорну передачу – засвоєння інформації через фізичний показ або демонстрацію;

– друковану передачу – запис і відтворення інформації шляхом роботи з документами;

– усну передачу – запис і відтворення інформації. Прикладом є лекція, розповідь, пояснення тощо.

За допомогою спеціальних методів ви можете оволодіти руховими навичками та вміннями, покращити ці навички та розвити фізичну форму [2].

Для оволодіння руховими навичками використовуйте методи, що передбачають формування цілісної моторики, а також методи, що передбачають формування часткових цільних рухів з подальшим їх об’єднанням в ціле.

Для вдосконалення рухових навичок і розвитку фізичної підготовленості використовуються методи, що дозволяють точно нормувати і регулювати навантаження які виникають під час тренування: рівномірний метод, перемінний метод, повторний метод, інтервальний метод, метод колового тренування [1].

З цією ж метою використовуються ігрові та змагальні методи, які забезпечують створення ігрових та змагальних ситуацій у процесі виконання завдання [3]. Розвитку цих форм взаємодії у сфері вправ сприяла поява нових технологій, таких як Інтернет, комп’ютери та розробка мобільних додатків на смартфонах і планшетах.

Інші методи розроблені спеціально відповідно до вимог спортивної практики [1]. Це аутогенне тренування, психомоторне тренування, тренування в екстремальних умовах тощо.

Сучасний період розвитку фітнес-індустрії характеризується великою кількістю та різноманітністю тренувальних програм, модернізацією та адаптацією оздоровчих програм для залучення більшої кількості людей за допомогою популяризації та використання реклами [2].

У індустрії існує більше ста програм на основі гімнастики. Класифікація цих програм має певні труднощі через їх велику кількість, різне призначення та мету [4].

Застосунки для фітнесу є однією з найпопулярніших форм контролювання здоров’я та спорту сьогодні [3]. Ці програми пропонують користувачам спосіб покращити свій фізичний стан, займаючись на індивідуальній основі за допомогою програмного забезпечення, яке допомагає їм відстежувати свій прогрес і за потреби вносити поступові коригування [5]. Враховуючи той факт, що рівень ожиріння продовжує зростати за останні роки, потреба в комп’ютерних фітнес-програмах існую досить велика.

Оскільки концепція фітнесу складається з багатьох компонентів, кількість фітнес-програм, які можна створити, майже безмежна [5].

Різноманітність фітнес-програм не означає, що їх можна робити випадковим чином – використання різних видів рухів має відповідати основним принципам фітнесу. Будь-яка фітнес-програма повинна сприяти розвитку всіх частин тіла [3].

Структура фітнес-програми може змінюватися в залежності від мети заняття, фізичного стану учасників програми та інших факторів. Але незалежно від того, наскільки новою є фітнес-програма, її структура повинна включати такі частини, як розминка, тренування та заминка [1].

Розробка фітнес-програм є одним з основних джерел доходу для сучасної фітнес-індустрії. Створення нової фітнес-програми супроводжується

продуманою маркетинговою політикою, яка передбачає не лише публікацію посібників та рекомендацій, а й широку рекламу вживаного обладнання, випуск аудіо- та відеопродукції [1], а також освітні семінари та курси підготовки медичних фітнес інструкторів.

Розробляючи фітнес-програму, необхідно враховувати цілі людини, початковий рівень підготовки, вік, фізичну підготовку, рухові навички, інтереси. Важливим фактором є розробка фітнес-програми для забезпечення правильного рівня фізичної активності для досягнення максимальної користі з мінімальним ризиком [2].

Розпочинаючи програму вправ, важливо починати повільно. Це не тільки допоможе запобігти травмам і перетренованості, але також допоможе вам продовжувати виконувати встановлену програму фітнесу. Відомо, що 60% людей припиняють тренуватися вже після першого місяця.

Особливу увагу слід приділити необхідності дотримуватися фітнес-програми і зробити її частиною свого життя [2]. Для цього найефективнішим є встановлення фітнес-цілей, які мають бути:

– конкретними;
– реалістичними;
– містять результат і завдання.

Частою причиною фізичних вправ є бажання схуднути або покращити фізичну форму. Щоб збільшити свої шанси на успіх, потрібно бути більш конкретним. Якщо мета – схуднути, то потрібно визначити, на скільки кілограмів потрібно схуднути і за який період часу.

На додаток до конкретності, фітнес-цілі мають бути легкими, але досягнутими. Якщо фітнес-цілі вимагають від людини екстраординарних змін у поведінці, шанси на успіх будуть значно нижчими [1].

Незважаючи на переваги фізичних вправ, багатьом людям все одно не вистачає мотивації для того щоб інтегрувати їх у своє повсякденне життя. Адже вони вважають, що такі заняття менш приємні та втомливі порівняно з сидячим

способом життя. Останнім часом спостерігається зростання інтересу до використання ігрових принципів у звичайних та нецікавих ситуаціях [6]. Це потрібно для того, щоб зробити діяльність, яка сприймається складною чи нудною, приємнішою.

Очікується, що зі збільшенням задоволення за рахунок додавання ігрових елементів люди будуть більш мотивовані брати участь у діяльності. Дослідження показали [5], що геймифікація покращує не лише ставлення до вправ та задоволення від них, а й формує поведінку з погляду збільшення фізичної активності. Це є хорошим знаком для платформ в яких є елементи з ігрових механік та його корисності для мотивації фізичних вправ серед людей.

В іграх часто використовуються нагороди. Основною метою включення винагород є мотивація зусиль користувачів, що відображається в балах і значках. Бали надають користувачам форму зворотного зв'язку про їхню роботу з погляду зусиль та інтенсивності, вкладених у вправу. Наприклад, вони отримують більше балів, тому що витрачають більше часу на фізичні вправи, збільшують кількість повторень або докладають більше зусиль у своїх тренуваннях. Значки - це ще одна форма винагороди, яка є підтвердженням статусу, джерелом репутації та досягненням мети.

Вважається також, що якщо докладати великих зусиль, особливо фізично, то задоволення від винагороди сприймається ближче [5]. Це також накладається на те, що під час тренування ендорфіни вивільняються для поліпшення настрою людини. Як наслідок, у сумі задоволення може змінити негативний вплив фактичного зусилля, що додається для виконання вправи. Крім того, це може спонукати людину вкладати більше енергії в поточну діяльність. Водночас це веде до наміру робити більше в майбутньому. У довгостроковій перспективі це приносить психологічні переваги на додачу до фізіологічних.

Вже зрозуміло, що для того щоб досягнути цікавого ігрового процесу потрібно використати винагороди у вигляді значків чи балів. Окрім цього

потрібно врахувати, що у гарній ігровій системі повинен бути "основний цикл геймплею" [5] або OCR (Objective, Challenge & Reward).

– Завдання: дати зрозумілу та певну мету для гравця;
– Випробування: створити веселий та цікавий спосіб досягнення мети;
– Нагорода: заохотити до посилення залучення.

Також потрібно врахувати три рівня мотивації користувача:

1) **Короткострокова мотивація** – те, що спонукає гравця подолати поточне випробування. У випадку з тренуваннями це може бути завершення даного підходу. Наприклад, зробити 10 присідань. Далі за це повинна йти винагорода.

2) **Середньострокова мотивація** – те, що спонукає гравця завершити поточний розділ, рівень чи місію. У випадку з тренуваннями це може бути завершення комплексу вправ на сьогодні, або на неділю.

3) **Довгострокова мотивація** – це те, що спонукає користувача завершити гру. Потрібна конкретна ціль до якої можна дійти. У випадку з фітнес застосунком це може бути завершення всієї програми тренувань, скидання ваги тіла до цільового значення, або збільшення кількості виконання вправ користувачем. Що і буде означати завершення «гри».

1.2 Огляд та аналіз існуючих систем контролю занять спортом. Висвітлення проблем та їх рішення

Мобільний застосунок – “BetterMe: Тренер здоров'я”.

Цей застосунок розроблений українською компанією. Створений для підтримки здоров’я користувачів. Включає в себе як тренування, так і харчування, психологічні поради, випробовування, та інше.

Плюси програми:

– гарний візуальний стиль;
– є можливість створення плану харчування;

– великий вибір тренувань. Можна обрати що більше підходить. Є програми навіть для хворих;

– є індивідуальний підхід до користувача. При реєстрації можна вказати свої цілі та свій поточний стан;

– є відстеження води та калорій;

– є психологічні лекції.

Мінуси програми:

– тренування фіксуються не за кількістю виконаних вправ, а за часом (рис. 1.1). В цілому це не дуже погано, але складніше відстежувати свій прогрес у тренуваннях;

– не контролюється правильність виконання вправ;

– мало даних, які показують користувачу його результати (рис. 1.2).

В цілому, застосунок чудовий, але за рахунок великої кількості функціоналу, йому дещо бракує спрямованості на тренуваннях.

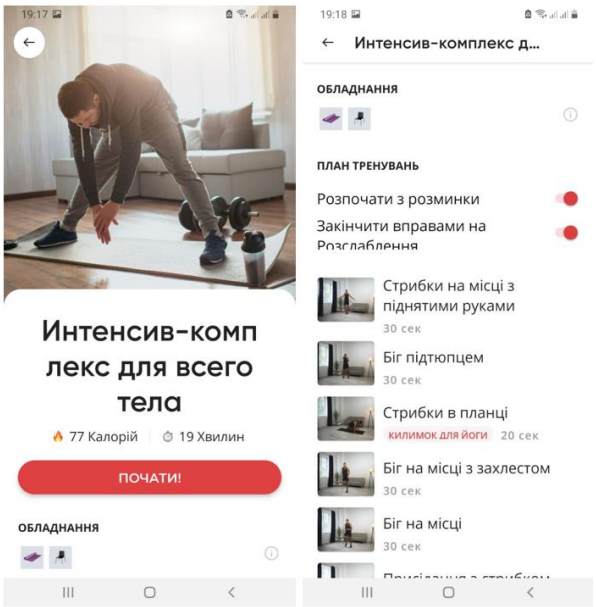

Рисунок 1.1 – Застосунок BetterMe – сторінки з вправами

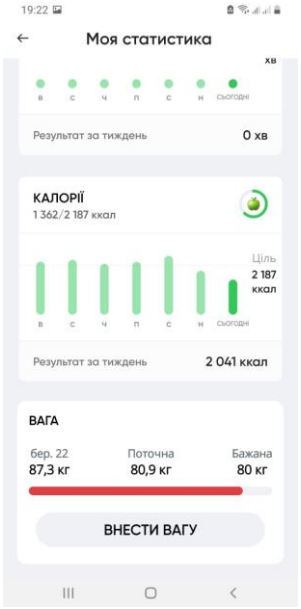

Рисунок 1.2 – Застосунок BetterMe – сторінка зі статистикою

Мобільний застосунок – “200 Віджимань: тренування вдома”

Розробники програми кажуть, що цей застосунок допоможе швидко збільшити кількість віджимань. Він має унікальний алгоритм, що на основі результатів створює індивідуальний план тренувань.

Плюси програми:

– зрозуміла програма тренувань з чіткою ціллю;

– можна спостерігати прогрес виконання завдань, що додає мотивації (рис. 1.5);

– програма підлаштовується індивідуально під користувача (рис. 1.4). Кожної неділі провіряє прогрес

Мінуси програми:

– не відстежується виконання вправ;

– спеціалізація лише на віджиманнях;

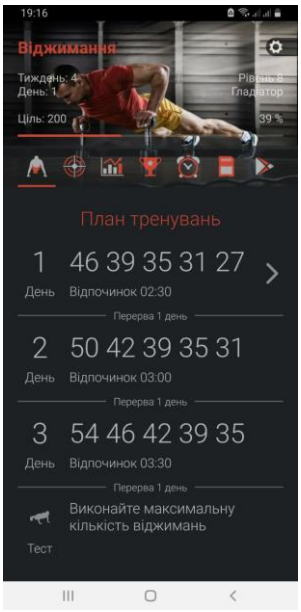

Рисунок 1.3 – Застосунок 200 Віджимань – Сторінка за планом тренувань

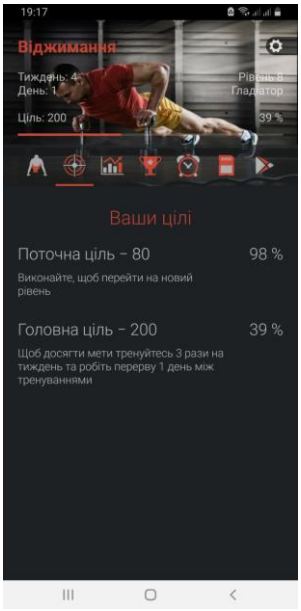

Рисунок 1.4 – Застосунок 200 Віджимань – Сторінка з цілями

Мобільний застосунок – “EZPT – AI Movement Trainer”

EZPT — це тренер з рухів зі штучним інтелектом, фізіотерапевт та персональний тренер, який аналізує вашу форму під час вправ і дає вам звуковий зворотний зв’язок у реальному часі.

Плюси програми:

– наявність штучного інтелекту, який відстежує і коригує вправи (рис. 1.5);

– підбір кількості підходів у вправах;

Мінуси програми:

– хоч застосунок підбирає кількість вправ, це важко назвати програмою тренувань. Не зрозуміло в якій послідовності і що робити (рис. 1.6);

– відсутні цілі і, як результат, мотивація;

– відсутній індивідуальний підхід до користувача;

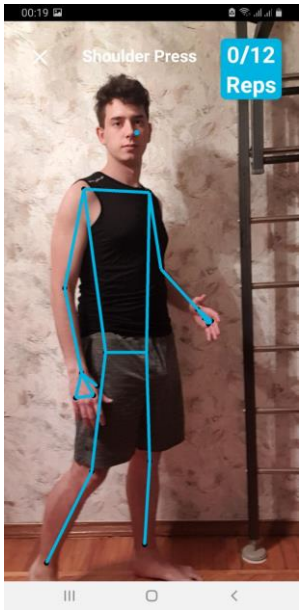

Рисунок 1.5 – Застосунок EZPT – сторінка з відстеженням рухів

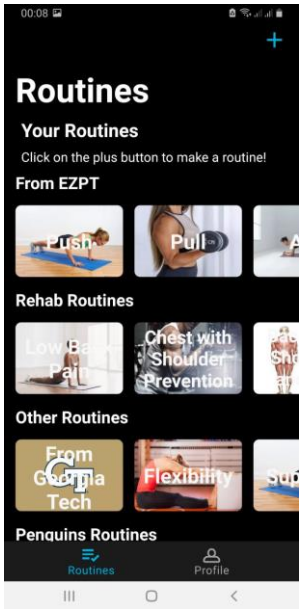

Рисунок 1.6 – Застосунок EZPT – сторінка з тренуваннями

Застосунок гарний тим, що відстежує дії користувача. Але окрім цього у нього багато мінусів, таких як, відсутність мотивації та індивідуального підходу.

В застосунку використовується корисна технологія відстеження тіла. За допомогою неї можливо зробити ефективний контроль вправ у тренуваннях [7].

1.3 Вимоги до програмного забезпечення, та постановка задачі

Враховуючи все вищесказане, можна описати основні цілі та **вимоги** майбутнього застосунку.

1) Застосунок повинен відстежувати та контролювати виконання вправ;

2) Застосунок повинен індивідуально створювати програму тренувань, на основі можливостей користувача;

3) Користувач повинен бути замотивований. Для цього можна використати елементи з гейм дизайну;

4) Необхідно відстежувати прогрес та успіхи користувача;

5) Застосунок повинен бути мобільним, на базі операційної системи android чи ios, щоб ним можна було користуватись де завгодно.

Отже, згідно з вимогами можна поставити наступну **задачу**: Розробити android-застосунок, який буде створювати індивідуальний графік тренувань, та відстежувати виконання вправ методом штучного інтелекту. Також для додаткової мотивації користувача та включення його у процес додати елементи з ігрових практик.

Висновки до розділу 1

У першому розділі проведений аналіз існуючих технологій тренувань та засобів мотивації користувача. Як потрібно створювати програми тренувань, щоб їх дотримувались та що потрібно врахувати. Розглянуто які бувають типи фізичних тренувань. Які існують методи поступового навантаження спортсмену. Розглянуто які бувають програми тренувань, та їх актуальність у індустрії фітнесу. Зрозуміло яких цілей хочуть досягти люди купуючи програми

тренувань. Розглянуто як саме розробляти програму та що слід обов'язково врахувати при їх розробці. Розглянуто що таке ігрова мотивація та геймифікація.

Було розглянуті існуючі варіанти розроблених застосунків, які також пропонують графіки занять спортом. Конкретно розглянуто три застосунки - "BetterMe: Тренер здоров'я", "EZPT – AI Movement Trainer" та "200 Віджимань: тренування вдома". Після розгляду програм стало зрозуміло, що в кожного є якісь плюси та мінуси.

На основі проведеного аналізу було створено вимоги та чітке завдання щодо власного застосунку. Він повинен відслідковувати виконання індивідуального графіку тренувань, та мати елементи гри, щоб мотивувати користувача.

2 МАТЕМАТИЧНІ МОДЕЛІ СПОРТИВНИХ ЗАНЯТЬ

2.1 Методи фізіологічних вправ

Неправильно виконані вправи можуть призвести до травм, які можуть затримати або перешкодити проведенню тренування. Також травми можуть призвести до зниження продуктивності, що може негативно вплинути на цілі спортсмена. Для досягнення оптимальних результатів важливо правильно виконувати вправи як на початковому етапі тренувань, так і протягом усього тренувального процесу [8]. Також знання про правильне виконання вправ допоможе у створенні алгоритму відстеження правильності виконання вправ.

В рамках БКР буде розглянуто декілька основних, всім знайомих вправ:

1) віджимання;
2) присідання;
3) підйом ніг лежачи;

Віджимання

Віджимання є важливою вправою для тих, хто хоче покращити свою статуру та загальну форму. Вони опрацьовують все тіло, включаючи м’язи грудей, плечей, рук і спини. Віджимання можна виконувати де завгодно – вдома чи в спортзалі, тому вони ідеально підходять для будь-якого графіка і стилю життя. Та віджимання корисні не тільки для вашої зовнішності: віджимання допомагають вам підтримувати міцні кістки, запобігати хворобам серця, зменшувати жир на животі, нарощувати м’язову масу тощо!

Віджимання важливі для спортсменів, оскільки вони допомагають розвивати силу, витривалість і мускулатуру, необхідні для інших видів діяльності. Вони також допомагають покращити ваш баланс, координацію та гнучкість. Жоден тренувальний процес не проходить без перевірки спортсмена на силу та витривалість. Багато спортсменів щодня роблять віджимання від підлоги, щоб підтримувати вагу і форму, коли регулярні тренування неможливі.

Цей вид діяльності ми знаємо зі школи, адже всі хлопці 5-11 класів здавали норматив на віджимання від підлоги. Однак навіть у зрілому віці мало хто правильно виконує цю вправу. Техніка виконання сильно страждає, оскільки люди зовсім не підковані теоретично.

Для правильного виконання віджимань потрібно виконувати певні правила.

1) Прийміть упор лежачи. Тіло має бути напруженим, а хребет повністю прямим. Прогину в попереку не повинно бути, таз треба тримати опущеним. Ноги трохи розставлені, долоні дивляться вперед.

2) Почніть згинати руки в ліктях і повільно опустіть тіло. У найнижчій точці відстань від підлоги до тіла не повинна перевищувати трьох сантиметрів.

3) Обов’язково зверніть увагу на своє дихання. У негативній фазі руху, коли тіло опускається – вдих, а в активній фазі, коли тіло піднімається – видих.

4) Щоб підняти корпус, випрямляйте руки поступово, але не робіть це до кінця. Виконуйте вправу в межах амплітуд і тримайте всі м’язи в стані постійної напруги.

Рисунок 2.1 – Верхнє положення при віджиманні

Рисунок 2.2 – Нижнє положення при віджиманні

Присідання

Присідання є одним із найпоширеніших рухів, які ми виконуємо протягом дня. Сильні ноги – не єдина користь, яку ви можете отримати від цієї вправи. Різні варіанти присідань використовуються в реабілітації спортсменів після травм. Ми присідаємо десятки разів на день, не усвідомлюючи цього.

Для виконання присідань правильно також потрібно виконати декілька правил.

1) Встаньте прямо з випрямленими колінами і стегнами трохи ширше за плечі. Дивіться вперед в нейтральному положенні голови. Підніміть груди вперед і вгору і зведіть плечі разом, ніби між ними щось стиснуто. Стегна розташовані рівно.

2) Тримайте тулуб у прямому положенні. Хребет повинен залишатися стабільним протягом усього руху.

3) Розгорніть ноги на 45-30 градусів. Таке положення ноги дозволить коліну рухатися в напрямку шкарпеток. Щільно притисніть п’яти до землі.

4) Згинайтеся одночасно в гомілки, колінах та тазі. Коліна розташовані на одній лінії. Стежте, щоб ваші коліні не завалювалися всередину.

5) Повільно опустіться з прямою спиною до кута у 90 градусів або нижче у колінному суглобі.

6) З нижньої точки присідання потужними рухом виштовхніться вгору, напружуючи спину та прес. Зупиніться у верхній точці із випрямленими ногами.

Рисунок 2.3 – Зліва – початкове положення при присіданнях.
Справа – нижня точка положення при присіданнях

Підйом ніг лежачи

У спорті одним з основних елементів тренування низу живота є підйом ніг лежачи. Це базовий рух, що дозволяє безпечно і ефективно розвивати нижній прямий м’яз пресу.

Виконання підйомів ніг лежачи на підлозі є полегшеним варіантом підняття ніг у висі. Цю вправу легко виконати вдома. Для цього не потрібно додаткового обладнання. Тому це універсальний засіб для підтримки тонусу м’язів живота в будь-якій ситуації.

1) Ляжте на спину, випрямивши ноги, не торкаючись підлоги. Витягніть руки вздовж тіла. Можете схопитися за край килимка.

2) Тримайте ноги прямими, трохи зігнутими в колінах, і підіймайте ноги під кутом 90 градусів. Видихайте під час цієї частини вправи.

3) На вдиху повільно опустіть ноги вниз у вихідне положення.

Рисунок 2.4 – Верхнє положення ніг

Рисунок 2.5 – Нижнє положення ніг

2.2 Відстеження позицій тіла людини

Оцінка пози — це завдання використання моделі ML для оцінки пози людини з зображення або відео, оцінюючи розташування ключових точок суглобів тіла у просторі [9].

Оцінка пози відноситься до технік комп’ютерного зору, які визначають людські фігури на зображеннях і відео, щоб була змога визначити, наприклад, де з’являється чийсь лікоть на зображенні. Ці моделі в основному є картою суглобів тіла, які ми відстежуємо під час руху [9]. Це робиться для того, щоб комп’ютер не тільки знаходив різницю між людиною, яка просто сидить і присідає, а й щоб розрахувати кут згинання в конкретному суглобі та визначити, чи правильно

виконано рух. Ця особливість буде використана у подальшій роботі для відстежування правильності виконання рухів

Застосунки для фітнесу та тренери з штучним інтелектом – одні з найбільш очевидних випадків використання оцінки поз тіла [10]. Реалізуючи цю модель в мобільному застосунку, можна використовувати апаратну камеру як датчик, щоб зафіксувати, як хтось виконує вправу, і одразу аналізувати її. Це здійснюється шляхом відстеження ключових точок. Це можна обробляти в режимі реального часу або після певної затримки, надаючи користувачам аналітику щодо основних рухів і механіки вправ.

Важливо пам’ятати також про те, що оцінка пози лише визначає, де знаходяться ключові суглоби тіла, а не розпізнає хто на зображенні. Моделі оцінки пози беруть оброблене зображення камери як вхід і виводять інформацію про ключові точки. Виявлені ключові точки індексуються показником достовірності від 0,0 до 1,0. Оцінка впевненості вказує лиш на **ймовірність** існування ключової точки в цій позиції.

Розглянемо існуючі **технології відстеження сегментації тіла** (body segmentation tracking) з використанням нейронних мереж.

Основні з них це BodyPix, MoveNet, BlazePose та PoseNet. Найзручніший спосіб їх використання це використати бібліотеку TensorFlow [11].

TensorFlow — це безкоштовна бібліотека з відкритим кодом для машинного навчання та штучного інтелекту, яка розроблена командою Google Brain. TensorFlow можна використовувати в широкому діапазоні мов програмування. Ця гнучкість підходить для широкого спектру застосувань.

Розглянемо детальніше кожну з технологій відстеження тіла.

BodyPix для сегментації зображення аналізує, чи є пікселі частиною людини, чи не є її частиною. А також розрізняє пікселі, які належать кожній із частин тіла людини. Він може розрізняти кількох людей у вхідному зображенні або відео. Може виділити силует тала (рис. 2.6), та розмити чи закрасити окремі його частини.

Більше підходить для використання у відео конференціях, ніж для відслідковування позицій тіла.

Рисунок 2.6 – Приклад роботи BodyPix

Наступні технології використовуються саме для визначення позицій тіла у просторі.

BlazePose може виявити 33 ключові точки [11], на додаток до 17 ключових точок COCO, він надає додаткові ключові точки для обличчя, рук і ніг (рис. 2.7). Виходячи із додаткових точок на руках (рис. 1.5), можна дійти до висновку, що застосунок «EZPT – AI Movement Trainer» використовував саме цю технологію.

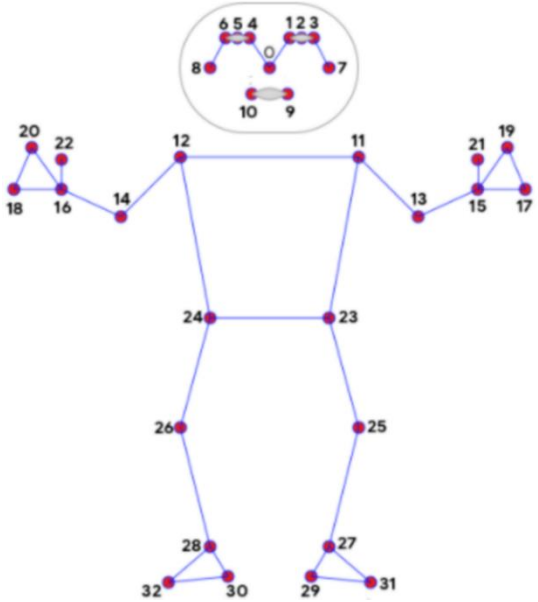

Рисунок 2.7. Точки які використовуються у BlazePose

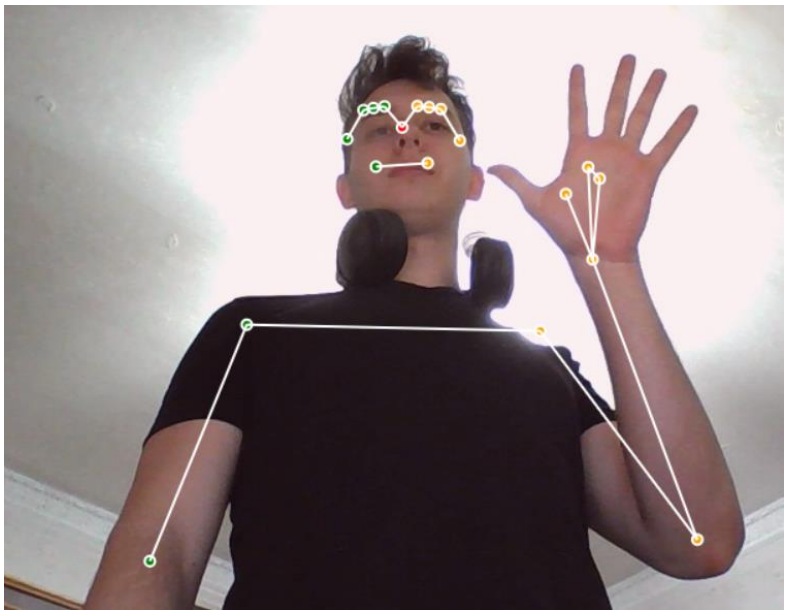

Рисунок 2.8 – Приклад роботи BlazePose. Середня частота кадру – 53 fps

Наступна технологія PoseNet. Вона може виявляти кілька поз, кожна поза містить 17 ключових точок (рис. 2.9).

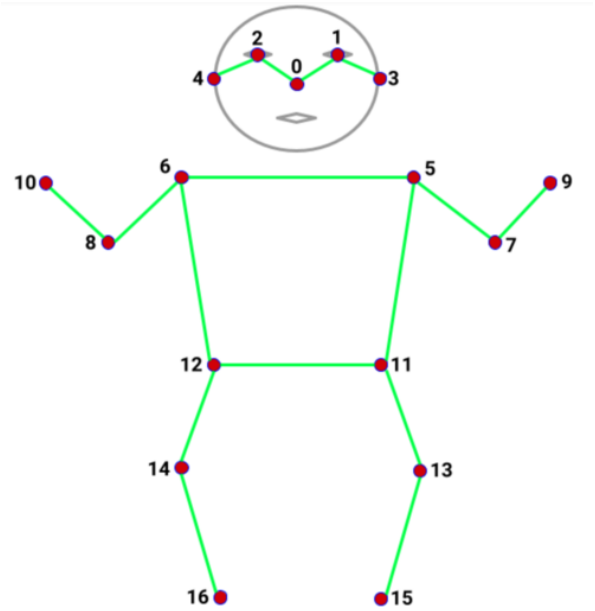

Рисунок 2.9 – Точки які використовуються у PoseNet та MoveNet

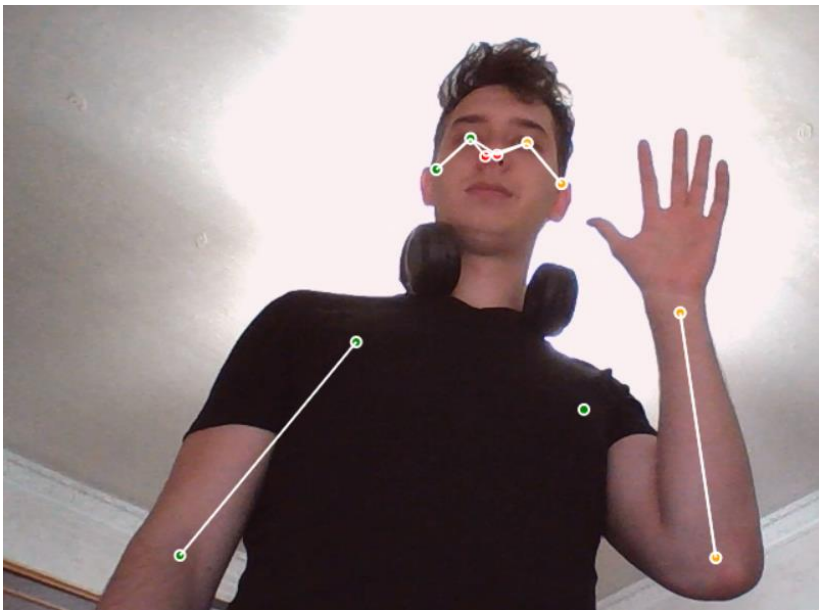

Рисунок 2.7 – Приклад роботи PoseNet. Середня частота кадру - 25 fps

Далі йде MoveNet – модель згорткової нейронної мережі, яка працює на зображеннях RGB і передбачає розташування людей у спільних місцях людей у кадрі зображення. Дуже швидка і точна модель, яка, як і PoseNet виявляє 17 ключових точок тіла (рис. 2.9) [11]. Він може працювати зі швидкістю понад 50 кадрів в секунду (рис. 2.11) на сучасних ноутбуках і телефонах.

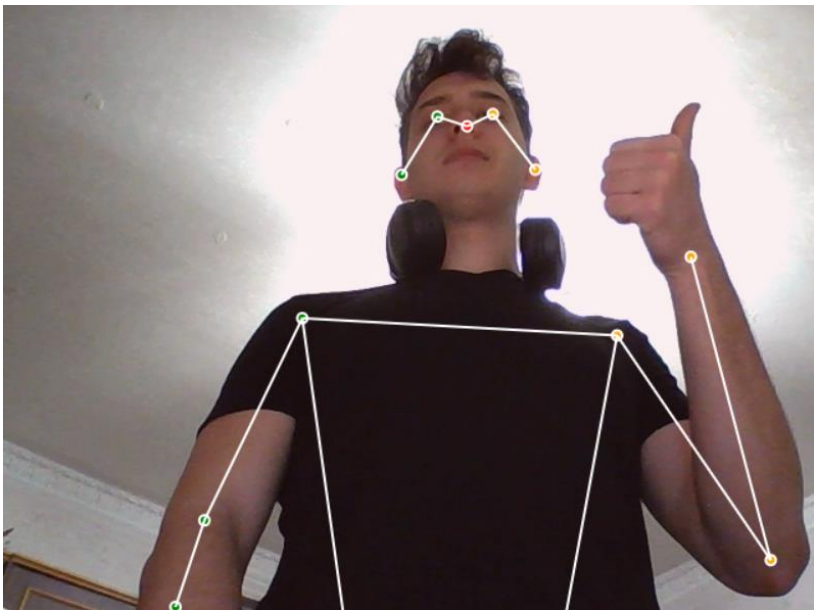

Рисунок 2.8. Приклад роботи MoveNet. Середня частота кадрів - 78 fps

Найкращим варіантом для відстеження виконання вправ буде технологія відстеження позицій тіла MoveNet. BlazePose має додаткові точки, але вони

будуть тільки заважити. А ось частота кадрів може бути вирішальною при відстежуванні виконання вправ, тому MoveNet буде найкращим вибором.

MoveNet розпізнає різні суглоби тіла. Які суглоби може виявляти модель оцінки пози наведені в табл. 2.1.

Таблиця 2.1 – Відповідність суглобів у MoveNet

Id	Part	переклад
0	nose	ніс
1	leftEye	ліве око
2	rightEye	праве око
3	leftEar	ліве вухо
4	rightEar	праве вухо
5	leftShoulder	ліве плече
6	rightShoulder	праве плече
7	leftElbow	лівий лікоть
8	rightElbow	правий лікоть
9	leftWrist	лівого зап'ястя
10	rightWrist	правого зап'ястя
11	leftHip	ліве стегно
12	rightHip	правого стегна
13	leftKnee	ліве коліно
14	rightKnee	праве коліно
15	leftAnkle	ліву щиколотку
16	rightAnkle	права щиколотка

Використовуючи ці суглоби та можливість виявляти кути між ними, була розроблена модель відстеження виконання вправ. За основу взяті вправи, розглянуті раніше: віджимання, присідання та підняття ніг.

Для відслідковування **віджимань** обрано два кути:

1) ∠α – кут між суглобами плеча, стегна та коліна (рис. 2.12)
2) ∠β – кут між суглобами зап’ястя, ліктя та плеча

Кути повинні знаходитись як для лівої, так і для правої частини тіл людини.

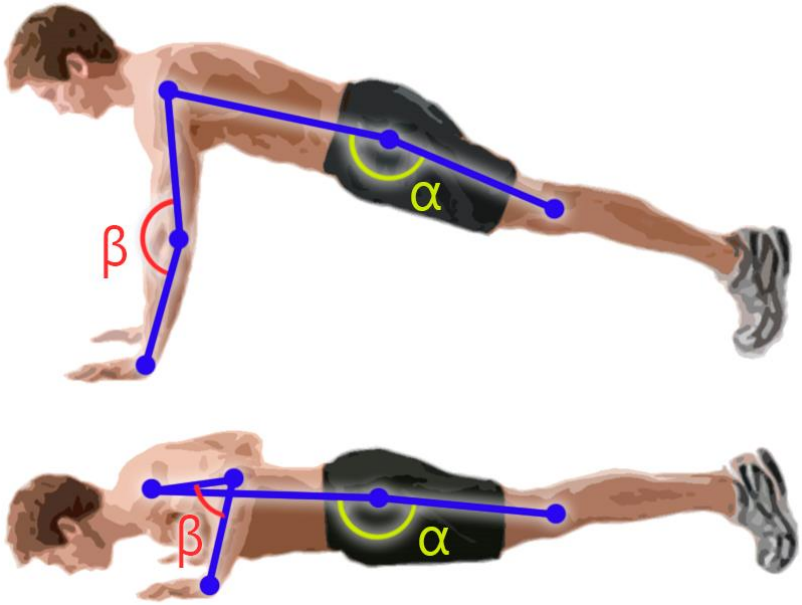

Рисунок 2.12 – Положення суглобів при віджиманні

Далі знаходяться межі кутів, в яких поза буде задовільною. Це робиться в наслідок того, що класифікація поз може бути не точним іноді. І тіло людини не може завжди бути під одним кутом, тому потрібна дозволена похибка.

Мінімальне значення для ∠α = 155°

Максимальне значення для ∠α = 185°

Це для місць, в яких кут не повинен згинатись

Для ∠β Потрібно вказати значення, перетнувши які буде визначатись які це позиція віджимань – нижня чи верхня.

Якщо ∠β > 160°, то це зараховується як верхня позиція. Якщо ∠β < 90°, то це зараховується як нижня позиція. Якщо нижня позиція закінчилась, і знов йде перехід до верхньої позиції, то вправа вважається виконаною, і лічильник збільшується на одиницю.

Присідання. В цьому випадку потрібно відслідковувати більше кутів.

Для цього обрані чотири кути (та треба враховувати ще чотири для інших сторін тіла). А саме (рис. 2.13):

1) ∠α – кут між суглобами зап’ястя, ліктя та плеча
2) ∠β – кут між суглобами ліктя, плеча та стегна
3) ∠γ – кут між суглобами плеча, стегна та коліна
4) ∠φ – кут між суглобами стегна, коліна та щиколотки

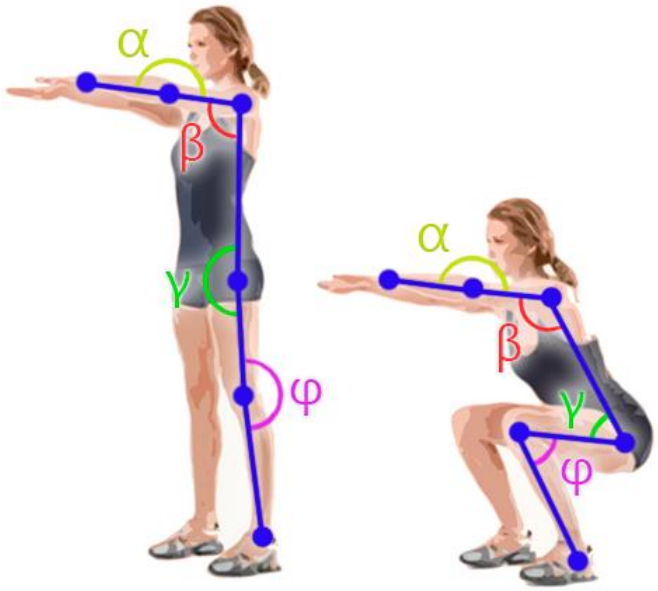

Рисунок 2.13 – Відстеження кутів при віджиманнях

Як і раніше, для кутів що повинні триматись в одному діапазоні вказано мінімальний та максимальній кут.

– Мінімальне значення для ∠α = 155°
– Максимальне значення для ∠α = 185°
– Мінімальне значення для ∠β = 85°
– Максимальне значення для ∠β = 115°

Далі, якщо ∠γ > 160° , та ∠φ > 160°, то це зараховується як позиція стоячи. Якщо ∠γ < 70° , та ∠φ < 70°, то як нижня позиція. Коли позиція переходить в нижню, то лічильник присідань збільшується на один.

Наступна вправа для відстежування – **підняття ніг**. На цей раз не потрібно відстежувати багато кутів. Достатньо Одного найважливішого:

1) ∠α – кут між суглобами плеча, стегна та коліна (рис. 2.14)

Рисунок 2.14 – Положення кутів між суглобами для підйому ніг

Якщо ∠α > 165°, то це початкова позиція. Якщо ∠α < 90°, то це активна позиція. Коли Активна позиція змінюється на початкову, то лічильник підняття ніг збільшується на один.

Таким чином у застосунку будуть відстежуватись позиції тіла, та контролюватись процес виконання вправ.

2.3 Алгоритми створення індивідуального графіку занять

Для контролю індивідуального графіку занять буде створений мобільний додаток.

Індивідуальний графік занять потрібно розробляти в залежності від цілей і потреб користувача. Також потрібно врахувати його фізичні здібності.

Як ціль взято збільшення кількості виконаних вправ з часом.

Фізичні здібності будуть визначатись під час алгоритму.

Для цього використовується простий алгоритм:

1) Визначається скільки максимально зараз може робити вправ користувач
2) В залежності від цього формуються підходи та кількість вправ. Індивідуальний тренувальний режим початкового фізичного навантаження буде складати 50–70% від максимальної кількості рухів [12]. Також потрібно

врахувати, що для вправ для нижньої частини тіла потрібно робити більше повторень [13].

3) Розробляється програма на три дні з поступовим збільшенням навантаження на 5–10% [14].

4) Робиться перевірка скільки тепер максимум може робити вправ користувач і алгоритм починається з кроку 2.

Враховуючи все вищезазначене розробимо програму тренувань **для віджимань:**

Перше тренування:

– Підхід 1 - 60% від максимуму
– Підхід 2 - 50% від максимуму
– Підхід 3 - 45% від максимуму

Друге тренування:

– Підхід 1 - 65% від максимуму
– Підхід 2 - 55% від максимуму
– Підхід 3 - 50% від максимуму

Друге тренування:

– Підхід 1 - 70% від максимуму
– Підхід 2 - 60% від максимуму
– Підхід 3 - 55% від максимуму

Для присідань:

Перше тренування:

– Підхід 1 - 70% від максимуму
– Підхід 2 - 60% від максимуму
– Підхід 3 - 50% від максимуму

Друге тренування:

– Підхід 1 - 80% від максимуму
– Підхід 2 - 70% від максимуму

– Підхід 3 - 60% від максимуму

Друге тренування:

– Підхід 1 - 90% від максимуму

– Підхід 2 - 80% від максимуму

– Підхід 3 - 70% від максимуму

Для підняття ніг:

Перше тренування:

– Підхід 1 - 65% від максимуму

– Підхід 2 - 55% від максимуму

– Підхід 3 - 50% від максимуму

Друге тренування:

– Підхід 1 - 70% від максимуму

– Підхід 2 - 60% від максимуму

– Підхід 3 - 55% від максимуму

Друге тренування:

– Підхід 1 - 75% від максимуму

– Підхід 2 - 65% від максимуму

– Підхід 3 - 60% від максимуму

Ці значення будуть використані в якості матриці для блок-схеми на рис. 2.15

Таким чином програма тренувань буде підлаштовуватись під користувача. З кожним разом показники максимальної кількості вправ будуть збільшуватись, і програма буде змінюватись. І навіть якщо користувач пропускав заняття, та повернувся через довгий час, у нього просто зменшаються максимальні показники і програма зміниться відповідним чином. Даний алгоритм наведений на рис. 2.15.

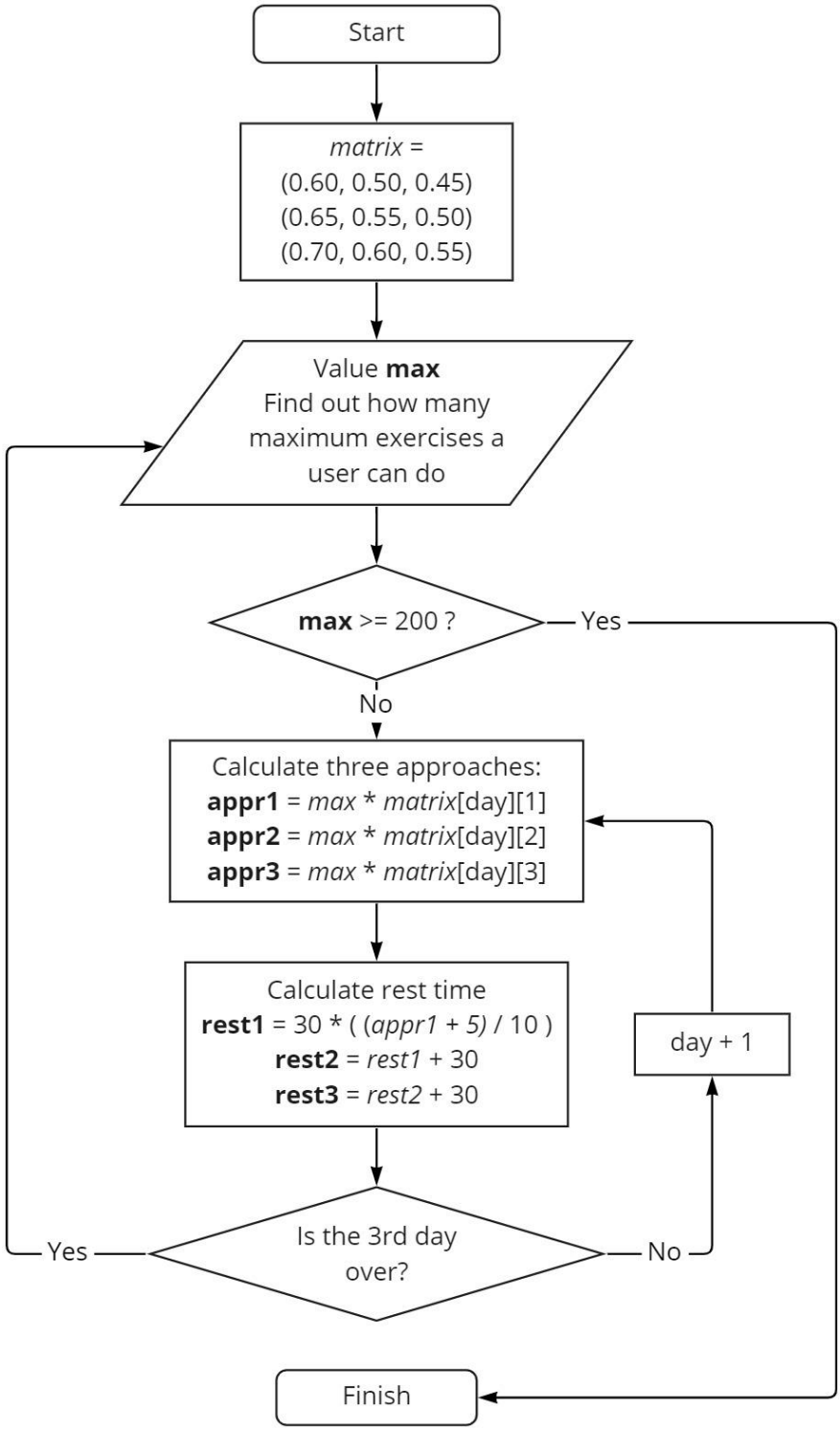

Рисунок 2.15 – Діаграма алгоритму створення індивідуального графіку занять

Висновки до розділу 2

У другому розділі біли розглянуті такі базові вправи як віджимання, присідання та підйом ніг, та досліджено як їх правильно робити.

Розглянуті основні існуючі технології відстеження сегментації тіла, а саме: BodyPix, MoveNet, BlazePose та PoseNet. Кращий варіант з них MoveNet, так як працює найшвидше з усіх варіантів. Найзручніший спосіб їх встановлення це використати бібліотеку TensorFlow. Ще було детально розглянуто як зробити алгоритм класифікації вправ і зарахування балів за їх виконання. Також були розглянуті алгоритми створення індивідуального графіку занять спортом і створено свій.

3 МОДЕЛЮВАННЯ ТА ТЕХНІЧНЕ ПРОЄКТУВАННЯ ЗАСТОСУНКУ

3.1 Прототипування та дизайн програмного забезпечення

Правильне розміщення елементів інтерфейсу підвищує зручність використання та робить продукт більш привабливими для користувачів. У деяких випадках при грамотному проектуванні можна схилити користувачів зробити певні дії. Потрібно чітко розуміти, які очікування від програми та які елементи повинні бути на кожному екрані.

Для цього на етапі проектування спочатку створюється прототип – чорно-білий макет, що представляє спрощену схему програми Він містить усі основні елементи, представлені у вигляді блоків, тому можна оцінити основну концепцію.

Прототипування – це один із початкових етапів розробки, під час якого створюється ескізний вигляд програми. При створенні прототипу створюється макет для імітації взаємодії користувача з інтерфейсом проекту. Прототип потрібен для попереднього перегляду проекту і оцінки його зручності. Тестування макета дозволяє виявити та усунути помилки завчасно, ще до того як почати розробку. Прототип дуже схожий за структурою і функціями на готовий продукт, але відрізняється від нього. Його можна намалювати на папері або створити в графічному редакторі. Основною відмінністю цих методів є рівень деталізації та інтерактивність елементів.

Основна мета створення прототипів – заощадити гроші та час. З самого початку важко створити ідеальний продукт, зручний для користувачів. Прототипи дозволяють протестувати вибране рішення без великих вкладень і зробити зміни, якщо необхідно, перед початком розробки дизайну та програмування.

Створення прототипів не тільки допомагає визначити та розвивати основний напрямок дизайну, але й може заощадити багато часу. На створення концепції уйде день, але не доведеться витрачати тиждень на розробку та дизайн. Але це не єдина причина приділяти час прототипу.

Прототипування вирішує кілька важливих завдань:

Пошук кращих ідей. Прототипування відбувається набагато швидше, тому мається можливість підготувати кілька варіантів перевірки гіпотез і можна обрати найбільш вдалий.

Виявлення помилок. На етапі створення макета можна відстежити ключові слабості майбутніх програм. Ви витратите менше часу, грошей і зусиль на їх виправлення на початку, ніж на внесення змін до кінцевого продукту.

Оцінка зручності. Створення прототипів і тестування користувацьких сценаріїв — це чудовий спосіб перевірити, наскільки зручне рішення є на ранніх стадіях.

Важливо створити простий у використанні інтерфейс. Інтерфейс має бути розроблений таким чином, щоб він був простим і легким у використанні, щоб навіть початківці користувачі могли отримати доступ до функцій програми та використовувати їх. Крім того, інтерфейс повинен бути візуально привабливим і сучасним, щоб зацікавити користувачів, які шукають інноваційний і стильний мобільний додаток.

Для розробки початкового прототипу була обрана програма Figma.

Figma — це програмне забезпечення для розробки користувальницького інтерфейсу, яке дозволяє користувачам швидко та ефективно створювати гарні та зручні інтерфейси. Завдяки інтуїтивно зрозумілому інтерфейсу дизайну, Figma робить створення складних інтерфейсів легким і зрозумілим. Крім того, ця програма пропонує широкий спектр функцій, які дозволяють оптимізувати процес розробки інтерфейсу.

Спочатку створені прототипи початкового екрану, логіну та реєстрації (рис 3.1)

Рисунок 3.1 – Прототипи початкового екрану, логіну та реєстрації

Починається все з екрану «Starting», з логотипом застосунку та єдиною кнопкою «Start». На етапі дизайну назви кнопок та опис можуть змінитись.

Після натискання кнопки старт йде перехід на екран «Sign In», на якому користувач має змогу увійти, ввівши свої дані, або зареєструватись на екрані «Sign Up» якщо їх немає.

Авторизація відноситься до процесу, за допомогою якого користувачеві надаються різні права та привілеї в програмному забезпеченні. Реєстрація також є процесом, за допомогою якого користувач реєструється для використання програмного забезпечення. Цей процес необхідний для функціонування програмного забезпечення та дозволяє користувачам отримати доступ до його функцій. Авторизація та реєстрація зазвичай відбуваються на початку взаємодії користувача з програмним забезпеченням. Реєстрація є важливим заходом безпеки, який допомагає захистити ваш обліковий запис і дані від несанкціонованого доступу.

Далі створені три основних екрани, переключатись між якими можна за допомогою нижньої навігаційної панелі, та створено додаткове меню зверху, щоб можна було залишити відгук та вийти з аккаунту (рис. 3.2).

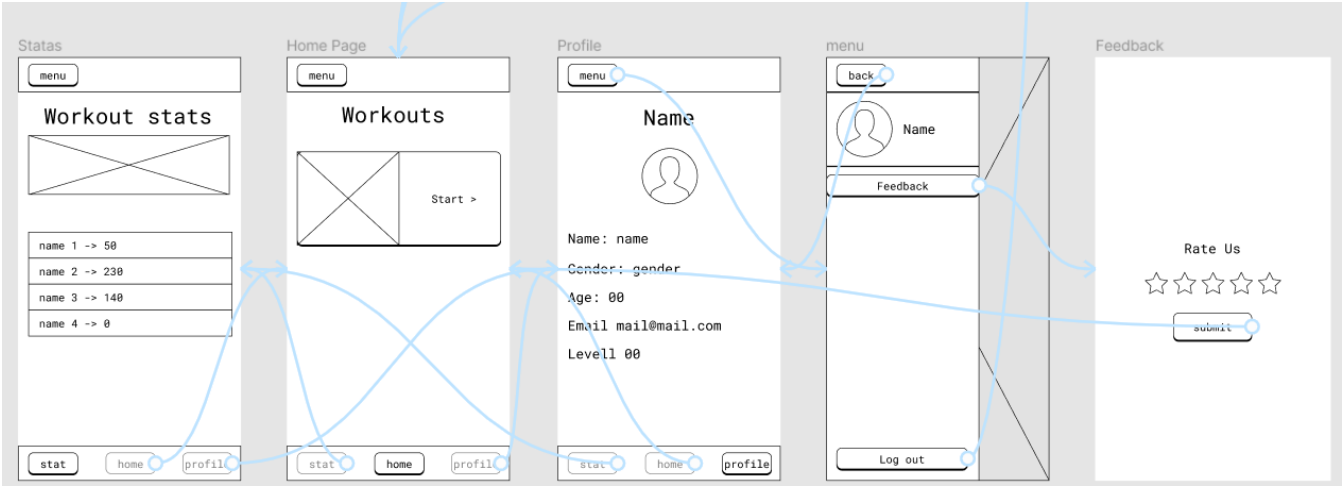

Рисунок 3.2 – Екрани: “Stats”, “Home Page”, “Profile”, “Menu”, “Feedback”

На екрані “Stats” Можна буде переглянути поточну статистику та кількість виконаних вправ за весь час. Це додає у процес виконання вправ цікавості та елементів з ігрової індустрії.

Сторінка “Home Page” обирається за замовчуванням після логіну. Сторінка “Profile” відображує інформацію про користувача. Там теж можна побачити ігрову механіку – рівень користувача.

Сторінка “Menu” викликається з будь-якої з вищезазначених сторінок, якщо натиснути на верхню кнопку меню. На цій сторінці можна вийти з аккаунту або перейти на сторінку “Feedback”, да можна поставити рейтинг.

Панель навігації є важливою частиною будь-якої програми. Вона надає користувачеві швидкий і простий спосіб отримати доступ до різних областей програми. Панель навігації має бути простою, ефективною та чутливою. Раніше їх розташовували зверху, але досвід показав, що найкраще місце для розташування меню це низ програми. Туди простіше натиснути, і люди, які далекі від новітніх технологій не розуміють де меню зверху, а от знизу це інтуїтивно зрозуміло. Панель навігації важлива, оскільки забезпечує швидкий спосіб для користувачів отримати доступ до важливих функцій програми.

Далі створено сторінки з тренуваннями (рис. 3.3), на які можна перейти з “Home Page”.

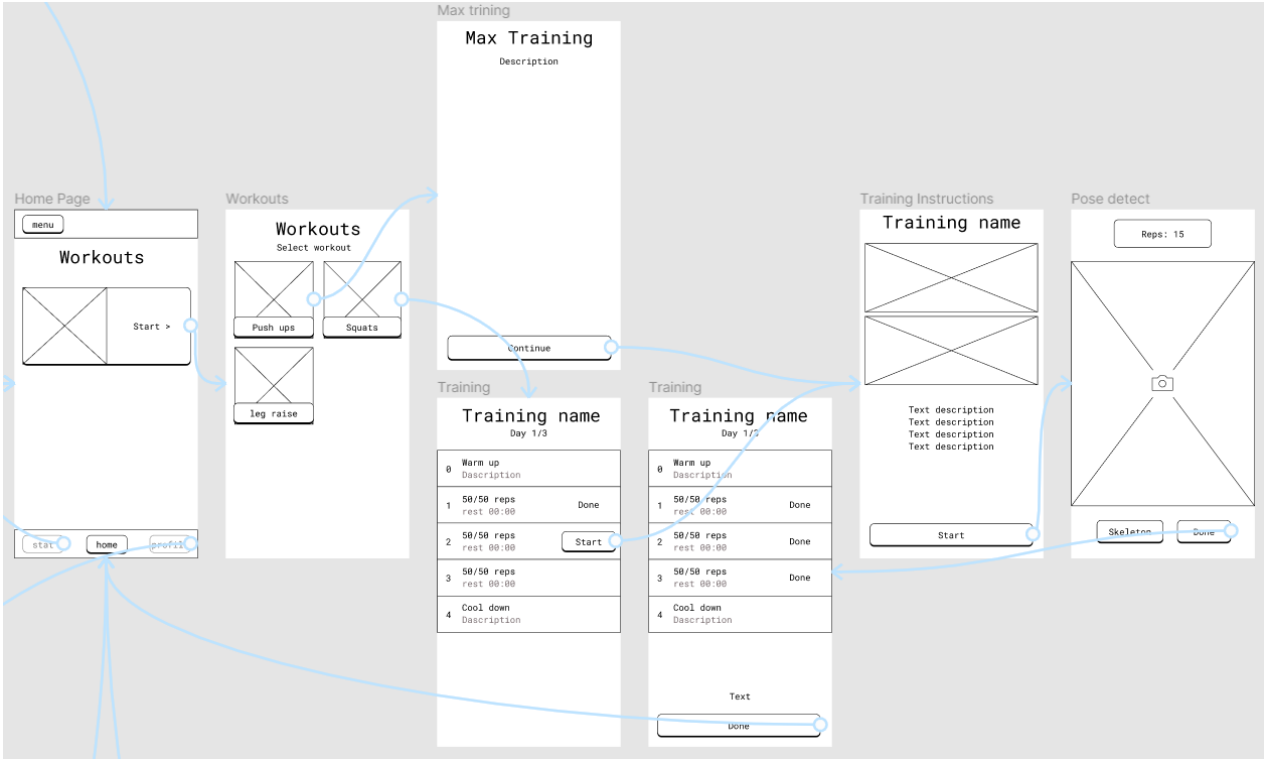

Рисунок 3.3 – Сторінки з логікою тренувань.

Сторінка “Workouts” пропонує обрати тип тренувань. В рамках даної БКР буде створено три вправи – Віджимання, Присідання та Підняття ніг (вправа на прес).

Якщо це перший запуск, або програма була звершена, то користувач переходить до сторінки “Max Training”. На ній буде сказано що треба прикласти зусиль. Далі буде йти сторінка “Training” з поясненням, як виконувати вправу щоб програма правильно працювала. І потім йде “Pose detect” – сторінка з контролем виконання вправ.

Сторінка “Training instructions” буде відкриватись коли буде відомо скільки максимум вправ зможе виконати користувач. В залежності від цього буде створена програма на три дні, з трьома підходами у кожному.

Далі, на основі прототипу створено **дизайн застосунку.** Під час створення Потрібно зробити підбір стилістики, яка відповідає завданням, авторитету та іміджу бажаного застосунку. Потрібно створити концепцію програми та розробки дизайну. Виконати розробку станів екранів, елементів дизайну згідно з

прототипами. Та не забути підготувати графічні матеріали — іконки, ілюстрації, фото для розробників.

Дизайн - найкращий спосіб створити потрібну атмосферу для майбутніх користувачів. Після запуску програми кольори, форми, шрифти та кнопки викликують у користувача або бажання продовжувати користуватися програмою або бажання якнайшвидше видалити застосунок. Потрібно, щоб атмосфера закохала користувача з першого натискання.

Коли є остаточна структура та прототип, які були протестовані та затверджені, починається розробка дизайну інтерфейсу. Звичайно, завдання — створити гарний продукт, але це ще не все. Насправді зовнішній вигляд будь-якого продукту має набагато глибший, психологічний вплив на користувача. Наприклад, враження від інтерфейсу зі світлими постільними кольорами будуть відрізнятись від інтерфейсу з чорними брутальними кольорами. Це лише деякі з рішень, які враховують дизайнери інтерфейсу користувача.

Створення дизайну було продовжене у програмі Figma. На наступних рисунках неведені наглядні результати розробки дизайну з прототипів. Всі переходи між екранами працюють аналогічним, запланованим раніше чином.

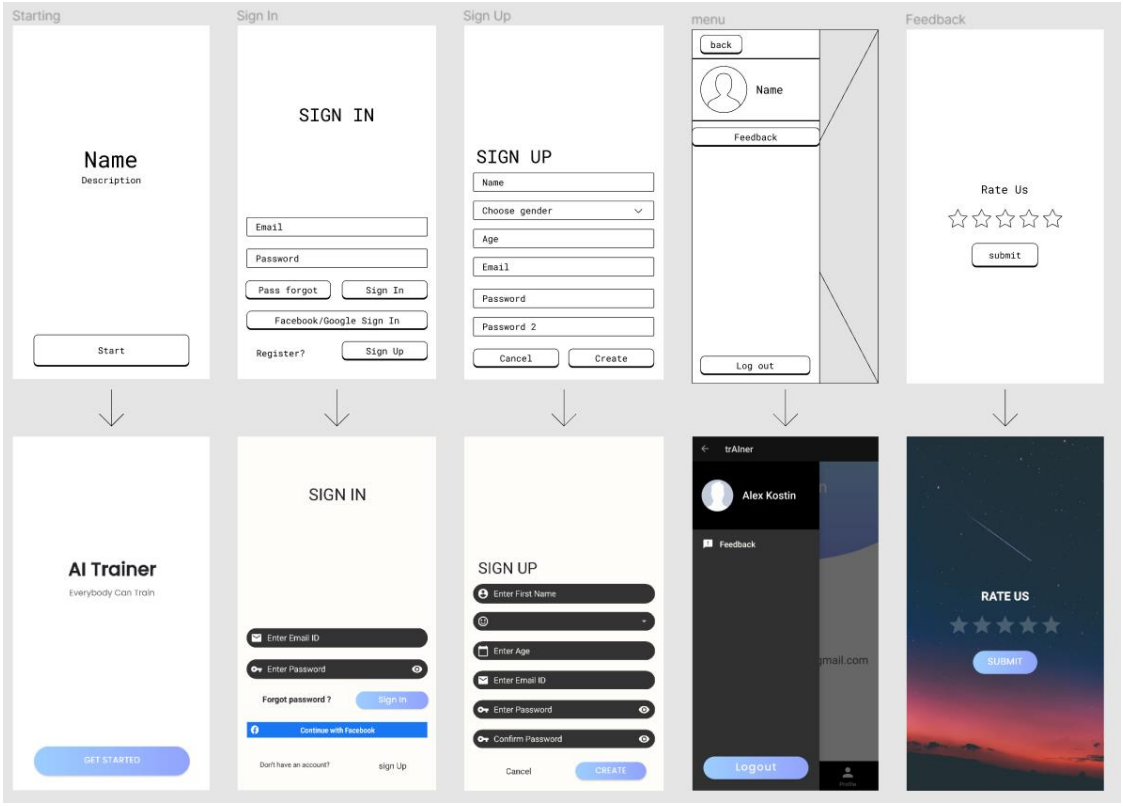

Рисунок 3.4 – Дизайн для прототипів входу у застосунок

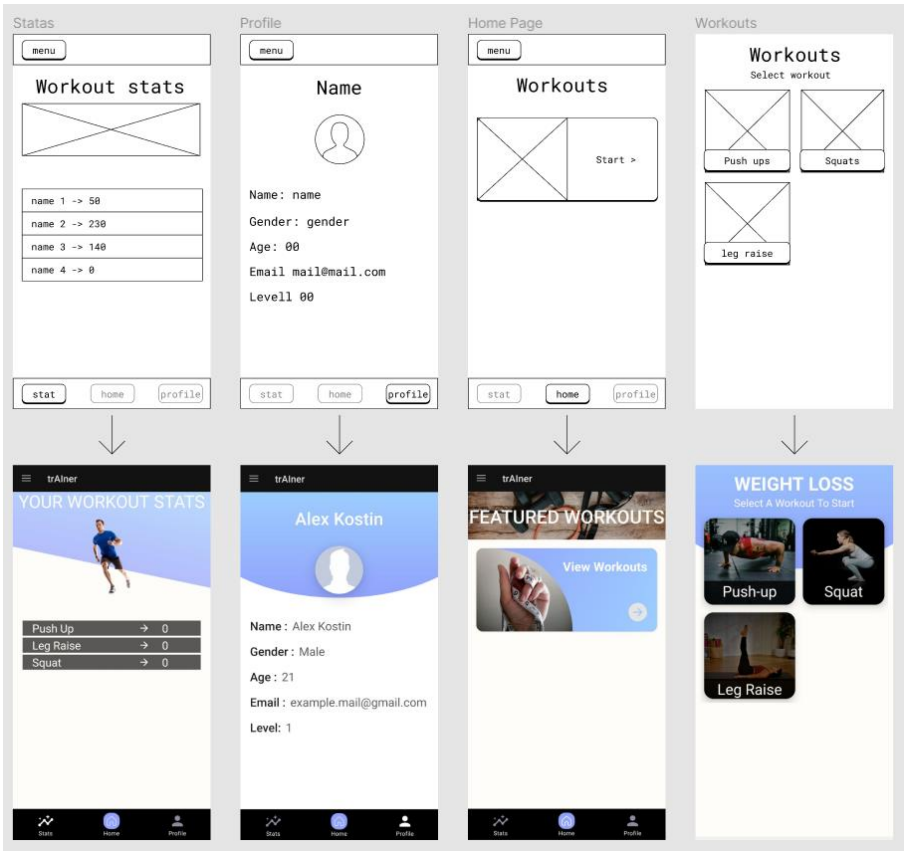

Рисунок 3.5 – Дизайн для прототипів основних меню

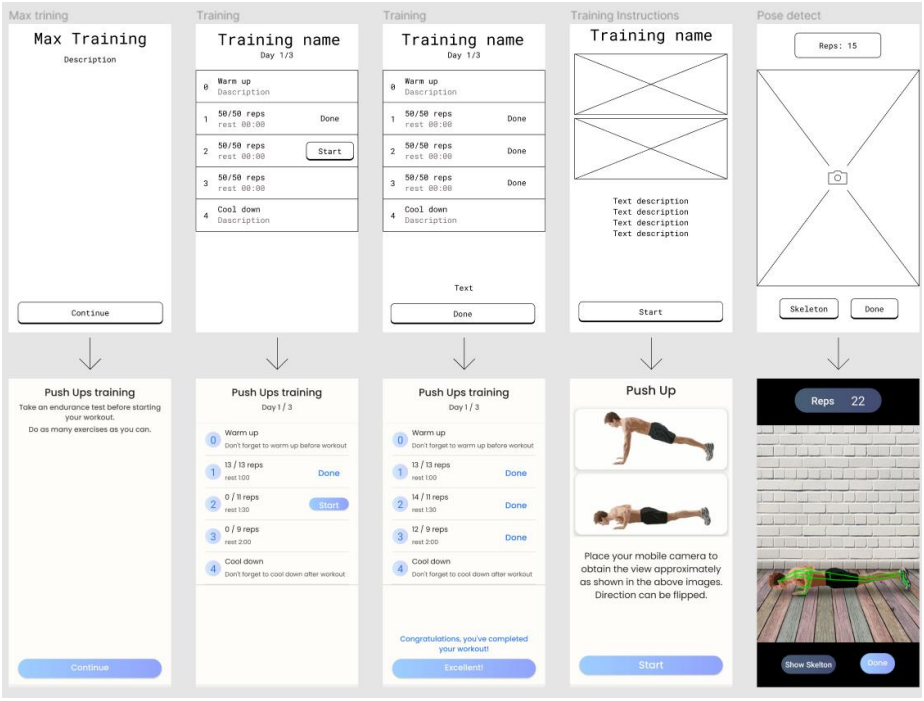

Рисунок 3.6 – Дизайн для прототипів тренувань

3.2 Архітектура програмного забезпечення

Після розробки прототипу та дизайну, слід розробляти архітектуру застосунку.

Архітектура програмного забезпечення - це план створення та підтримки програмної системи. Він визначає структуру та розташування компонентів системи, а також надає вказівки для безпомилкової роботи.

Добре розроблена програмна система може досягти видатної продуктивності та надійності. Однак, якщо система погано спроектована, вона може страждати від кількох поширених проблем, зокрема:

Складність: погано розроблена система важка для зрозуміння та підтримки. В результаті вона може стати неефективною, що призведе до ускладнень і втрати часу.

Крихкість: погано спроектована система також схильна до нестабільності та помилок. Якщо один компонент системи виходить з ладу, вся система може стати непотрібною.

Складні взаємозалежності: погано спроектована система, як правило, взаємопов’язана складним чином. Це дозволяє проблемам швидко поширюватися і спричиняти широкі збитки.

Для розробки архітектури буде використана UML діаграма.

UML – це уніфікована мова моделювання. Modeling передбачає створення моделей, що описують об'єкти. Unified (універсальний) – застосовується до широкого спектру систем проектування, різних галузей застосування, типів організацій, рівнів можливостей та масштабів проекту [15]. UML описує об’єкти в єдиному синтаксисі, тому незалежно від того, де ви намалюєте діаграму, будь-хто, хто знайомий з мовою графіки, буде знати правила, навіть в іншій країні.

Деякі типи діаграм специфічні для системи та програми.

В UML діаграми дозволяють узагальнити відомості про користувачів системи (також званих акторами) і те, як вони взаємодіють із системою. Для його створення ви будете використовувати спеціалізований набір символів і сполучників. Ефективне використання діаграм може допомогти вашій команді обговорити та продемонструвати:

– Сценарії, в яких ваша система або програма взаємодіють з людьми, організаціями або зовнішніми системами

– Цілі, яких ваша система або програма допомагає досягти цим суб’єктам (відомим як актори).

– Сфера дії вашої системи

Use Case – це сценарна техніка опису взаємодії [15]. Діаграма варіантів використання не містить багато деталей. Наприклад, не очікуйте, що вона моделює порядок виконання кроків. Натомість, правильна діаграма варіантів використання зображує високорівневий огляд взаємозв’язків між варіантами використання, акторами та системами. Експерти рекомендують використовувати діаграми варіантів використання, щоб доповнити більш описовий текстовий варіант використання.

Випадки використання представлені овальною формою з маркуванням. Фігурки зображують акторів у процесі, а участь актора в системі моделюється лінією між актором і варіантом використання. Щоб зобразити межі системи, намалюйте прямокутник навколо самого варіанту використання.

Нижче наведена UML Use case діаграма застосунку.

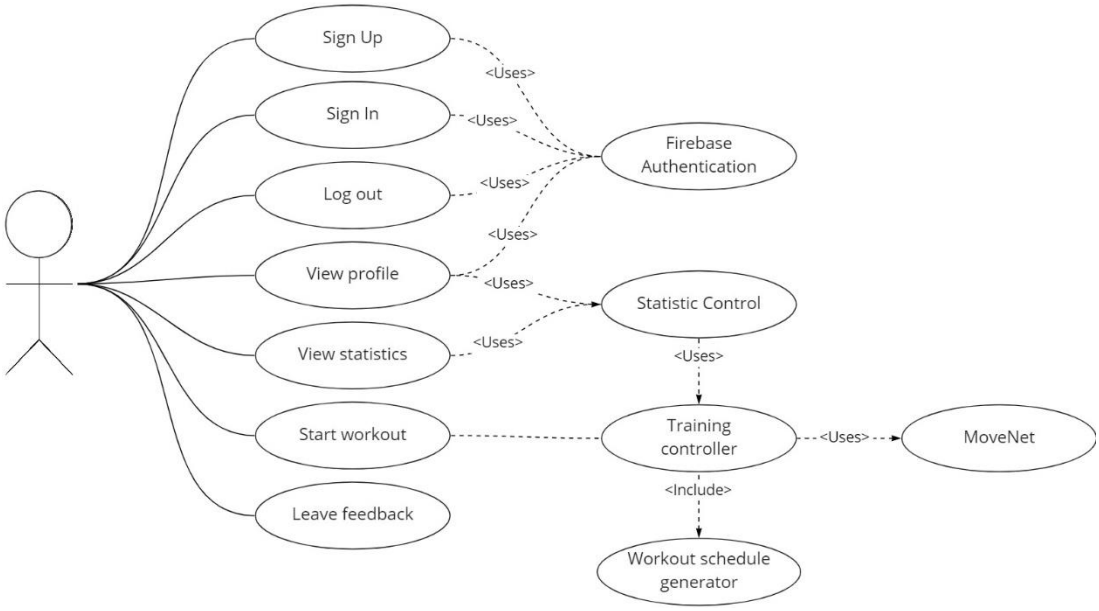

Рисунок 3.7 – Use Case Діаграма

На ній зображений користувач, якому потрібно виконувати такі дії як реєстрація, авторизація, вихід за аккаунту та переглянути свій профіль. За це все відповідає модуль авторизації Firebase Authentication.

Також потрібно переглядати свою статистику. Буде розроблений окремий модуль Statistic Control. Він же буде відповідати за рівень гравця, що буде виводитись у його профілі.

Користувач може почати тренування. Це – основний модуль для розробки. Щоб він добре працював потрібно Зробити контролер тренувань. Він буде зараховувати віджимання через контролер положень тіла MoveNet, та через модуль генерації тренувань. Ще він буде ділитись даними з контролером статистики.

Також користувач може залишити свій фідбек.

Але це не достатньо детальна діаграма для розробки. В якості основи архітектури доцільно використати також діаграму класів.

Діаграма класів є одною з найбільш популярних типів в UML. Діаграми класів популярні серед інженерів-програмістів для документування архітектури програмного забезпечення. Вони є різновидом структурної діаграми, оскільки вони описують те, що має існувати в змодельованій системі.

Незалежно від того, чи ви знайомі з UML або діаграмами класів, UML розроблено так, щоб кожен міг її зрозуміти [16]. UML був створений як стандартизована модель для опису методів об'єктно-орієнтованого програмування. Оскільки класи є будівельними блоками об’єктів, діаграми класів є будівельними блоками UML. Різні компоненти діаграми класів можуть представляти фактичні класи програмування, основні об’єкти або взаємодії між класами та об’єктами.

Форма самого класу складається з прямокутника з трьома лініями. Верхній рядок містить назву класу, середній рядок містить властивості класу, а нижній рядок містить методи чи операції, які може використовувати клас. Класи та підкласи згруповані разом, щоб показати статичні відносини між кожним об’єктом. На рис. 3.8 зображена діаграма класів для застосунку.

Там можна побачити кожен клас що буде у застосунку. Все буде починатись з правого нижнього кутку, з класу LoginActivity. Від нього походять SignIn та SugnUp. За допомогою них користувач матиме змогу зареєструватись та увійти у свій аккаунт.

Наступний клас це клас HomeScreen. Основний клас, від якого ще походять HomeFragment, FeedbackActivity, StatsFragment,ProfileFragment та SettingsActivity. Вони відповідають за основне навгаційне меню.

Також досить важливі класи WeightLoss, та TrainingPage. Останній відповідає за індивідуальний графік тренування користувача і статистикою що у нього є. Лівіше від нього є клас MyPreferences. Це клас де зберігаються основні дані про користувача і його тренування.

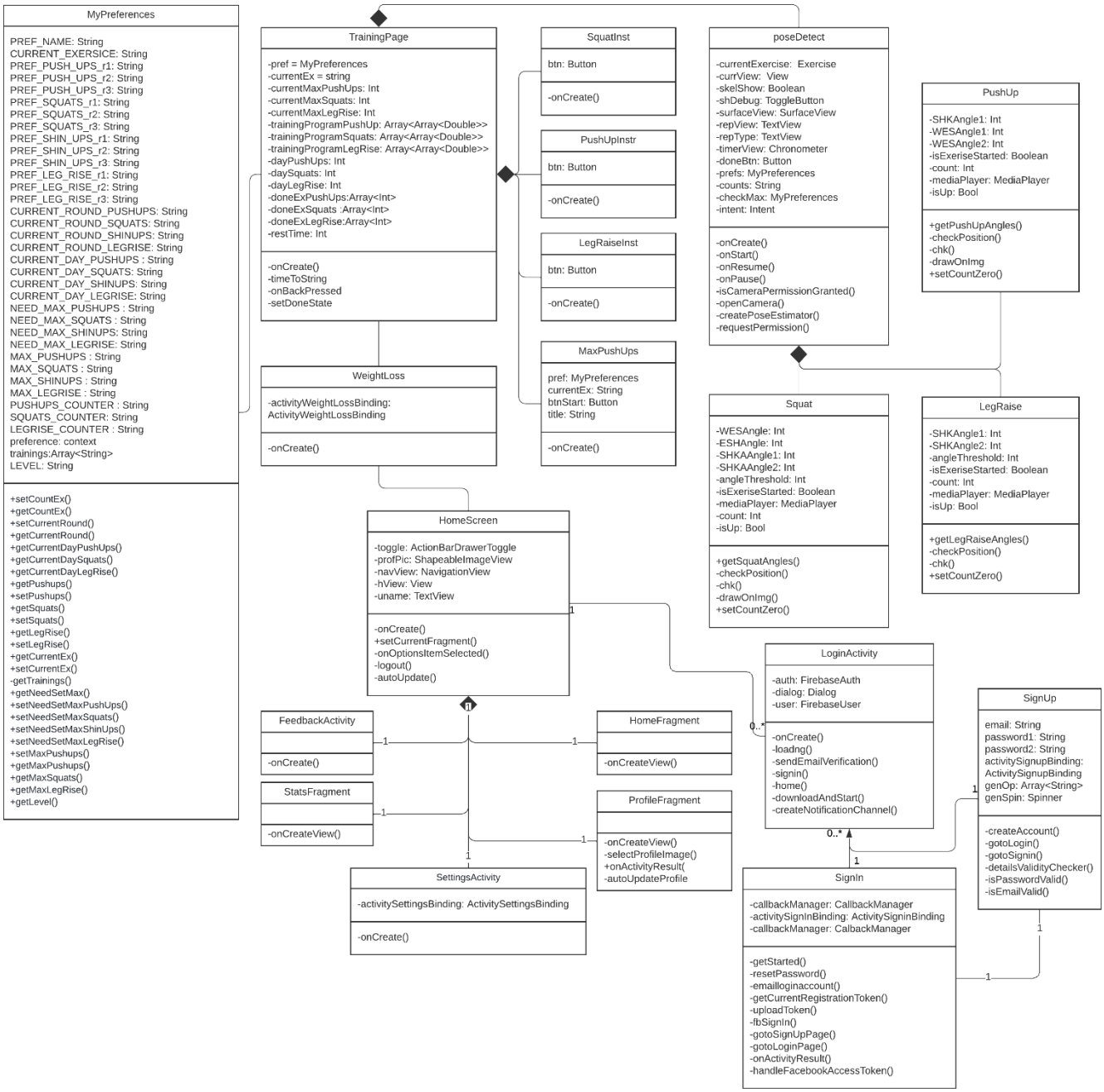

Рисунок 3.8 – Діаграма класів застосунку

Від класу TrainingPage залежать класи з інструкціями, так як без нього вони не потрібні. А ще цей клас викликає Pose Detect, від якого, в свою чергу, йдуть класи класифікації поз.

Висновки до розділу 3

Розроблено прототип та дизайн для майбутньої програми. Створені сторінки авторизації та реєстрації, сторінки у головному меню та переключення між ними. Зроблене бокове меню з фідбеком та дизайн і логіка для сторінки з вправами і заняттями. Також промальована сторінка з індивідуальним графіком занять і сторінки де камера з відстеженням дій.

Розроблено архітектуру програмного забезпечення, розроблено use-case UML діаграму, де описані дії користувача. Також розроблено UML діаграму класів, де прописані всі класи, їх дані, методи, властивості за залежності.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Обґрунтування та вибір технологій розробки ПЗ

Для реалізації всіх потреб необхідно обрати технології, які зможуть втілити у життя всі поставлені цілі.

Операційна система.

Операційна система — це набір програм, які допомагають пристрою запускати різні програми. Без операційної системи електронний пристрій не зможе отримати доступ до файлів або інших пристроїв на жорсткому диску і не запуститься взагалі. Щоб користуватися пристроями, потрібна операційна система. Щоб користуватися додатками, потрібна операційна система. Для доступу до даних потрібна операційна система. Операційна система потрібна, щоб мати можливість взаємодіяти з фізичним світом.

Оскільки для контролю занять потрібно мати камеру та мати змогу розташувати її де завгодно, та ще й взаємодіяти з інтерфейсом, найзручнішим буде використання смартфонів з камерою та використовувати їх операційні системи. Для цього була обрана зручна система Android.

Android – найпопулярніша мобільна операційна система у світі. Вона доступна на телефонах різних виробників. ОС Android – це безкоштовна платформа, а це означає, що вам не потрібно нічого платити, щоб використовувати її. Ця операційна система заснована на ядрі Linux. Це повна операційна система, що означає, що вона включає драйвери для обладнання, інтерфейс користувача, мережу, сховище тощо. ОС Android має багато функцій, які можна використовувати.

Мова програмування.

Коли справа доходить до програмування, мова, яка використовується для написання коду, настільки ж важлива, як і сам код. Існує багато мов програмування для написання програм, які працюють на операційній системі Android. Основні з них такі:

1) Java
2) Kotlin
3) Flutter

Для розробки застосунку обрана мова програмування Kotlin.

Kotlin – це нова мова програмування, яка почала набирати популярності в останні кілька років. Він розроблений, щоб бути швидким, виразним і лаконічним. Популярність Kotlin серед розробників Android пов’язана з його потужними функціями для розробки мобільних додатків. Ця мова створена JetBrains, компанією, яка розробляє середу розробки IntelliJ.

Kotlin дуже добре працює разом з Java і схожий на Java за своїм синтаксисом. Це означає, що кодери, які знайомі з Java, можуть дуже легко почати працювати з Kotlin. Крім того, Kotlin взаємодіє з Java, що означає, що існуючий код Java можна перенести на Kotlin з незначними змінами або без них.

Середовище розробки.

Програмістам потрібне середовище розробки, щоб писати код ефективно та ефективно. Хороше середовище розробки допомагає програмістам слідувати набору вказівок щодо кодування, належним чином керувати файлами проекту та відстежувати хід виконання своїх проектів. Крім того, це може допомогти програмістам швидко компілювати свій код і виконувати на ньому тести.

Android Studio — інтегроване середовище розробки, створене Google для роботи з конкретно платформою Android. Android Studio є найкращим середовищем розробки для розробки Android з кількох причин; він простий у використанні, пропонує гарні функції, добре розроблений та має чудову підтримку.

Android Studio, заснована на програмному забезпеченні IntelliJ IDEA від компанії JetBrains. Це офіційний засіб розробки Android додатків. Також Google анонсував підтримку мови Kotlin у 2017 році, як офіційної мови програмування для платформи Android на додаток до Java та C++.

Система контролю версій.

Найпопулярнішою системою контролю версій є GitHub.

Git — розподілена система контролю версій, яка дає змогу розробникам відслідковувати зміни у файлах та працювати над одним проектом разом із колегами.

GitHub - сервіс онлайн-хостингу репозиторіїв, що володіє всіма функціями розподіленого контролю версій та функціональністю управління вихідним кодом - все, що підтримує Git і навіть більше. Також GitHub має контроль доступу, багтрекінг, керування завданнями та вікі для кожного проекту.

Модель трекінгу поз людини.

У пункті 1.2 уже був проведений детальний огляд існуючих систем відстежування позицій тіла. Там було обрано MoveNet.

MoveNet - це дуже швидка і точна модель, яка на основі нейронних мереж виявляє 17 ключових точок тіла. Вона працює швидко у режимі реального часу (30+ FPS) на більшості сучасних настільних комп'ютерів, ноутбуків і телефонів, що виявляється вирішальним для додатків для фітнесу, здоров'я та оздоровлення.

Ця модель підключається за допомогою бібліотеки TensorFlow.

Робота з хмарними сервісами.

Firebase – це платформа для розробки програм Backend-as-a-Service (BaaS), яка надає розміщені серверні послуги, такі як база даних у реальному часі, хмарне сховище, аутентифікація, звіти про аварійне завершення роботи, машинне навчання, віддалена конфігурація та розміщення ваших статичних файлів.

Розробка дизайну

Figma — це універсальна програма, яку можна використовувати для різних цілей, наприклад, для створення інтерфейсів користувача для додатків. На відміну від деяких інших програм, розроблених для певних цілей, Figma є більш універсальним і може використовуватися для різних завдань, починаючи від

розробки інтерфейсу користувача для програми і закінчуючи створенням прототипів. Крім того, Figma є зручною та легкою в освоєнні, що робить її чудовим варіантом для тих, хто не знайомий з програмуванням чи програмами для проектування.

Lucidchart — це веб-додаток для створення діаграм, який дозволяє користувачам візуально співпрацювати над малюванням, переглядом та обміном діаграмами та діаграмами, а також покращувати процеси, системи та організаційні структури.

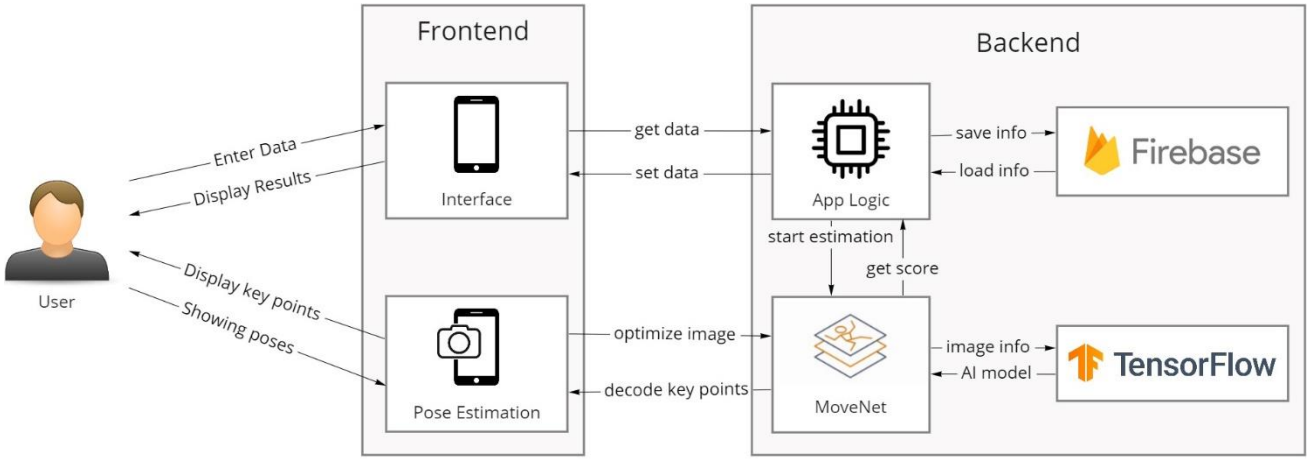

Рисунок 4.1 – Структурна діаграма застосунку

4.2 Опис програмної реалізації

Розглянемо програму реалізацію. Проект побудований на класах. UML Діаграму класів можна переглянути на рис. 3.8

Реалізація Sign In.

При створені, йде перевірка що застосунок тільки включений. Якщо це так, то викликається метод getStarted. Він одразу викликає діалогове вікно з привітанням. У XML файлі в вікні є Назва програми, привітання і кнопка. В класі йде пошук цієї кнопки, та програма слухає, коли вона буде натиснута. Після натискання діалогове вікно закривається і вікно з логіном відкривається повністю. Також флаг що застосунок тільки запущений перемикається на false.

Вікно логіну чекає на заповнення даних та натискання кнопки входу. І перевіряє, щоб пошта і пароль були заповнені. Якщо вони путі, то виводиться

помилка «Please Enter Your email and Password». Якщо паролі заповнені, то визивається функція emailloginaccount, яка перевіряє, чи є такі дані у Firebase. Якщо є, то логін успішний і користувача переводять на домашню сторінку. В іншому випадку користувачу біде виведена помилка «SignIn failed.»

Також відслідковується натискання на логін через Facebook. При натисканні спрацьовує функція fbSignIn. Вона перевіряє чи вийшло у користувача успішно війти в аккаунт Facebook. Якщо ні то виводиться помилка, а якщо так, то передає токен фейсбук у функцію handleFacebookAccessToken. Якщо токен проходить перевірку, то користувач входить у систему. В іншому випадку виводиться помилка «Authentication failed.»

Також є очікування на натискання кнопки переходу до реєстрації. Після її натискання запускається функція gotoSignUpPage, і користувача переправляє на вікно SignUp

Реалізація Sign Up.

При старті класу одразу йде відслідковування на натискання випадаючого списку з вибором статі. Для нього реалізований окремий layout, з пунктами вибору. Полю введення передається обраний пункт.

Далі йде очікування на натискання кнопки «Create». Потім з усіх полів для вводу беруться дані та перевіряються на коректність.

1) При реєстрації потрібно обов’язково ввести всі поля. Якщо вони не введені то буде помилка «Please Fill all details».

2) Вік може бути в діапазоні від 18 до 65. Якщо це не так, то буде помилка «Only 18 - 65 are allowed»

3) Якщо в статі нічого не обрано то виводиться помилка «Please select a gender»

4) Пароль та повторний пароль повинні бути однакові. Якщо це не так, т виводиться помилка «Passwords does not Match»

5) Пароль повинен бути довшим за 6 символів. Якщо це не так то виводиться помилка «Minimum password Length 6».

6) У паролі не повинно бути пробілу. Помилка якщо це не так: «Password can't Contain Space».

7) Пошта повинна мати формат пошти. Якщо це не так, то виводиться помилка «Enter A valid email».

Якщо всі дані вірні, то до Firebase додається користувач з введеними даними і йде перехід до вікна логіну, де ці дані можна ввести.

Реалізація Home Screen.

При запуску домашнього екрану створюється шапка з меню зверху. При натисканні меню в'їжджає з лівого боку. Там промальовується аватар користвуаа та його ім'я. Також йде прослуховування на натискання кнопки Feedback, після чого користувача переведе на сторінку Feedback Activity. Також при натисканні на кнопку Logout користувач виходить з аккаунту, та його направляють на початковий екран.

Потім створюється навігаційна панель знизу. Там доступні три кнопки – Stats, Home, Profile, при натисканні на які запускаються відповідні фрагменти.

Реалізація Feedback Activity.

Якщо користувач у меню натиснув на кнопку Feedback, його буде направлено на вікно залишення відгуку. Так як застосунок ще не доданий у жоден з магазинів застосунків, то кнопки нічого не роблять.

Реалізація Profile Fragment.

При створенні дані беруться з Firebase та вноситься у поля імені, статі, віку, та пошти. Рівень береться з класу MyPrefrences, бо він там і обраховується. Рівень залежить від статистики. Береться сума всіх виконаних вправ і ділиться на 100.

Реалізація Stats Fragment.

При запуску дані для кожної із виду тренувань беруть з класу MyPrefrences. Вони обраховуються при відстежуванні рухів та натисканні на завершення вправ.

Реалізація Home Fragment

При викликанні очікує на натискання на карточку початку тренувань. При натисканні переводить у вікно Weight Loss

Реалізація My Preferences

Клас для роботи з preferences. Має змінні які відповідають за те, яка вправа поточна, скільки на даний момент зроблено вправ, який раунд поточний, який день поточний, скільки користувач робить максимум вправ, змінні які рахують скільки всього зроблено вправ і змінні що вказують чи потрібно зараз вичислити максимальну кількість вправ.

Для кожної зі змінних зроблені свої функції гетери та сетери для зручного користування ними.

Також є функція getLevel, яка на основі того скільки сього зроблено вправ розраховує рівень користувача. Рівень збільшується кожен раз, коли в сумі виконано 100 вправ.

Реалізація Weight Loss

При створенні сворюється об’єкт класу MyPrefrences.

Йде відслідковування натискання на одну з карток. При натисканні на віджимання, до MyPrefrences у змінну «Current Exercise» передаються віджимання. Потім йде перевірка, чи потрібно для віджимань дізнатись максимальну кількість вправ. При першому старті скоріш за все прийдеться. Але також це відбувається коли програма тренувань завершена.

Тож якщо потрібно дізнатись максимум, то користувач переходить у вікно Max, якщо не потрібно, то у вікно Training Page. Точно так, як і віджимання, працє логіка у присіданнях і підняттях ніг, але вказується інша вправа у змінній «Current Exercise».

Реалізація Max

При створенні йде перевірка поточної вправи у змінній «Current Exercise». Якщо це віджимання, то заголовок буде «Push Ups training», та при натисканні на старт користувача направить до вікна Push Up Instruction.

Якщо це присідання, то заголовок буде «Squats training», та при натисканні на старт користувача направить до вікна Squats Instruction.

Якщо це підняття ніг, то заголовок буде «Leg Raise training», та при натисканні на старт користувача направить до вікна Leg Raise Instruction.

Реалізація Training Page

При створенні цього класу у змінні одразу передаються всі кнопки та написи. Також записується скільки максимум користувач може робити кожної вправи. Тут же створюються масиви з планом тренувань. У розділі 2.3 були продумані алгоритми створення індивідуального графіку занять. Потрібно взяти конкретний процент від максимуму занять, які були зафіксовані під час роботи у вікні Max.

Для віджимань це матриця trainingProgramPushUp = ((0.6, 0.5, 0.45), (0.65, 0.55, 0.5), (0.7, 0.6, 0.55)), у якому перший індекс це день, а другий це підхід.

Аналогічно для присідань та віджимань.

trainingProgramSquats = ((0.7, 0.6, 0.5), (0.8, 0.7, 0.6), (0.9, 0.8, 0.7))

trainingProgramLegRise = ((0.65, 0.55, 0.50), (0.7, 0.6, 0.55), (0.75, 0.65, 0.60))

Тут же створена змінна з поточним днем тренування для кожної з вправ, та масив з даними скільки вже зроблено вправ.

Також є змінна відпочинку, яка на початку дорівнює 30 секундам.

Далі йде визначення яка зараз поточна вправа у змінній «Current Exercise». Якщо це віджимання, то заголовок змінюється на «Push Ups training». Також для поточного дня встановлюються програма. Кількість вправ для підходу розраховується за множенням проценту з матриці trainingProgramPushUp на максимальну кількість вправ яку зараз може зробити користувач.

Формула розрахунку відпочинку наведена нижче.

$$T_r = \frac{M_1 + 5}{10} \times T_r ,$$

де T_r – час відпочинку;

M_1 – вправ у першому підході.

Це відпочинок після першого підходу. Для наступних потрібно додавати ще по 30 с.

Потім йде перевірка на поточний раунд. Річ у тім, що у XML файлі є одразу всі кнопки (рис. 4.2), але для кожного раунду якісь є, а якихось немає.

Рисунок 4.2 – Реалізація Training Page у XML файлі

Таким чином йде перевірка який раунд, і для першого раунду активна лише перша кнопка старт. Для другого раунду активна друга кнопка старт, а перша становиться написом «Done». Для третього раунду перші дві кнопки стають написом «Done», а третя кнопка активна. Після проходження всіх раундів всі

кнопки становляться «Done», у з’являється нижня кнопка «Excellent» яка завершує тренування.

Кожна з кнопок встановлює свій раунд. Таким чином при натисканні на першу, встановлюється другий раунд і ця кнопка скривається. При натисканні на другу кнопку встановлюється тертій раунд і так далі. Після натискання на них відкривається вікно Push Up Instruction.

При натисканні на «Excellent» раунди починається з початку, а день становиться наступним.

Так все працює не тільки для віджимань, а й для присідань і підйому ніг, але написи і значення різні. Також для віджимань відкривається вікно Squat Instruction, а для підйому ніг Leg Raise Instruction.

Реалізація Push Up Instruction

Йде відстеження натискання на кнопку, після якої запускається вікно відстеження позицій тіла Pose Detect.

Реалізація Squat Instruction

Аналогічно Push Up Instruction, тільки в XML файлі інша інструкція

Реалізація Leg Raise Instruction

Аналогічно Push Up Instruction, тільки в XML файлі інша інструкція

Реалізація Pose Detect

При запуску завантажується бібліотека native, вичислюється поточна вправа для виконання та створюються змінні для показу скелетону MoveNet. Потім йде запит на можливість використання камери і внутрішнього сховища. Якщо є дозвіл, то запускається метод запуску камери. При включені камери також запускається відстежування поз людини через клас Camera Source.

Йде очікування на натискання кнопки Done. При натисканні, якщо цей екран був викликаний для того щоб дізнатись максимум повторень, то вказується

це максимальне число, флаг на дізнавання максимуму становиться false, а раунд для занять стає першим та викликається домашня сторінка.

Якщо дізнаватись максимум вправ не потрібно було, а це частина тренувань, то програма дізнається який зараз раунд, і для цього раунду присвоюється отримане значення для поточної вправи, також раунд збільшується. Після цього йде перехід на екран Training Page.

Також після натискання кнопки закривається відслідковування камери та завершується поточний екран.

Реалізація Push Up Classify, Squat Classify, Leg Raise classify

У пункті 2.2 описані необхідні кути та логіка. Дані класи реалізовані з тією логікою

Спочатку створюються змінні з пороговими кутами та змінні isUp ті лічильник підходів.

Далі через Pose Estimator у персони дізнаються кути між суглобами. Які кути та суглоби для яких вправ також описано у пункті 2.2.

Потім йде постійне порівняння цих кутів з пороговими значеннями та між собою. Якщо значення кута збільшується, то isUp = true, інакше – false. Також коли isUp = true та значення кута пересікає порогове, то лічильник збільшується на один.

4.3 Керівництво користувача

Огляд програмного забезпечення починається з першої сторінки. Першим йде привітання з написом, що кожен може тренуватись. Після натискання «Get Started» йде перехід на екран логіну. Там можна ввести свої дані та увійти у аккаунт (рис 4.3).

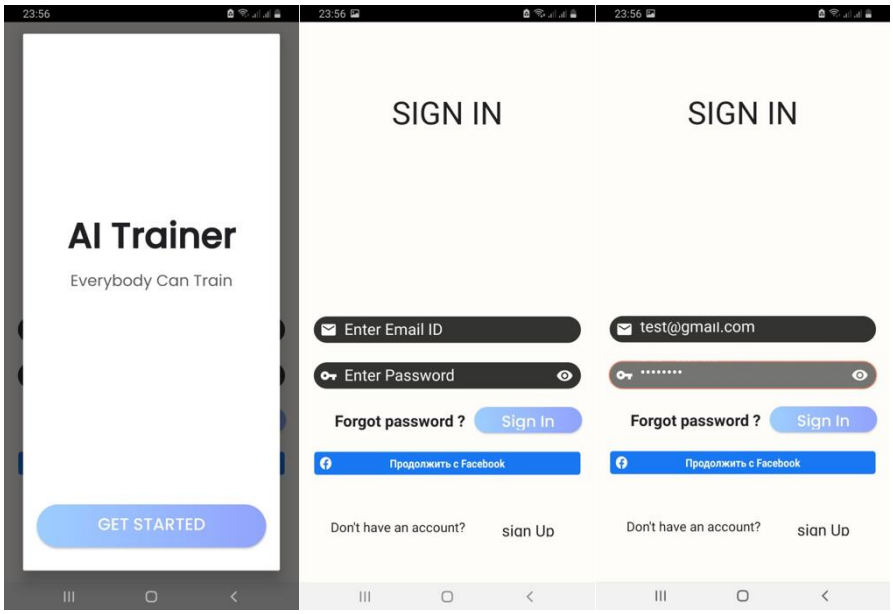

Рисунок 4.3 – Екрани початку та логіну

Якщо користувач заходить вперше, то, скоріш за все, йому потрібно буде пройти етап реєстрації. Для цього потрібно натиснути кнопку «Sign Up», після чого відкриється однойменна сторінка. На ній потрібно ввести ім’я, стать, вік, пошта, та пароль (рис. 4.4).

Але це не єдиний спосіб реєстрації. Також є можливість увійти через аккаунт Facebook (рис. 4.4).

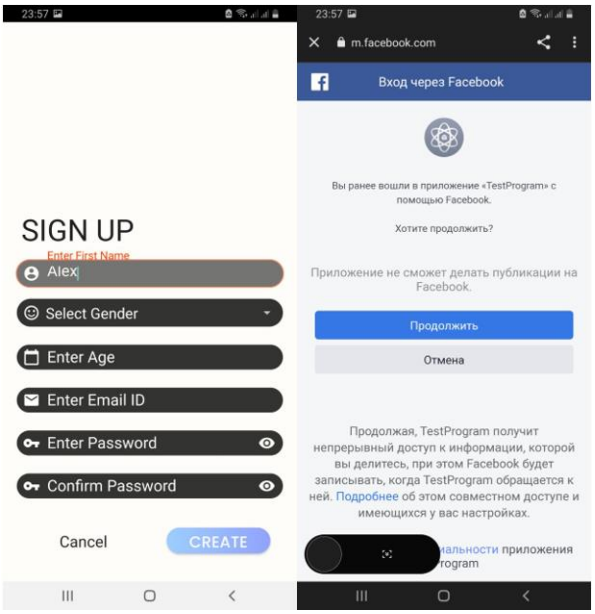

Рисунок 4.4 – Екрани реєстрації та входу через Facebook

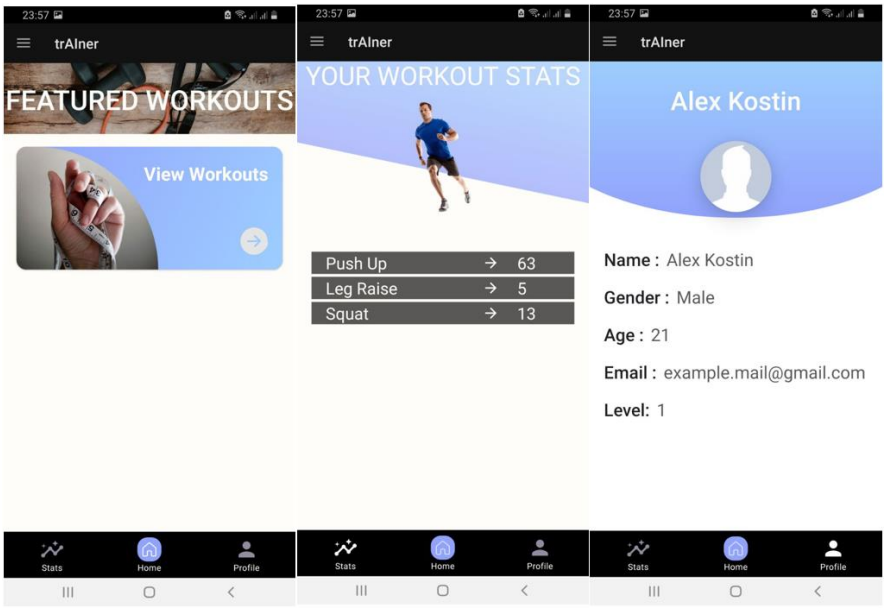

Рисунок 4.5 – Основні екрани: домашня сторінка, статистика та профіль користувача

Після входу у аккаунт користувач потрапляє на домашню сторінку, та може переключатись між основними екранами (рис. 4.5)

На домашній сторінці він може почати тренування. На екрані статистики видно скільки користувач виконав вправ за весь час користування застосунком. На сторінці профілю видно дані що вводились при реєстрації, а саме: ім’я користувача, стать, вік та пошта. Також тут видно рівень користувача. Як він знаходиться буде описано у керівництві користувача.

Наступна сторінка – це бокове меню. В нього можна зайти через кнопку гамбургера зверху зліва. На екрані є ім’я користувача, кнопка виходу із аккаунту та кнопка “Feedback” (рис. 4.6). Якщо на неї натиснути то буде перехід на інший екран, де можна поставити кількість зірочок у Google Play. Але застосунок не завантажений у магазин, тому кнопка не працює.

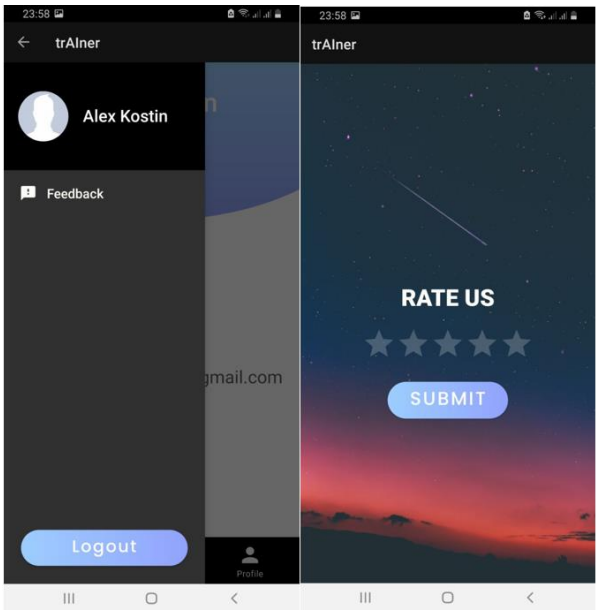

Рисунок 4.6 – Сторінка меню та фідбек

Наступна сторінка це сторінка з самими тренуваннями. На неї можна потрапити з домашньої сторінки. Тут пропонується обрати тип занять: віджимання, присідання чи підняття ніг. В цілому візуально у них однакові екрани, різниця в кількості підходів, малюнках на інструкції та в алгоритмі відслідковування поз.

Для прикладу обрані віджимання. При першому заході, чи після завершення минулого плану тренувань буде пропозиція зафіксувати скільки максимум підходів може зробити користувач. Після чого йому буде відображена інструкція, та після буде перехід до екрану з відслідковуванням поз. Потрібно докласти зусиль та зробити максимальну кількість підходів (рис. 4.7). На цьому перше тренування буде завершено.

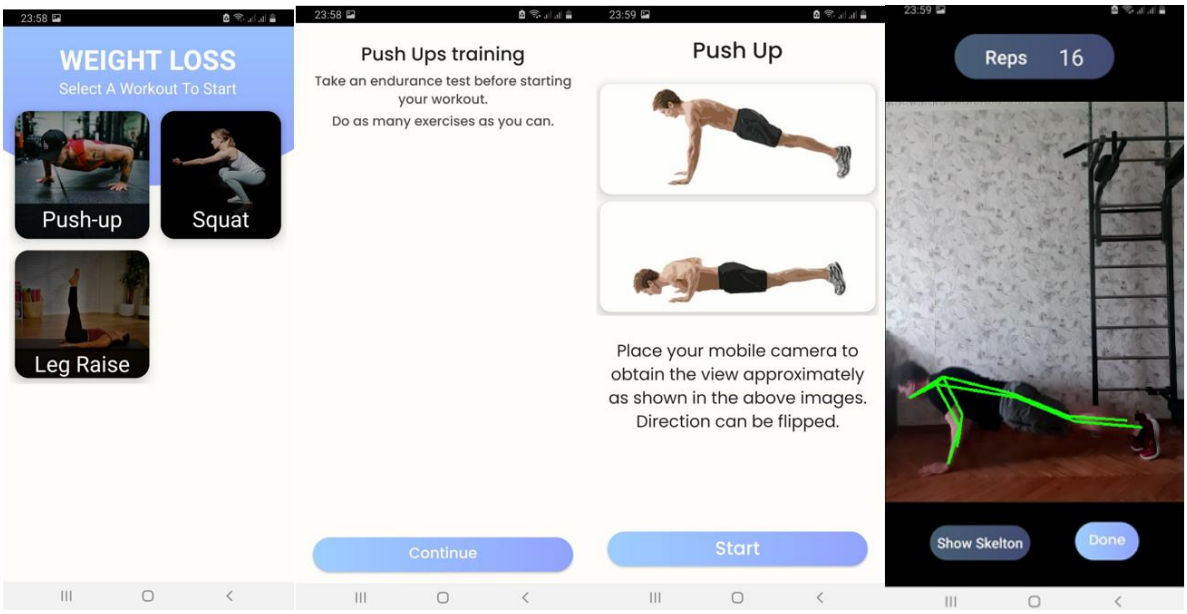

Рисунок 4.7 – Екрани вибору тренування, пропозиції зробити максимум вправ, інструкції та екрану відстеження рухів тіла

При наступному вході на сторінку тренувань вже буде розроблений індивідуальний графік. Описано скільки робити вправ, та скільки відпочивати. Достатньо натиснути кнопку старту, та вам знов з’явиться інструкція яка приведе до екрану відслідковування рухів. Потів ваш результат буде записаний. Після виконання всієї програми вас привітають і тренування на цей день буде завершене (рис 4.8).

Далі потрібно завершити всі три дні, та знов зафіксувати свій максимальний результат. Таким чином ми завжди будете поліпшувати свої результати і ставати краще.

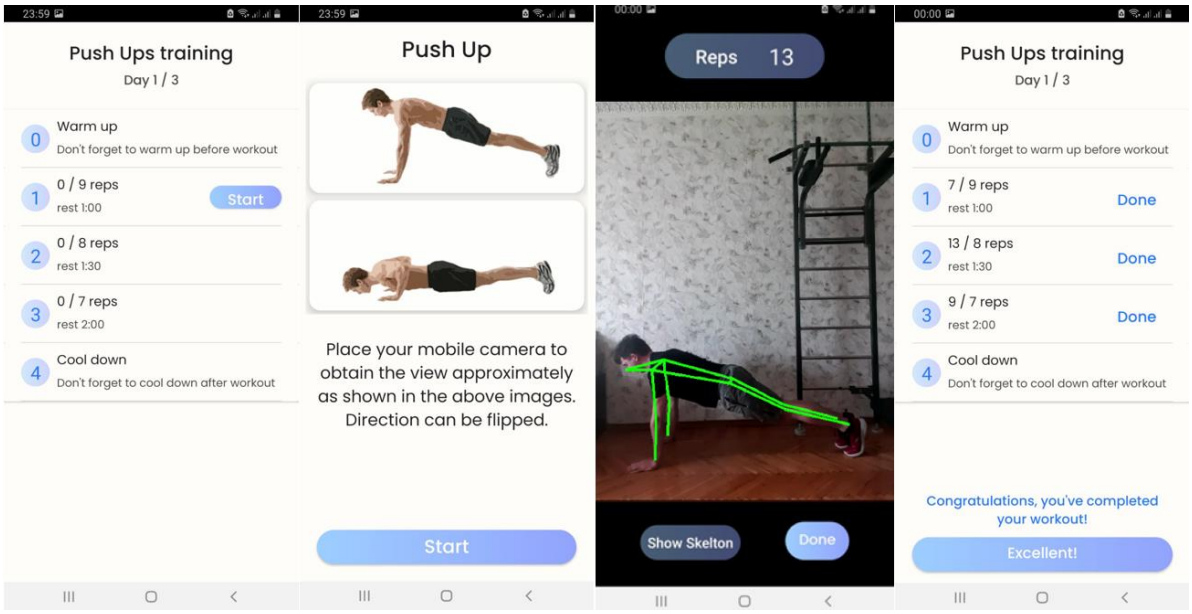

Рисунок 4.8 – Сторінка програми тренувань для віджимань, сторінка інструкцій, сторінка відстеження рухів та сторінка успішного завершення програми

Висновки до розділу 4

Були розглянуті технології та програм для розробки та обрані ті, які більше всього підходять для реалізації запланованого застосунку.

Був проведений опис програмної реалізації. Кожен клас був обстежений, та описані принципи його роботи.

Було зроблене керівництво користувача, де описані можливості створеного застосунку. Як зареєструватись та увійти у свій аккаунт. Описане навігаційне меню програми, сторінку статистики, домашню сторінку і сторінку користувача. Розглянуте меню-бургер та фідбек. Розглянуто сторінку з тренуванням, створенням індивідуального графіку та відстеженням положення тіла під час тренувань користувача.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Спеціальний розділ

ОХОРОНА ПРАЦІ

до кваліфікаційної роботи

на тему:

«Методи ефективного контролю індивідуального графіку занять спортом з ігровою системою мотивації»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810313

Виконав студент 4-го курсу, групи 402
_____О. А. Костін_____
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Консультант_____ *канд. наук, доцент* __
(наук. ступінь, вчене звання)
_____А. О. Алексєєва____
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Миколаїв – 2022

5 ОХОРОНА ПРАЦІ

5.1 Шкідливі фактори роботи за комп'ютером

Сьогодні неможливо уявити роботу офісного працівника без комп'ютера. Цей електронний помічник міцно укорінився в трудовій діяльності [17]. Але усім знайомі викликані ним неприємні відчуття після напруженого робочого дня. Про те, що дотримуватися їх зобов'язані всі роботодавці, періодично нагадує Держпраці. Сьогодні і ми нагадаємо про основні правила роботи за комп'ютером.

Екранні пристрої - це електронні засоби для відтворення будь-якої графічної або алфавітно-цифрової інформації на основі електронно-променевої трубки, рідкокристалічні, плазмові, проекційні, органічні світлодіодні монітори та інші новітні розробки у сфері інформаційних технологій [17].

Якщо ваші працівники використовують комп'ютери, то ви як роботодавець у будь-якому випадку повинні дотримуватися вимог [18].

У п. 4 розд. І Вимог № 207 [18], визначаються випадки, коли ці Вимоги не застосовуються. У ньому сказано, що ці вимоги не поширюються, зокрема, на портативні системи обробки даних, але за умови, що вони не постійно використовуються на робочому місці.

Далі розглянуто ноутбук. Його можна віднести до портативної системи обробки даних. Сучасні ноутбуки бувають різними. Деякі моделі нічим не поступаються стаціонарним комп'ютерам, а деякі за своїми «екранними» параметрами ближче до планшетів. Але все-таки роботодавцеві безпечніше орієнтуватися на більш жорсткі вимоги [18]. І тут вирішальним буде питання, чи постійно ноутбук використовується на робочому місці. Якщо ні, то з-під регулювання його використання випадає.

Щодо визначення постійного використання. У Державних санітарних правилах зазначено, що робота з комп'ютером вважається основною, якщо вона забирає не менше 50 % часу протягом робочої зміни.

Наприклад, якщо робочий день для працівника становить 8 годин, то щоб вимоги на нього не поширювалися, він повинен працювати з ноутбуком менше 4 годин щоденно [19].

Вимоги при роботі зі стаціонарними комп'ютерами потрібно обов'язково незалежно від часу, який працівники проводять за комп'ютером протягом робочого дня.

Роботодавець повинен проінформувати працівників під розписку про умови праці та наявність на їх робочих місцях небезпечних і шкідливих факторів (фізичних, хімічних, біологічних, психофізіологічних), які виникають під час роботи з комп'ютером і ще не усунені, а також про можливі наслідки їх впливу на здоров'я працівників [20].

Якщо все-таки небезпечні та/або шкідливі фактори на робочому місці присутні, роботодавець повинен повідомити про них працівникові під розписку.

Також роботодавець повинен забезпечити навчання і перевірку знань працівників з питань охорони праці та безпечного використання екранних пристроїв до початку роботи з ними, а також у випадках модифікації й організації роботи устаткування [21].

Він повинен вживати заходів, щоб забезпечити відповідність робочого місця працівника вимогам, зокрема до організації робочого місця працівника, мікроклімату приміщень тощо;

За рахунок тривалості робочої зміни організувати внутрішні регламентовані перерви для відпочинку відповідно до ДСанПіН 3.3.2.007-98.

За необхідності проводити лабораторні дослідження умов праці працівників з метою виявити шкідливі та небезпечні фактори виробничого середовища, тяжкості й напруженості трудового процесу (зокрема, виявити ризики, пов'язані з погіршенням зору, порушенням фізичного стану, стресом), а також вживати заходів щодо усунення виявлених ризиків [20].

Ще є обов’язок роботодавця за свій рахунок проводити медогляди працівників і за їх результатами за необхідності виконувати відповідні оздоровчі заходи.

Також усі користувачі персональних комп’ютерів підлягають обов’язковим медоглядам раз на рік комісією у складі невропатолога, офтальмолога й отоларинголога.

Порушення законодавства про охорону праці може спричинити накладення штрафу на роботодавця. Максимальний розмір такого штрафу не може перевищувати 5 % середньомісячного фонду зарплати за попередній рік.

5.2 Вимоги до приміщення з використанням екранних пристроїв

Приміщення також відіграє важливу роль у роботі. Тому держава розробила норми вимог [18] щодо них.

Вони описують, що робочі місця працівників з екранними пристроями повинні бути спроектовані та мати такі розміри, щоб працівники могли змінювати положення та вільно рухатись [19].

Крім того, з метою забезпечення безпеки та здоров'я працівників необхідно зробити щоб усе випромінювання від екранних пристроїв має бути зведене до гранично допустимого рівня (вплив на людину факторів зовнішнього середовища [22] – шуму, вібрації, забруднення, температури тощо) з огляду безпеки та здоров'я працюючих.

Крім того, організація робочого місця працівника з пристроями контролю має забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним, антропологічним, психофізіологічним вимогам а також відповідати роду виконуваної роботи [21].

Освітлення робочого місця працівника екранними пристроями також має створювати відповідний контраст між екраном та навколишнім середовищем (з урахуванням виду роботи) та відповідати вимогам ДСанПІН 3.3.2.007-98.

Слід пам'ятати, що мікроклімат виробничих приміщень з робочими місцями, обладнаними пристроями відображення повинен підтримуватися на постійному рівні та відповідати вимогам норм гігієнічного мікроклімату для виробничих приміщень ДСН 3.3.6.042-99.

Крім того, стіл або робоча поверхня повинні бути відповідного розміру і мати поверхню з низькою відбивною здатністю, що дозволяє гнучко розмістити монітор, клавіатуру, документи і супутні пристрої [18].

Слід пам'ятати, що робоче крісло має бути стійким і дозволяти працівнику з екранними пристроями вільно рухатися та приймати зручне положення. Сидіння має регулюватися по висоті, спинка сидіння як по висоті, так і по нахилу. Слід передбачати підніжку для тих, кому це необхідно для зручності.

Слід передбачати підніжку для тих, кому це необхідно для зручності [18].

5.3 Вимоги до роботи із комп’ютерними пристроями

Існують мінімальні вимоги безпеки під час роботи з екранними пристроями [23]:

1. Щодня перед початком роботи необхідно очищати екранні пристрої від пилу та інших забруднень.

2. Після закінчення роботи екранні пристрої слід відключати від електричної мережі.

3. У разі виникнення аварійної ситуації необхідно негайно відключити екранний пристрій від електричної мережі.

4. Не допускається:

– виконувати технічне обслуговування, ремонт і налагодження екранних пристроїв безпосередньо на робочому місці працівника під час роботи з екранними пристроями;

– відключати захисні пристрої, самочинно проводити зміни у конструкції та складі екранних пристроїв або їх технічне налагодження;

– працювати з екранними пристроями, у яких під час роботи виникають нехарактерні сигнали, нестабільне зображення на екрані та інші несправності [23].

5. Під час виконання робіт операторського типу, пов’язаних з нервово-емоційним напруженням, у приміщеннях під час роботи з екранними пристроями, на пультах і постах керування технологічними процесами та в інших приміщеннях мають дотримуватися оптимальні умови мікроклімату відповідно до вимог ДСН 3.3.6.042-99.

Також є мінімальні вимоги безпеки до екранних пристроїв [18]:

1. Екранні пристрої не мають бути джерелом ризику для працівників.

2. Усе випромінювання, за винятком видимої частини електромагнітного спектра, має бути зведене до незначного рівня з погляду безпеки і охорони здоров’я працівників.

3. Символи на екранних пристроях мають бути чіткими, відповідного розміру. Між символами і рядками символів має бути належна відстань.

4. Зображення на екрані має бути стабільним, без миготінь або інших видів нестабільності.

5. Яскравість та/або контрастність символів має легко регулюватися працівником під час роботи з екранними пристроями, а також швидко адаптуватися до навколишніх умов.

6. Вибираючи екрани, слід надавати перевагу таким екранам, які легко та вільно повертаються і нахиляються відповідно до потреби працівника.

7. За необхідності може використовуватись окрема підставка або регульований стіл для розміщення екрана.

8. Екран не має відблискувати або відбивати світло, щоб не викликати дискомфорту у працівника під час роботи з екранними пристроями [18].

9. Вибираючи клавіатуру, слід надавати перевагу такій клавіатурі, яка відкидається і є автономною (відокремленою від екрана), щоб працівник міг

вибрати зручну робочу позу й уникнути втоми рук (кисті і верхньої частини руки).

10. Поверхня клавіатури має бути матовою, щоб уникнути віддзеркалювання. Розташування клавіш і самі клавіші мають полегшувати роботу із клавіатурою. Позначення клавіш повинно бути достатньо контрастним і розбірливим.

11. Устаткування, яке входить до робочої станції, не має виділяти надлишкового тепла, що може спричинити незручності працівникам під час роботи з екранними пристроями [18].

12. Під час розробки, вибору, замовлення та модифікації програмного забезпечення, а також під час розробки завдань, що передбачають використання устаткування з екранними пристроями, роботодавець має керуватися таким програмним забезпеченням, яке відповідає розв’язуваним завданням і є простим у використанні, а де необхідно - адаптованим до рівня знань і досвіду працівника [23].

5.4 Виробничий шум та вібрації

Шум як несприятливий фактор робочого середовища помітний на більшості промислових підприємств, транспорті та сільському господарстві. Джерелами шуму можуть бути системи вентиляції та кондиціонування повітря, аерогазодинамічні установки, системи охолодження комп'ютерів тощо [24].

Інтенсивний промисловий шум може викликати професійні захворювання, такі як втрата слуху або глухота. Крім того, працівники, які щодня перебувають під його впливом мають такі проблеми:

1) падає ефективність роботи;

2) увага послаблюється і реакція сповільнюється, запаморочення, дратівливість, працездатність, погіршується гострота зору;

3) підвищується артеріальний тиск, змінюється ритм дихання і серцева діяльність, порушується працездатність клітин кори головного мозку та ін.

Нормальний фоновий шум людини з рівнем звукового тиску на частотах 15-35 дБ. При підвищенні рівня звукового тиску до 40-70 дБ спостерігається незначне зниження працездатності та погіршення самопочуття (гучна музика, шум від технічних засобів тощо) [24].

Рівень звукового тиску в діапазоні 75-120 дБ вражає органи слуху і кровоносну систему. Постійний шум з рівнем звукового тиску вище 120 дБ може призвести до акустичної травми (значної втрати слуху). Прояв патологічних змін в організмі, що викликав шум, залежить від його параметрів (інтенсивності та частотного складу), стажу роботи, тривалості діяльності протягом робочого дня, індивідуальної чутливості організму та поєднання з іншими професійними факторами.

Засоби колективного захисту від шуму. Вирішення проблеми захисту від шуму досягається шляхом проведення комплексу заходів, спрямованих на зниження інтенсивності шкідливих виробничих факторів у їх джерелах або шляхом поширення звукового тиску [24].

Зниження шуму при його поширенні досягається насамперед за рахунок архітектурного планування та акустичних заходів колективного шумозахисту. Архітектурно-планувальна діяльність включає:

1) раціональне розміщення будівель і споруд на території підприємства (здійснюється під час проектування, реконструкції та експлуатації підприємств, цехів, ділянок);

2) раціональне розташування технологічних пристроїв і робочих місць;

3) оптимальний акустичний розподіл зон і режимів руху транспортних засобів і потоків;

4) створення шумозахисних зон.

Акустичні заходи передбачають застосування звукоізоляції, звукопоглинання, віброізоляції, гасіння (гасіння коливань механічних систем з нелінійними динамічними пристроями) і застосування шумогасників. На практиці також широко застосовуються організаційно-технічні заходи

колективного захисту від шуму, такі як оснащення приладів дистанційним керуванням, використання малошумних технологічних процесів і пристроїв, дотримання правил технічної експлуатації приладів, проведення планових профілактичних оглядів та ремонти тощо.

Устаткування для захисту від шуму. Якщо колективні заходи захисту не знижують рівень шуму на робочому місці до прийнятного рівня, застосовуються індивідуальні шумозахисту. Вони дозволяють перекрити найбільш чутливий канал для проникнення звуку в організм через вуха і не дають порушити нервову систему таким інтенсивним подразником, як шум.

З цією метою використовують протишуми, або антифони, які поділяються на 3 види:

1) внутрішнього використання – втулки, вкладки, тампони;
2) зовнішнього використання – навушники, шоломи, костюми;
3) змішані, які вставляються при вході в слуховий прохід.

Як правило, вибір засобів індивідуального захисту залежить від виду та особливостей шуму на робочому місці, зручності його використання на конкретній роботі, кліматичних умов та інших факторів. Зокрема, допускається використання вкладок при рівні звуку не вище 100 дБ, навушників - 110 дБ, шоломів - 120 дБ. При рівні шуму вище 120 дБ, крім шоломів, рекомендується носити шумозахисні костюми, ремінь і взуття.

Чому промислові вібрації небезпечні Промислові вібрації можуть завдати серйозної шкоди здоров'ю людини [25], зокрема, спричинити струс мозку, серцеву недостатність, нервові та судинні розлади, судоми перевтоми та призвести до вібраційної хвороби.

Найнебезпечніші для людини частоти вібрації знаходяться в діапазоні 6-9 Гц, оскільки вони збігаються з природною частотою коливань внутрішніх органів людини [25]. З метою профілактики вібраційних захворювань при прийомі на роботу проводиться первинний медичний відбір.

Люди з початковими симптомами вібраційної хвороби ретельно виявляються під час періодичних медичних оглядів і своєчасно лікуються.

5.5 Режим праці та відпочинку на підприємствах з комп'ютерними пристроями

Внутрішньозмінні режими праці і відпочинку при роботі з ПК розроблено з урахуванням характеру трудової діяльності, напруженості і важкості праці диференційовано до кожної професії [19].

За характером трудової діяльності виділено три професійні групи згідно з класифікатором професій (ДК-003-95 і Зміна № 1 до ДК-003-95):

1) розробники програм (інженери-програмісти) - виконують роботу переважно з ПК та документацією. При цьому відбувається інтенсивний обмін інформацією з ПК і висока частота прийняття рішень. Робота виконується у вільному темпі і пов'язана з періодичним пошуком помилок в умовах дефіциту часу, характеризується інтенсивною розумовою творчою працею з підвищеним напруженням зору, концентрацією уваги, нервово-емоційним напруженням, статичною робочою позою, періодичним навантаженням на кисті верхніх кінцівок.

2) оператори електронно-обчислювальних машин – виконують роботу, яка пов'язана з обліком інформації, одержаної з ПК, супроводжується перервами різної тривалості, пов'язана з виконанням іншої роботи і характеризується як робота з напруженням зору, невеликими фізичними, зусиллями, нервовим напруженням середнього ступеня та виконується у вільному темпі [26];

3) оператор комп'ютерного набору - виконує одноманітні за характером роботи з документацією та клавіатурою і нечастими нетривалими переключеннями погляду на екран монітора, з введенням даних з високою швидкістю, робота характеризується як фізична праця з підвищеним навантаженням на кисті верхніх кінцівок, з напруженням зору (фіксація зору переважно на документи), нервово-емоційним напруженням [21].

Встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ПК при 8-годинній денній робочій зміні залежно від характеру праці [23]:

1) для розробників програм із застосуванням ПК слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи;

2) для операторів із застосуванням ПК слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;

3) для операторів комп'ютерного набору слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожної години роботи за ПК.

У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи з ПК не повинна перевищувати 4 години.

При 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин роботи аналогічно перервам при 8-годинній робочій зміні, а протягом останніх 4 годин роботи, незалежно від характеру трудової діяльності, через кожну годину тривалістю 15 хвилин.

З метою зменшення негативного впливу монотонності на працюючого слід чергувати деякі операції, наприклад, введення тексту за допомогою клавіатури та редагування тексту тощо. Для зниження нервово-емоційного напруження, втоми зорового аналізатора, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі доцільно деякі перерви використовувати для виконання комплексу вправ.

В окремих випадках - при постійних скаргах працюючих з ПК на зорову втому, незважаючи на дотримання санітарно-гігієнічних вимог до режимів праці і відпочинку, а також застосування засобів локального захисту очей - допускаються індивідуальний підхід до обмеження часу робіт з ПК, зміни характеру праці, чергування з іншими видами діяльності, не пов'язаними з ПК.

Активний відпочинок має полягати у виконанні комплексу гімнастичних вправ, спрямованих на зняття нервового напруження, м'язове розслаблення, відновлення функцій фізіологічних систем, що порушуються протягом трудового процесу, зняття втоми очей, поліпшення мозкового кровообігу і працездатності. За умови високого рівня напруженості робіт з ПК необхідне психологічне розвантаження у спеціально обладнаних приміщеннях (в кімнатах психологічного розвантаження) під час регламентованих перерв або в кінці робочого дня.

Висновки до розділу 5

Під час виконання спеціальної частини з охорони праці було проведено аналіз шкідливих факторів при роботі з екранними пристроями. Стало зрозуміло, що якщо наявні на робочих місцях небезпечні і шкідливі фактори (фізичні, хімічні, біологічні, психофізіологічні), то роботодавець повинен у першу чергу їх виправити, та повідомити про них працівникові.

Під час роботи важливо щоб приміщення де працює люди з комп’ютером було у гарному стані. Тому були розглянуті умови які повинні бути дотримані у приміщенні, де знаходяться комп’ютерні пристрої. Також розглянуті вимоги до роботи з персональним комп’ютером, щоб вони працювали довго, і щоб робітникам було зручно і безпечно.

Розглянуто проблеми шуму та вібрацій на підприємствах. Знайдені заходи які сприяють покращенню стану працівників, та запобіганню збільшення захворюваності. Якщо приміщення знаходиться поблизу значного шуму, це може викликати серйозні проблеми у працівників офісу.

Знайдено оптимальні режими праці на відпочинку серед працівників, що працюють на підприємствах з комп’ютерними пристроями, в залежності від того, який клас робіт вони виконують.

Під час виконання спеціальної частини з охорони праці було проведено аналіз чинних нормативних документів, законодавчих та нормативно-правових актів.

Розглянуті нормативні документи є досить актуальними, оскільки зараз процес автоматизації та модернізації підприємств прискорюється, розвивається, а комп’ютерні системи найчастіше замінюють старе обладнання, що змушує врегулювати вимоги до роботи з новітнім обладнанням, створювати необхідні умови для комфортної та ефективної роботи, та безпеці праці людей.

ВИСНОВКИ

Під час виконання бакалаврської кваліфікаційної роботи було розроблено мобільний застосунок для ефективного контролю індивідуального графіку занять спортом з ігровою системою мотивації на базі android.

Був проведений аналіз існуючих технологій тренувань та засобів мотивації користувача. Розглянуто типи фізичних тренувань, як саме розробляти програму та що слід врахувати. Також проведено аналіз того, що таке ігрова мотивація та геймифікація.

Розглянуті існуючі варіанти вже розроблених застосунків, які пропонують схожі функції. На основі проведеного аналізу створено вимоги та чітке завдання щодо власного застосунку.

Були розглянуті базові фізичні вправи та досліджено як їх правильно робити. Розглянуті основні існуючі технології відстеження положення тіла та обрано технологію MoveNet, яка показувала найшвидший результат роботи – 78 кадрів у секунду. Розглянуто як зробити алгоритм класифікації вправ і зарахування балів за їх виконання. Був розроблений індивідуальний графік занять спортом на основі навантаження 50-70% від максимуму повторень.

Розроблено прототип та дизайн для майбутньої програми. Для створення архітектори програмного забезпечення розроблено use-case UML діаграму та діаграму класів, де прописані всі класи, їх дані, методи, властивості за залежності.

Обрані технології та програми для розробки, які підходять для реалізації запланованого застосунку. Далі застосунок був розроблений та створено керівництво користувача.

В результаті роботи отриманий застосунок, який сворює індивідуальний графік тренувань для людини, і може за допомогою штучного інтелекту відстежувати виконання вправ користувачем. Також в застосунку присутня ігрова система мотивації, щоб людина не закінчували свої тренування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Костюкевич В. М. Теорія і методика тренування спортсменів високої кваліфікації : Навчальний посібник. Вінниця: «Планер», 2007. 273 с.

2. Сучасні програми оздоровчого фітнесу : вебсайт. URL: http://enpuir.npu.edu.ua/bitstream/handle/123456789/18980/Volovik%20N.pdf?sequence=1 (дата звернення: 24.05.2022)

3. Realtime Indoor Workout Analysis Using Machine Learning & Computer Vision : вебсайт. URL: https://ieeexplore.ieee.org/abstract/document/8856547/figures#figures (дата звернення: 23.05.2022)

4. Програма тренувань вдома: основні вправи для домашніх занять спортом. UA-Футбол : вебсайт. URL: https://www.ua-football.com/ua/sport-i-zdorovie/siz-news/1619174555-programa-trenuvan-vdoma-osnovni-vpravi-dlya-domashnih-zanyat-sportom.html?%2Fua%2Fsport-i-zdorovie%2Fsiz-news%2F1619174555-programma-trenirovok-doma-osnovnye-uprazhneniya-dlya-domashnih-trenirovok_html= (дата звернення: 26.05.2022)

5. Masaaki Kurosu. Human-Computer Interaction: Interaction Technologies: Наукова праця. Лос-Анджелес, 2015. – 73 с.

6. Munson, S.A., Consolvo, S. Exploring goal-setting, rewards, self-monitoring, and sharing to motivate physical activity : Доповідь конференції. Сан-Дієго, 2012. 9 с.

7. Just Used Machine Learning in My Workout! Towards Data Science : вебсайт. URL: https://towardsdatascience.com/just-used-machine-learning-in-my-workout-ff079b8e1939 (дата звернення: 27.05.2022)

8. Нестерова С.Ю. Основи здорового способу життя : Навчально-методичний посібник. Вінниця, 2019. 142 с.

9. Gerard Pons-Moll, Bodo Rosenhahn. Model-Based Pose Estimation : Книга. Лондон, 2011. 170 с.

10. Katia Bourahmoune, Toshiyuki Amagasa. Fitness devices and artificial intelligence trainers are one of the most common uses of body posture assessments. Цукуба, 2019. С. 5808–5814.

11. Models - Explore pre-trained TensorFlow.js models that can be used in any project out of the box. TensorFlow : вебсайт. URL: https://www.tensorflow.org/js/models (дата звернення: 23.05.2022)

12. Міністерство Внутрішніх Справ України, Національна Академія Внутрішніх Справ. Розвиток фізичних якостей здобувачів вищої освіти ЗВО МВС України : Методичні рекомендації. Київ, 2021. 109 с.

13. FITNESS AND TRAINING CONCEPTS : вебсайт. URL: https://www.lths.net/cms/lib/IL01904810/Centricity/domain/165/units/Fitness%20and%20Training%20Concepts.pdf (дата звернення: 25.05.2022)

14. Griffin, John. Client-Centered Exercise Prescription 3E : Книга. Торонто, 2014. 496 с.

15. Tony Clark, Andy Evans. Foundations of the Unified Modeling Language, Бредфорд, 1999. 14 с.

16. Harald Störrle, Alexander Knapp. Unified Modeling Language 2.0, Мюнхен, 2006. 244 с.

17. Зеркалов Д.В. Охорона праці в галузі: Загальні вимоги: навч. посіб. Київ : "Основа", 2011. 551 с.

18. Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями : Наказ від 14 лют. 2018 р. № 207. URL: https://zakon.rada.gov.ua/laws/show/z0508-18 (дата звернення: 21.05.2022).

19. В.І. Голінько, М.Ю. Іконніков, Я.Я. Лебедєв. Охорона праці в галузі інформаційних технологій : навч. посіб. Дніпро, 2015. 247 с.

20. Робота за комп’ютером: дотримуйтесь правил!. *Factor* : вебсайт. URL: https://i.factor.ua/journals/ot/2019/march/issue-6/1/article-43512.html (дата звернення: 24.05.2022).

21. Катренко Л. А., Катренко А. В. Охорона праці в галузі комп'ютинґу : підручник. Львів: "Магнолія 2006", 2012. 544 с.

22. Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99 : Постанова від 1 грудня 1999 р. №42. URL: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення: 23.05.2022).

23. Про охорону праці : Закон України від 14 жов. 1992 р. за № 49. URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 22.05.2022).

24. Санітарні норми виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037-99 : Постанова від 1 грудня 1999 р. №37. URL: https://zakon.rada.gov.ua/rada/show/va037282-99 (дата звернення: 23.05.2022).

25. Державні санітарні норми виробничої загальної та локальної вібрації ДСН 3.3.6.039-99 : Постанова від 1 грудня 1999 р. №39. URL: https://zakon.rada.gov.ua/rada/show/va039282-99 (дата звернення: 23.05.2022).

26. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98 : Постанова від 10 грудня 1998 р. №7. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98 (дата звернення: 22.05.2022).

ДОДАТОК А

КОД ПРОГРАМНОГО ЗАСТОСУНКУ

Код класу SignIn.

```
package com.dedsec_x47.trainer.auth

import android.app.AlertDialog
import android.content.Intent
import android.os.Bundle
import android.text.TextUtils
import android.util.Log
import android.view.LayoutInflater
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.dedsec_x47.trainer.R
import com.dedsec_x47.trainer.databinding.ActivitySigninBinding
import com.facebook.AccessToken
import com.facebook.CallbackManager
import com.facebook.FacebookCallback
import com.facebook.FacebookException
import com.facebook.login.LoginManager
import com.facebook.login.LoginResult
import com.google.android.gms.tasks.OnCompleteListener
import com.google.firebase.auth.FacebookAuthProvider
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.firestore.SetOptions
import com.google.firebase.firestore.ktx.firestore
import com.google.firebase.ktx.Firebase
import com.google.firebase.messaging.FirebaseMessaging


class SignIn : AppCompatActivity() {

    private lateinit var auth: FirebaseAuth
    private lateinit var callbackManager: CallbackManager
    private lateinit var activitySignInBinding: ActivitySigninBinding

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)
        if (isgetStart) {
            getStarted()
        }

        activitySignInBinding = ActivitySigninBinding.inflate(layoutInflater)
        setContentView(activitySignInBinding.root)
        auth = Firebase.auth
        callbackManager = CallbackManager.Factory.create()

        activitySignInBinding.btnSignIn.setOnClickListener {

            val email =
activitySignInBinding.textInputEditTextEmail.text.toString()
            val password =
```

```
activitySignInBinding.textInputEditTextPassword.text.toString()

            if (TextUtils.isEmpty(email) || TextUtils.isEmpty(password)) {
                Toast.makeText(
                    baseContext, "Please Enter Your email and Password",
                    Toast.LENGTH_SHORT
                ).show()
            } else {
                emailloginaccount(email, password)
            }
        }

        activitySignInBinding.btnFacebook.setReadPermissions("public_profile",
"email")
        activitySignInBinding.btnFacebook.setOnClickListener {
            fbSignIn()
        }


        activitySignInBinding.btnSignUp.setOnClickListener {
            gotoSignUpPage()
        }
    }

    private fun getStarted() {

        val dialogView =
LayoutInflater.from(this).inflate(R.layout.activity_starting, null)
        val builder = AlertDialog.Builder(this).setView(dialogView)
        val alertDialog = builder.show()

        val btngetSt = dialogView.findViewById<Button>(R.id.btnGetStarted)

        btngetSt.setOnClickListener {
            gotoLoginPage()
            isgetStart = false
            alertDialog.dismiss()
        }
    }

    private fun emailloginaccount(email: String, password: String) {
        auth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {
                    Log.d(TAG, "signInWithEmail:success")
                    getCurrentRegistrationToken(email)
                } else {
                    Log.w(TAG, "signInWithEmail:failure", task.exception)
                    Toast.makeText(
                        baseContext, "SignIn failed.",
                        Toast.LENGTH_SHORT
                    ).show()
                }
            }
    }

    private fun getCurrentRegistrationToken(email: String) {

FirebaseMessaging.getInstance().token.addOnCompleteListener(OnCompleteListener {
task ->
            if (!task.isSuccessful) {
```

```
                Log.w(TAG, "Fetching FCM registration token failed",
task.exception)
                return@OnCompleteListener
            }
            if (task.result != null) {
                Log.w("TOK", "Token not null" + task.result)
                fcm = task.result
                uploadToken(email)
            } else {
                Log.w("TOK", "Token null")
                gotoLoginPage()
            }
        })
    }

    private fun uploadToken(email: String) {
        val userData: MutableMap<String, Any> = HashMap()
        userData["Registration Token"] = fcm
        Firebase.firestore.collection("users").document(email).set(userData,
SetOptions.merge())
            .addOnSuccessListener {
                Log.w(TAG, "update Token success")
                gotoLoginPage()
            }
    }

    private fun fbSignIn() {

        activitySignInBinding.btnFacebook.registerCallback(callbackManager,
            object : FacebookCallback<LoginResult> {

                override fun onSuccess(result: LoginResult) {
                    //isfacebookLogin = true
                    Log.d(TAG, "facebook:onSuccess:$result")
                    handleFacebookAccessToken(result.accessToken)
                    //getFacebookData(result.accessToken)
                }

                override fun onCancel() {
                    Log.d(TAG, "facebook:onCancel")
                }

                override fun onError(error: FacebookException) {
                    Log.d(TAG, "facebook:onError", error)
                }
            })
    }

    private fun gotoSignUpPage() {
        val intent = Intent(this, SignUp::class.java)
        startActivity(intent)
        finish()
    }

    private fun gotoLoginPage() {
        val intent = Intent(this, LoginActivity::class.java)
        startActivity(intent)
        finish()
    }

    override fun onActivityResult(requestCode: Int, resultCode: Int, data:
```

```
Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        callbackManager.onActivityResult(requestCode, resultCode, data)
    }

    private fun handleFacebookAccessToken(token: AccessToken?) {
        Log.d(TAG, "handleFacebookAccessToken:$token")

        val credential = FacebookAuthProvider.getCredential(token!!.token)
        auth.signInWithCredential(credential)
            .addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {
                    Log.d(TAG_FB, "signInWithCredential:success")
                    //linkWithCredential(credential)
                } else {
                    if (AccessToken.getCurrentAccessToken() != null) {
                        LoginManager.getInstance().logOut()
                    }
                    Log.w(TAG_FB, "signInWithCredential:failure",
task.exception)
                    Toast.makeText(this, "Authentication failed.",
Toast.LENGTH_SHORT).show()
                }
            }
    }

    companion object {
        private const val TAG = "EmailPassword"
        private const val TAG_FB = "FacebookLogin"
    }
}
```

Код класу SignUp.

```
package com.dedsec_x47.trainer.auth

import android.content.Intent
import android.os.Bundle
import android.text.TextUtils
import android.util.Log
import android.view.View
import android.widget.AdapterView
import android.widget.ArrayAdapter
import android.widget.Spinner
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.dedsec_x47.trainer.R
import com.dedsec_x47.trainer.databinding.ActivitySignupBinding
import com.google.firebase.auth.EmailAuthProvider
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase
import java.util.regex.Pattern

class SignUp : AppCompatActivity() {

    lateinit var auth: FirebaseAuth
    lateinit var email: String
    lateinit var password1: String
    lateinit var password2: String
```

```
    lateinit var activitySignupBinding: ActivitySignupBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        activitySignupBinding = ActivitySignupBinding.inflate(layoutInflater)
        setContentView(activitySignupBinding.root)

        auth = Firebase.auth

        val genOp = resources.getStringArray(R.array.genderDropdown)
        val genSpin = findViewById<Spinner>(R.id.spinnerGender)

        if (genSpin != null) {
            val adapter = ArrayAdapter(
                this,
                android.R.layout.simple_spinner_dropdown_item, genOp
            )
            genSpin.adapter = adapter

            genSpin.onItemSelectedListener = object :
                AdapterView.OnItemSelectedListener {
                override fun onItemSelected(
                    parent: AdapterView<*>, view: View, position: Int,
                    id: Long
                ) {
                    userGender = genOp[position]
                }

                override fun onNothingSelected(parent: AdapterView<*>) {
                    Toast.makeText(baseContext, "Select Gender",
Toast.LENGTH_SHORT).show()
                }
            }
        }

        activitySignupBinding.btnCreate.setOnClickListener {

            newUserName =
activitySignupBinding.textInputEditTextUserName.text.toString()
            email =
activitySignupBinding.textInputEditTextNewEmail.text.toString()
            password1 =
activitySignupBinding.textInputEditTextNewPassword.text.toString()
            password2 =
activitySignupBinding.textInputEditTextConfirmPassword.text.toString()
            newUserAge = if
(activitySignupBinding.textInputEditTextAge.text.toString() != "") {

Integer.parseInt(activitySignupBinding.textInputEditTextAge.text.toString())
            } else 0

            if (detailsValidityChecker(
                    newUserName,
                    email,
                    password1,
                    password2,
                    newUserAge,
                    userGender
                )
            ) {
                createAccount(email, password1)
```

```
                }
            }

            activitySignupBinding.btnCancel.setOnClickListener {
                gotoSignin()
            }

        }

        private fun createAccount(email: String, password: String) {
            isVertificationEmailSent = false

            auth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(this) { task ->
                    if (task.isSuccessful) {
                        isNewUser = true
                        Log.d(TAG, "createUserWithEmail:success")
                        val credential = EmailAuthProvider.getCredential(email,
password)
//                        linkWithCredential(credential)
                        gotoLogin()
                    } else {
                        Log.w(TAG, "createUserWithEmail:failure", task.exception)
                        Toast.makeText(
                            baseContext,
                            task.exception?.message.toString(),
                            Toast.LENGTH_SHORT
                        ).show()
                    }
                }
        }

        private fun gotoLogin() {
            val intent = Intent(this, LoginActivity::class.java)
            startActivity(intent)
            finish()
        }

        private fun gotoSignin() {
            val intent = Intent(this, SignIn::class.java)
            startActivity(intent)
            finish()
        }

        companion object {
            private const val TAG = "EmailPassword"
        }


        private fun detailsValidityChecker(
            name: String,
            email: String,
            password1: String,
            password2: String,
            age: Int,
            gender: String
        ): Boolean {

            if (TextUtils.isEmpty(name) || TextUtils.isEmpty(email) ||
TextUtils.isEmpty(password1)
                || TextUtils.isEmpty(password2) || (age == 0)
```

```
    ) {
        Toast.makeText(
            baseContext, "Please Fill all details",
            Toast.LENGTH_SHORT
        ).show()
        return false
    }

    if (age < 18 || age > 65) {
        Toast.makeText(
            baseContext, "Only 18 - 65 are allowed",
            Toast.LENGTH_SHORT
        ).show()
        return false
    }

    if (gender != "Male" && gender != "Female") {
        Toast.makeText(
            baseContext, "Please select a gender",
            Toast.LENGTH_SHORT
        ).show()
        return false
    }

    if (password1 != password2) {
        Toast.makeText(
            baseContext, "Passwords does not Match",
            Toast.LENGTH_SHORT
        ).show()
        return false
    }

    return (isPasswordValid(password1) && (isEmailValid(email)))
}

private fun isPasswordValid(password: String): Boolean {

    if (!(password.length > 6)) { //minimum chars 8
        Toast.makeText(
            baseContext, "Minimum password Length 6",
            Toast.LENGTH_SHORT
        ).show()
        return false
    }

    if (password.contains(" ")) { //can't contain space
        Toast.makeText(
            baseContext, "Password can't Contain Space",
            Toast.LENGTH_SHORT
        ).show()
        return false
    }


    var isPasswordHaveSmallLetters = false
    for (i in 90..122) {
        val c = i.toChar()
        val str1 = Character.toString(c)
        if (password.contains(str1)) {
            isPasswordHaveSmallLetters = true
        }
```

```
        }
        if (!isPasswordHaveSmallLetters) {
            return false
        }

        return true
    }

    private fun isEmailValid(email: String?): Boolean {

        val emailRegex = "^[a-zA-Z0-9_+&*-]+(?:\\." + "[a-zA-Z0-9_+&*-]+)*@" +
                "(?:[a-zA-Z0-9-]+\\.)+[a-z" + "A-Z]{2,7}$"

        val pat: Pattern = Pattern.compile(emailRegex)
        if (!pat.matcher(email).matches()) {
            Toast.makeText(
                baseContext, "Enter A valid email",
                Toast.LENGTH_SHORT
            ).show()
            return false
        }
        return true
    }

}
```

Код класу HomeScreen.

```
package com.dedsec_x47.trainer

import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.view.MenuItem
import android.view.View
import android.widget.Button
import android.widget.TextView
import androidx.appcompat.app.ActionBarDrawerToggle
import androidx.appcompat.app.AppCompatActivity
import androidx.drawerlayout.widget.DrawerLayout
import androidx.fragment.app.Fragment
import com.dedsec_x47.trainer.auth.BooVariable
import com.dedsec_x47.trainer.auth.SignIn
import com.dedsec_x47.trainer.auth.UserDetails
import com.dedsec_x47.trainer.auth.getUserImage
import com.dedsec_x47.trainer.homeFragments.*
import com.facebook.AccessToken
import com.facebook.login.LoginManager
import com.google.android.material.imageview.ShapeableImageView
import com.google.android.material.navigation.NavigationView
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase

class HomeScreen : AppCompatActivity() {

    lateinit var toggle: ActionBarDrawerToggle
    lateinit var profPic: ShapeableImageView
    lateinit var navView: NavigationView
    lateinit var hView: View
    lateinit var uname: TextView
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_home)
        UserDetails().getAlluserNames()

        navView = findViewById(R.id.navView)
        hView = navView.getHeaderView(0)

        profPic = hView.findViewById(R.id.savProfileImage)
        profPic.setImageURI(getUserImage())

        uname = hView.findViewById(R.id.tvFullName)
        //uname.text = UserDetails().readData("Name")

        val drawLayout: DrawerLayout = findViewById(R.id.drawerLayout)
        toggle = ActionBarDrawerToggle(this, drawLayout, R.string.open,
R.string.close)

        drawLayout.addDrawerListener(toggle)
        toggle.syncState()

        supportActionBar?.setDisplayHomeAsUpEnabled(true)
        navView.setNavigationItemSelectedListener {
            //val intent1 = Intent(this, SettingsActivity::class.java)
            val intent2 = Intent(this, FeedbackActivity::class.java)
            //val intent3 = Intent(this, SetAlarm::class.java)
            when (it.itemId) {
                //R.id.miSettings ->
startActivity(intent1)//Toast.makeText(applicationContext, "clicked settings",
Toast.LENGTH_SHORT).show()
                R.id.miFeedback ->
startActivity(intent2)//Toast.makeText(applicationContext, "clicked feedback",
Toast.LENGTH_SHORT).show()
                //R.id.miRemainder -> startActivity(intent3)
            }
            true
        }

        //for bottom navigation
        //to remove the icon tint
        val botNavView:
com.google.android.material.bottomnavigation.BottomNavigationView =
            findViewById(R.id.bottomNavigationView)
        botNavView.itemIconTintList = null

        botNavView.selectedItemId = R.id.miHome

        //val challengeFragment = ChallengeFragment()
        val statsFragment = StatsFragment()
        val homeFragment = HomeFragment()
        //val leaderboardFragment = LeaderboardFragment()
        val profileFragment = ProfileFragment()

        //set current fragment as home
        setCurrentFragment(homeFragment)

        botNavView.setOnItemSelectedListener {
            when (it.itemId) {
                //R.id.miChallenges -> setCurrentFragment(challengeFragment)
                R.id.miStats -> setCurrentFragment(statsFragment)
```

```
                R.id.miHome -> setCurrentFragment(homeFragment)
                //R.id.miLeaderboard -> setCurrentFragment(leaderboardFragment)
                R.id.miProfile -> setCurrentFragment(profileFragment)
            }
            true
        }

        val btnLogout: Button = findViewById(R.id.btnLogOut)
        btnLogout.setOnClickListener {
            logout()
        }

        autoUpdate()
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        if (toggle.onOptionsItemSelected(item)) {
            return true
        }
        return super.onOptionsItemSelected(item)
    }

    private fun setCurrentFragment(fragment: Fragment) {
        supportFragmentManager.beginTransaction().apply {
            replace(R.id.flFragment, fragment)
            commit()
        }
    }

    private fun logout() {
        val intent = Intent(this, SignIn::class.java)
        if (AccessToken.getCurrentAccessToken() != null) {
            LoginManager.getInstance().logOut()
        }
        val auth = Firebase.auth
        auth.signOut()
        startActivity(intent)
        finish()
    }

    private fun autoUpdate() {
        isNameOrPicUpdated.listener = object : BooVariable.ChangeListener {
            override fun onChange() {
                Log.d("BOOL", isNameOrPicUpdated.get().toString())
                if (isNameOrPicUpdated.get()) {
                    uname.text = UserDetails().readData("Name")
                    profPic.setImageURI(getUserImage())
                    isNameOrPicUpdated.set(false)
                }
            }
        }
    }
}
```

Код класу MyPreferences

```
package com.dedsec_x47.trainer.aiTrainer

import android.content.Context

class MyPreferences(context: Context) {
    val PREF_NAME = "TrainingsPrefs"
```

```
    val CURRENT_EXERSICE = "CurrentExercise"

    val PREF_PUSH_UPS_r1 = "PUSH1"
    val PREF_PUSH_UPS_r2 = "PUSH2"
    val PREF_PUSH_UPS_r3 = "PUSH3"

    val PREF_SQUATS_r1 = "SQUATS1"
    val PREF_SQUATS_r2 = "SQUATS2"
    val PREF_SQUATS_r3 = "SQUATS3"

    val PREF_SHIN_UPS_r1 = "SHIN_UPS1"
    val PREF_SHIN_UPS_r2 = "SHIN_UPS2"
    val PREF_SHIN_UPS_r3 = "SHIN_UPS3"

    val PREF_LEG_RISE_r1 = "LEG_RISE1"
    val PREF_LEG_RISE_r2 = "LEG_RISE2"
    val PREF_LEG_RISE_r3 = "LEG_RISE3"

    val CURRENT_ROUND_PUSHUPS = "CurrentRoundPushUps"
    val CURRENT_ROUND_SQUATS = "CurrentRoundSquats"
    val CURRENT_ROUND_SHINUPS = "CurrentRoundShinUps"
    val CURRENT_ROUND_LEGRISE = "CurrentRoundLegRise"

    val CURRENT_DAY_PUSHUPS = "CurrentDayPushUps"
    val CURRENT_DAY_SQUATS = "CurrentDaySquats"
    val CURRENT_DAY_SHINUPS = "CurrentDayShinUps"
    val CURRENT_DAY_LEGRISE = "CurrentDayLegRise"

    val NEED_MAX_PUSHUPS = "NeedMaxPushUps"
    val NEED_MAX_SQUATS = "NeedMaxSquats"
    val NEED_MAX_SHINUPS = "NeedMaxShinUps"
    val NEED_MAX_LEGRISE = "NeedMaxLegRise"

    val MAX_PUSHUPS = "MaxPushUps"
    val MAX_SQUATS = "MaxSquats"
    val MAX_SHINUPS = "MaxShinUps"
    val MAX_LEGRISE = "MaxLegRise"

    val PUSHUPS_COUNTER = "PushUpsCounter"
    val SQUATS_COUNTER = "SquatsCounter"
    val LEGRISE_COUNTER = "LegRiseCounter"

    val preference = context.getSharedPreferences(PREF_NAME,
Context.MODE_PRIVATE)
    val trainings = arrayListOf("Push Ups", "Squats", "Shin Ups", "Leg Raise")
    val LEVEL = "level"


    // EXERCISE COUNTER

    fun setCountEx(currentEx: String?, count: Int) {
        val editor = preference.edit()
        when (currentEx) {
            "Push Ups" -> editor.putInt(PUSHUPS_COUNTER, count).apply()
            "Squats" -> editor.putInt(SQUATS_COUNTER, count).apply()
            else -> editor.putInt(LEGRISE_COUNTER, count).apply()
        }
    }

    fun getCountEx(currentEx: String?): Int {
        return when (currentEx) {
```

```
        "Push Ups" -> preference.getInt(PUSHUPS_COUNTER, 0)
        "Squats" -> preference.getInt(SQUATS_COUNTER, 0)
        else -> preference.getInt(LEGRISE_COUNTER, 0)
    }
}


//ROUNDS

fun setCurrentRound(currentEx: String?, round: Int) {
    val editor = preference.edit()
    when (currentEx) {
        "Push Ups" -> editor.putInt(CURRENT_ROUND_PUSHUPS, round).apply()
        "Squats" -> editor.putInt(CURRENT_ROUND_SQUATS, round).apply()
        "Shin Ups" -> editor.putInt(CURRENT_ROUND_SHINUPS, round).apply()
        else -> editor.putInt(CURRENT_ROUND_LEGRISE, round).apply()
    }
}

fun getCurrentRound(currentEx: String?): Int {
    when (currentEx) {
        "Push Ups" -> return preference.getInt(CURRENT_ROUND_PUSHUPS, 0)
        "Squats" -> return preference.getInt(CURRENT_ROUND_SQUATS, 0)
        "Shin Ups" -> return preference.getInt(CURRENT_ROUND_SHINUPS, 0)
        else -> return preference.getInt(CURRENT_ROUND_LEGRISE, 0)
    }
}


// DAYS

fun setCurrentDay(currentEx: String?, day: Int) {
    val editor = preference.edit()
    when (currentEx) {
        "Push Ups" -> editor.putInt(CURRENT_DAY_PUSHUPS, day).apply()
        "Squats" -> editor.putInt(CURRENT_DAY_SQUATS, day).apply()
        "Shin Ups" -> editor.putInt(CURRENT_DAY_SHINUPS, day).apply()
        else -> editor.putInt(CURRENT_DAY_LEGRISE, day).apply()
    }
}

fun getCurrentDayPushUps(): Int {
    return preference.getInt(CURRENT_DAY_PUSHUPS, 0)
}

fun getCurrentDaySquats(): Int {
    return preference.getInt(CURRENT_DAY_SQUATS, 0)
}

fun getCurrentDayShinUps(): Int {
    return preference.getInt(CURRENT_DAY_SHINUPS, 0)
}

fun getCurrentDayLegRise(): Int {
    return preference.getInt(CURRENT_DAY_LEGRISE, 0)
}


//EXERCISE COUNT

fun getPushups(round: Int): Int {
```

```
    when (round) {
        0 -> return preference.getInt(PREF_PUSH_UPS_r1, 0)
        1 -> return preference.getInt(PREF_PUSH_UPS_r2, 0)
        2 -> return preference.getInt(PREF_PUSH_UPS_r3, 0)
        else -> return 0
    }
}

fun setPushups(round: Int, count: Int) {
    val editor = preference.edit()
    when (round) {
        0 -> editor.putInt(PREF_PUSH_UPS_r1, count)
        1 -> editor.putInt(PREF_PUSH_UPS_r2, count)
        2 -> editor.putInt(PREF_PUSH_UPS_r3, count)
    }
    editor.apply()
}

fun getSquats(round: Int): Int {
    when (round) {
        0 -> return preference.getInt(PREF_SQUATS_r1, 0)
        1 -> return preference.getInt(PREF_SQUATS_r2, 0)
        2 -> return preference.getInt(PREF_SQUATS_r3, 0)
        else -> return 0
    }
}

fun setSquats(round: Int, count: Int) {
    val editor = preference.edit()
    when (round) {
        0 -> editor.putInt(PREF_SQUATS_r1, count)
        1 -> editor.putInt(PREF_SQUATS_r2, count)
        2 -> editor.putInt(PREF_SQUATS_r3, count)
    }
    editor.apply()
}

fun getShinUps(round: Int): Int {
    when (round) {
        0 -> return preference.getInt(PREF_SHIN_UPS_r1, 0)
        1 -> return preference.getInt(PREF_SHIN_UPS_r2, 0)
        2 -> return preference.getInt(PREF_SHIN_UPS_r3, 0)
        else -> return 0
    }
}

fun setShinUps(round: Int, count: Int) {
    val editor = preference.edit()
    when (round) {
        0 -> editor.putInt(PREF_SHIN_UPS_r1, count)
        1 -> editor.putInt(PREF_SHIN_UPS_r2, count)
        2 -> editor.putInt(PREF_SHIN_UPS_r3, count)
    }
    editor.apply()
}

fun getLegRise(round: Int): Int {
    when (round) {
        0 -> return preference.getInt(PREF_LEG_RISE_r1, 0)
        1 -> return preference.getInt(PREF_LEG_RISE_r2, 0)
        2 -> return preference.getInt(PREF_LEG_RISE_r3, 0)
```

```
        else -> return 0
    }
}

fun setLegRise(round: Int, count: Int) {
    val editor = preference.edit()
    when (round) {
        0 -> editor.putInt(PREF_LEG_RISE_r1, count)
        1 -> editor.putInt(PREF_LEG_RISE_r2, count)
        2 -> editor.putInt(PREF_LEG_RISE_r3, count)
    }
    editor.apply()
}


// CURRENT EXERCISE

fun getCurrentEx(): String? {
    return preference.getString(CURRENT_EXERSICE, getTrainings(0))
}

fun setCurrentEx(number: Int) {
    val editor = preference.edit()
    editor.putString(CURRENT_EXERSICE, getTrainings(number))
    editor.apply()
}

private fun getTrainings(number: Int): String {
    return trainings[number]
}


// NEED MAXIMUM CHECK?

fun getNeedSetMax(currentEx: String?): Boolean {
    when (currentEx) {
        "Push Ups" -> return preference.getBoolean(NEED_MAX_PUSHUPS, true)
        "Squats" -> return preference.getBoolean(NEED_MAX_SQUATS, true)
        "Shin Ups" -> return preference.getBoolean(NEED_MAX_SHINUPS, true)
        else -> return preference.getBoolean(NEED_MAX_LEGRISE, true)
    }

}

fun setNeedSetMaxPushUps(value: Boolean) {
    val editor = preference.edit()
    editor.putBoolean(NEED_MAX_PUSHUPS, value).apply()
}

fun setNeedSetMaxSquats(value: Boolean) {
    val editor = preference.edit()
    editor.putBoolean(NEED_MAX_SQUATS, value).apply()
}

fun setNeedSetMaxShinUps(value: Boolean) {
    val editor = preference.edit()
    editor.putBoolean(NEED_MAX_SHINUPS, value).apply()
}

fun setNeedSetMaxLegRise(value: Boolean) {
    val editor = preference.edit()
```

```
        editor.putBoolean(NEED_MAX_LEGRISE, value).apply()
    }


    //SET MAXIMUM OF THE EXERCISE

    fun setMaxPushups(exersice: String?, count: Int) {
        val editor = preference.edit()
        when (exersice) {
            "Push Ups" -> editor.putInt(MAX_PUSHUPS, count)
            "Squats" -> editor.putInt(MAX_SQUATS, count)
            "Shin Ups" -> editor.putInt(MAX_SHINUPS, count)
            else -> editor.putInt(MAX_LEGRISE, count)
        }

        editor.apply()
    }

    fun getMaxPushups(): Int {
        return preference.getInt(MAX_PUSHUPS, 0)
    }

    fun getMaxSquats(): Int {
        return preference.getInt(MAX_SQUATS, 0)
    }

    fun getMaxShinUps(): Int {
        return preference.getInt(MAX_SHINUPS, 0)
    }

    fun getMaxLegRise(): Int {
        return preference.getInt(MAX_LEGRISE, 0)
    }

    fun getLevel(): Int {
        val pushUps = getCountEx("Push Ups")
        val legRaise = getCountEx("Leg Raise")
        val squats = getCountEx("Squats")
        val lvl = (pushUps + legRaise + squats) / 100
        return if (lvl <= 1) 1
        else (pushUps + legRaise + squats) / 100
    }


}
```

Код класу Max

```
package com.dedsec_x47.trainer.exercisePages.instructions

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import com.dedsec_x47.trainer.R
import com.dedsec_x47.trainer.aiTrainer.MyPreferences

class MaxPushUps : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_max_push_ups)

        val pref = MyPreferences(this)
        val currentEx = pref.getCurrentEx()
        val btnStart = findViewById<Button>(R.id.btnStartMax)
        val title: TextView = findViewById(R.id.txtTrainingTitle)

        when (currentEx) {
            "Push Ups" -> {
                title.text = "Push Ups training"
                btnStart.setOnClickListener {
                    val intent = Intent(this, PushUpInstr::class.java)
                    startActivity(intent)
                }

            }
            "Squats" -> {
                title.text = "Squats training"
                btnStart.setOnClickListener {
                    val intent = Intent(this, SquatInst::class.java)
                    startActivity(intent)
                }
            }
            "Shin Ups" -> {
                title.text = "Shin Ups training"
                btnStart.setOnClickListener {
                    val intent = Intent(this, ChinUpInst::class.java)
                    startActivity(intent)
                }

            }
            "Leg Raise" -> {
                title.text = "Leg Raise training"
                btnStart.setOnClickListener {
                    val intent = Intent(this, LegRaiseInst::class.java)
                    startActivity(intent)
                }
            }
        }
    }
}
```

Код класу TrainingPage

```
package com.dedsec_x47.trainer.exercisePages

import android.annotation.SuppressLint
import android.content.Intent
import android.graphics.Color
import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.isVisible
import com.dedsec_x47.trainer.HomeScreen
import com.dedsec_x47.trainer.R
import com.dedsec_x47.trainer.aiTrainer.MyPreferences
import com.dedsec_x47.trainer.exercisePages.instructions.LegRaiseInst
import com.dedsec_x47.trainer.exercisePages.instructions.PushUpInstr
```

```
import com.dedsec_x47.trainer.exercisePages.instructions.SquatInst

class TrainingPage : AppCompatActivity() {


    @SuppressLint("SetTextI18n")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_training_page)

        val btnStart = findViewById<Button>(R.id.btnStartPushUpsTrainigs)
        val btnStart2 = findViewById<Button>(R.id.btnStartPushUpsTrainigs2)
        val btnStart3 = findViewById<Button>(R.id.btnStartPushUpsTrainigs3)
        val btnExcellent = findViewById<Button>(R.id.btnExcellent)
        val pref = MyPreferences(this)
        val currentEx = pref.getCurrentEx()

        val title: TextView = findViewById(R.id.txtPushUpsTraining)
        val exCounter: TextView = findViewById(R.id.txtTitle_ex1)
        val exCounter2: TextView = findViewById(R.id.txtTitle_ex2)
        val exCounter3: TextView = findViewById(R.id.txtTitle_ex3)
        val txtCong: TextView = findViewById(R.id.txtCong)
        val timer1: TextView = findViewById(R.id.txtDesc_ex1)
        val timer2: TextView = findViewById(R.id.txtDesc_ex2)
        val timer3: TextView = findViewById(R.id.txtDesc_ex3)
        val txtDays: TextView = findViewById(R.id.txtDayCount)

        val currentMaxPushUps: Int = pref.getMaxPushups()
        val currentMaxSquats: Int = pref.getMaxSquats()
        val currentMaxLegRise: Int = pref.getMaxLegRise()


        val trainingProgramPushUp = arrayOf(
            arrayOf(0.6, 0.5, 0.45),
            arrayOf(0.65, 0.55, 0.5),
            arrayOf(0.7, 0.6, 0.55)
        )
        val trainingProgramSquats = arrayOf(
            arrayOf(0.7, 0.6, 0.5),
            arrayOf(0.8, 0.7, 0.6),
            arrayOf(0.9, 0.8, 0.7)
        )
        val trainingProgramLegRise = arrayOf(
            arrayOf(0.65, 0.55, 0.50),
            arrayOf(0.7, 0.6, 0.55),
            arrayOf(0.75, 0.65, 0.60)
        )

        val dayPushUps = pref.getCurrentDayPushUps()
        val daySquats = pref.getCurrentDaySquats()
        val dayLegRise = pref.getCurrentDayLegRise()
        val doneExPushUps = arrayOf(pref.getPushups(0), pref.getPushups(1),
pref.getPushups(2))
        val doneExSquats = arrayOf(pref.getSquats(0), pref.getSquats(1),
pref.getSquats(2))
        val doneExLegRise = arrayOf(pref.getLegRise(0), pref.getLegRise(1),
pref.getLegRise(2))
        var restTime = 30

        when (currentEx) {
            "Push Ups" -> {
```

```
                title.text = "Push Ups training"
                val max1 = (trainingProgramPushUp[dayPushUps][0] *
currentMaxPushUps).toInt()
                val max2 = (trainingProgramPushUp[dayPushUps][1] *
currentMaxPushUps).toInt()
                val max3 = (trainingProgramPushUp[dayPushUps][2] *
currentMaxPushUps).toInt()
                restTime += 30 * ((max1 + 5) / 10)
                txtDays.text = "Day ${dayPushUps + 1} / 3"
                exCounter.text = "${doneExPushUps[0]} / $max1 reps"
                timer1.text = "rest ${timeToString(restTime)}"
                exCounter2.text = "${doneExPushUps[1]} / $max2 reps"
                timer2.text = "rest ${timeToString(restTime + 30)}"
                exCounter3.text = "${doneExPushUps[2]} / $max3 reps"
                timer3.text = "rest ${timeToString(restTime + 60)}"
                when (pref.getCurrentRound(pref.getCurrentEx())) {
                    0 -> {
                        btnStart2.isVisible = false
                        btnStart3.isVisible = false
                        txtCong.isVisible = false
                        btnExcellent.isVisible = false
                    }
                    1 -> {
                        setDoneState(btnStart)
                        btnStart3.isVisible = false
                        txtCong.isVisible = false
                        btnExcellent.isVisible = false
                    }
                    2 -> {
                        setDoneState(btnStart)
                        setDoneState(btnStart2)
                        txtCong.isVisible = false
                        btnExcellent.isVisible = false
                    }
                    else -> {
                        setDoneState(btnStart)
                        setDoneState(btnStart2)
                        setDoneState(btnStart3)
                        txtCong.isVisible = true
                        btnExcellent.isVisible = true
                    }
                }

                btnStart.setOnClickListener {
                    val intent = Intent(this, PushUpInstr::class.java)
                    pref.setCurrentRound(pref.getCurrentEx(), 0)
                    startActivity(intent)
                }
                btnStart2.setOnClickListener {
                    val intent = Intent(this, PushUpInstr::class.java)
                    pref.setCurrentRound(pref.getCurrentEx(), 1)
                    startActivity(intent)
                }
                btnStart3.setOnClickListener {
                    val intent = Intent(this, PushUpInstr::class.java)
                    pref.setCurrentRound(pref.getCurrentEx(), 2)
                    startActivity(intent)
                }

                btnExcellent.setOnClickListener {
                    if (dayPushUps < 2) {
```

```
                        pref.setCurrentDay(pref.getCurrentEx(), dayPushUps + 1)
                    } else {
                        pref.setCurrentDay(pref.getCurrentEx(), 0)
                        pref.setNeedSetMaxPushUps(true)
                    }
                    pref.setCurrentRound(pref.getCurrentEx(), 0)
                    pref.setPushups(0, 0)
                    pref.setPushups(1, 0)
                    pref.setPushups(2, 0)
                    val intent = Intent(this, HomeScreen::class.java)
                    startActivity(intent)
                    finish()
                }

            }
            "Squats" -> {
                title.text = "Squats training"
                btnStart.setOnClickListener {
                    val intent = Intent(this, SquatInst::class.java)
                    startActivity(intent)
                }
                val max1 = (trainingProgramSquats[daySquats][0] *
currentMaxSquats).toInt()
                val max2 = (trainingProgramSquats[daySquats][1] *
currentMaxSquats).toInt()
                val max3 = (trainingProgramSquats[daySquats][2] *
currentMaxSquats).toInt()
                restTime += 30 * ((max1 + 5) / 10)
                txtDays.text = "Day ${daySquats + 1} / 3"
                exCounter.text = "${doneExSquats[0]} / $max1 reps"
                timer1.text = "rest ${timeToString(restTime)}"
                exCounter2.text = "${doneExSquats[1]} / $max2 reps"
                timer2.text = "rest ${timeToString(restTime + 30)}"
                exCounter3.text = "${doneExSquats[2]} / $max3 reps"
                timer3.text = "rest ${timeToString(restTime + 60)}"
                when (pref.getCurrentRound(pref.getCurrentEx())) {
                    0 -> {
                        btnStart2.isVisible = false
                        btnStart3.isVisible = false
                        txtCong.isVisible = false
                        btnExcellent.isVisible = false
                    }
                    1 -> {
                        setDoneState(btnStart)
                        btnStart3.isVisible = false
                        txtCong.isVisible = false
                        btnExcellent.isVisible = false
                    }
                    2 -> {
                        setDoneState(btnStart)
                        setDoneState(btnStart2)
                        txtCong.isVisible = false
                        btnExcellent.isVisible = false
                    }
                    else -> {
                        setDoneState(btnStart)
                        setDoneState(btnStart2)
                        setDoneState(btnStart3)
                        txtCong.isVisible = true
                        btnExcellent.isVisible = true
                    }
```

```
                }

                btnStart.setOnClickListener {
                    val intent = Intent(this, SquatInst::class.java)
                    pref.setCurrentRound(pref.getCurrentEx(), 0)
                    startActivity(intent)
                }
                btnStart2.setOnClickListener {
                    val intent = Intent(this, SquatInst::class.java)
                    pref.setCurrentRound(pref.getCurrentEx(), 1)
                    startActivity(intent)
                }
                btnStart3.setOnClickListener {
                    val intent = Intent(this, SquatInst::class.java)
                    pref.setCurrentRound(pref.getCurrentEx(), 2)
                    startActivity(intent)
                }

                btnExcellent.setOnClickListener {
                    if (daySquats < 2) {
                        pref.setCurrentDay(pref.getCurrentEx(), daySquats + 1)
                    } else {
                        pref.setCurrentDay(pref.getCurrentEx(), 0)
                        pref.setNeedSetMaxSquats(true)
                    }
                    pref.setCurrentRound(pref.getCurrentEx(), 0)
                    pref.setSquats(0, 0)
                    pref.setSquats(1, 0)
                    pref.setSquats(2, 0)
                    val intent = Intent(this, HomeScreen::class.java)
                    startActivity(intent)
                    finish()
                }

            }
            "Shin Ups" -> {}
            "Leg Raise" -> {
                title.text = "Leg Raise training"
                btnStart.setOnClickListener {
                    val intent = Intent(this, LegRaiseInst::class.java)
                    startActivity(intent)
                }
                val max1 = (trainingProgramLegRise[dayLegRise][0] *
currentMaxLegRise).toInt()
                val max2 = (trainingProgramLegRise[dayLegRise][1] *
currentMaxLegRise).toInt()
                val max3 = (trainingProgramLegRise[dayLegRise][2] *
currentMaxLegRise).toInt()
                restTime += 30 * ((max1 + 5) / 10)
                txtDays.text = "Day ${dayLegRise + 1} / 3"
                exCounter.text = "${doneExLegRise[0]} / $max1 reps"
                timer1.text = "rest ${timeToString(restTime)}"
                exCounter2.text = "${doneExLegRise[1]} / $max2 reps"
                timer2.text = "rest ${timeToString(restTime + 30)}"
                exCounter3.text = "${doneExLegRise[2]} / $max3 reps"
                timer3.text = "rest ${timeToString(restTime + 60)}"
                when (pref.getCurrentRound(pref.getCurrentEx())) {
                    0 -> {
                        btnStart2.isVisible = false
                        btnStart3.isVisible = false
                        txtCong.isVisible = false
```

```
                    btnExcellent.isVisible = false
                }
                1 -> {
                    setDoneState(btnStart)
                    btnStart3.isVisible = false
                    txtCong.isVisible = false
                    btnExcellent.isVisible = false
                }
                2 -> {
                    setDoneState(btnStart)
                    setDoneState(btnStart2)
                    txtCong.isVisible = false
                    btnExcellent.isVisible = false
                }
                else -> {
                    setDoneState(btnStart)
                    setDoneState(btnStart2)
                    setDoneState(btnStart3)
                    txtCong.isVisible = true
                    btnExcellent.isVisible = true
                }
            }

            btnStart.setOnClickListener {
                val intent = Intent(this, LegRaiseInst::class.java)
                pref.setCurrentRound(pref.getCurrentEx(), 0)
                startActivity(intent)
            }
            btnStart2.setOnClickListener {
                val intent = Intent(this, LegRaiseInst::class.java)
                pref.setCurrentRound(pref.getCurrentEx(), 1)
                startActivity(intent)
            }
            btnStart3.setOnClickListener {
                val intent = Intent(this, LegRaiseInst::class.java)
                pref.setCurrentRound(pref.getCurrentEx(), 2)
                startActivity(intent)
            }

            btnExcellent.setOnClickListener {
                if (dayLegRise < 2) {
                    pref.setCurrentDay(pref.getCurrentEx(), dayLegRise + 1)
                } else {
                    pref.setCurrentDay(pref.getCurrentEx(), 0)
                    pref.setNeedSetMaxLegRise(true)
                }
                pref.setCurrentRound(pref.getCurrentEx(), 0)
                pref.setLegRise(0, 0)
                pref.setLegRise(1, 0)
                pref.setLegRise(2, 0)
                val intent = Intent(this, HomeScreen::class.java)
                startActivity(intent)
                finish()

            }
        }
    }
}

private fun timeToString(secs: Int): String {
    val min = secs / 60 % 60
```

```
        val sec = secs / 1 % 60
        return String.format("$min:%02d", sec)
    }


    override fun onBackPressed() {
        val intent = Intent(this, WeightLoss::class.java)
        startActivity(intent)
        finish()
    }

    private fun setDoneState(btnStart: Button?) {
        if (btnStart != null) {
            btnStart.text = "Done"
            btnStart.isEnabled = false
            btnStart.background = null
            btnStart.setTextColor(Color.parseColor("#2979FF"))
        }
    }
}
```

Код класу PoseDetect

```
package com.dedsec_x47.trainer.aiTrainer

import android.Manifest
import android.app.AlertDialog
import android.app.Dialog
import android.content.Intent
import android.content.pm.PackageManager
import android.os.Bundle
import android.os.Process
import android.view.SurfaceView
import android.view.WindowManager
import android.widget.Button
import android.widget.Chronometer
import android.widget.TextView
import android.widget.ToggleButton
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import androidx.core.view.isVisible
import androidx.fragment.app.DialogFragment
import androidx.lifecycle.lifecycleScope
import com.dedsec_x47.trainer.HomeScreen
import com.dedsec_x47.trainer.R
import com.dedsec_x47.trainer.View
import com.dedsec_x47.trainer.aiTrainer.camera.CameraSource
import com.dedsec_x47.trainer.aiTrainer.data.Accelerator
import com.dedsec_x47.trainer.aiTrainer.pose.PoseEstimate
import com.dedsec_x47.trainer.exercisePages.TrainingPage
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch

class poseDetect : AppCompatActivity() {
    companion object {
        init {
            System.loadLibrary("native-lib")
        }
```

```
        private const val FRAGMENT_DIALOG = "dialog"

        var currentExercise = Exercise.ShoulderpressDumbell
        var currView = View.right
        var skelShow: Boolean = false
        var shDebug: ToggleButton? = null
    }

    private lateinit var surfaceView: SurfaceView
    private lateinit var repView: TextView
    private lateinit var repType: TextView
    private lateinit var timerView: Chronometer
    private lateinit var doneBtn: Button

    private var accelerator = Accelerator.CPU

    private lateinit var fpsView: TextView
    private var cameraSource: CameraSource? = null

    private val requestPermission =
        registerForActivityResult(ActivityResultContracts.RequestPermission()) {
isGranted: Boolean ->
            if (isGranted) openCamera()
            else {
                ErrorDialog.newInstance("This app requires camera permission")
                    .show(supportFragmentManager, FRAGMENT_DIALOG)
            }
        }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        if ((ContextCompat.checkSelfPermission(
                this,
                Manifest.permission.READ_EXTERNAL_STORAGE
            ) != PackageManager.PERMISSION_GRANTED) ||
            (ContextCompat.checkSelfPermission(
                this,
                Manifest.permission.WRITE_EXTERNAL_STORAGE
            ) != PackageManager.PERMISSION_GRANTED) ||
            (ContextCompat.checkSelfPermission(
                this,
                Manifest.permission.CAMERA
            ) != PackageManager.PERMISSION_GRANTED)
        ) {
            ActivityCompat.requestPermissions(
                this, arrayOf(
                    Manifest.permission.READ_EXTERNAL_STORAGE,
                    Manifest.permission.WRITE_EXTERNAL_STORAGE,
                    Manifest.permission.CAMERA
                ), 101
            )
        }
        setContentView(R.layout.activity_posedetect)

        window.addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON)
//keep screen on while app is running
        surfaceView = findViewById(R.id.surfaceView)
        shDebug = findViewById(R.id.hideSkeletonModeBtn)
        repView = findViewById(R.id.tvRepetitionCount)
        repType = findViewById(R.id.tvRepetition)
```

```
    timerView = findViewById(R.id.c_meter)
    doneBtn = findViewById(R.id.doneBtn)

    doneBtn.setOnClickListener {
        val prefs = MyPreferences(this)
        val counts = repView.text.toString()
        var checkMax = prefs.getNeedSetMax(prefs.getCurrentEx())
        var intent: Intent

        if (checkMax) {
            prefs.setCountEx(
                prefs.getCurrentEx(),
                (prefs.getCountEx(prefs.getCurrentEx()) + counts.toInt())
            )
            prefs.setMaxPushups(prefs.getCurrentEx(), counts.toInt())
            checkMax = false
            prefs.setNeedSetMaxPushUps(checkMax)
            prefs.setNeedSetMaxSquats(checkMax)
            prefs.setNeedSetMaxShinUps(checkMax)
            prefs.setNeedSetMaxLegRise(checkMax)
            prefs.setCurrentRound(prefs.getCurrentEx(), 0)
            intent = Intent(this, HomeScreen::class.java)
        } else {
            var round = prefs.getCurrentRound(prefs.getCurrentEx())
            prefs.setCountEx(
                prefs.getCurrentEx(),
                (prefs.getCountEx(prefs.getCurrentEx()) + counts.toInt())
            )
            when (prefs.getCurrentEx()) {
                "Push Ups" -> prefs.setPushups(round, counts.toInt())
                "Squats" -> prefs.setSquats(round, counts.toInt())
                "Shin Ups" -> prefs.setShinUps(round, counts.toInt())
                else -> prefs.setLegRise(round, counts.toInt())
            }
            round++
            prefs.setCurrentRound(prefs.getCurrentEx(), round)
            intent = Intent(this, TrainingPage::class.java)
        }

        cameraSource?.close()
        cameraSource = null
        startActivity(intent)
        finish()
    }

    if (currentExercise == Exercise.Plank) {
        repView.isVisible = false
        timerView.isVisible = true
        repType.text = "Time"
    }
}

override fun onStart() {
    super.onStart()
    openCamera()
}

override fun onResume() {
    cameraSource?.resume()
    super.onResume()
}
```

```
    override fun onPause() {
        cameraSource?.close()
        cameraSource = null
        super.onPause()
    }

    private fun isCameraPermissionGranted(): Boolean {
        return (checkPermission(
            Manifest.permission.CAMERA,
            Process.myPid(),
            Process.myUid()
        ) == PackageManager.PERMISSION_GRANTED)
    }

    private fun openCamera() {
        if (isCameraPermissionGranted()) {
            if (cameraSource == null) {
                cameraSource = CameraSource(
                    this,
                    repView,
                    timerView,
                    surfaceView,
                    object : CameraSource.CameraSourceListener {
                        override fun onFPSListener(fps: Int) {
                            //fpsView.text = getString(R.string.fps, fps
                        }
                    }).apply {
                    prepareCamera()
                }
                lifecycleScope.launch(Dispatchers.Main) {
                    cameraSource?.initCamera()
                }
            }
            createPoseEstimator()
        }
    }

    private fun createPoseEstimator() {
        cameraSource?.setDetector(PoseEstimate.create(this, accelerator))
    }

    private fun requestPermission() {
        when (PackageManager.PERMISSION_GRANTED) {
            ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
-> {
                openCamera()
            }
            else -> {

//requestPermission.launch(Manifest.permission.WRITE_EXTERNAL_STORAGE)
                requestPermission.launch(Manifest.permission.CAMERA)
            }
        }
    }

    // Shows an error message dialog.
    class ErrorDialog : DialogFragment() {
        override fun onCreateDialog(savedInstanceState: Bundle?): Dialog =
            AlertDialog.Builder(activity)
                .setMessage(requireArguments().getString(ARG_MESSAGE))
```

```
                .setPositiveButton(android.R.string.ok) { _, _ -> }
                .create()

        companion object {

            @JvmStatic
            private val ARG_MESSAGE = "msg"

            @JvmStatic
            fun newInstance(message: String): ErrorDialog = ErrorDialog().apply
{
                arguments = Bundle().apply { putString(ARG_MESSAGE, message) }
            }
        }
    }
}
```

Код класу PushUpClassify

```
package com.dedsec_x47.trainer.aiTrainer.poseClassify

import android.content.ContentValues
import android.graphics.Bitmap
import android.graphics.PointF
import android.media.MediaPlayer
import android.util.Log
import android.view.SurfaceView
import com.dedsec_x47.trainer.R
import com.dedsec_x47.trainer.aiTrainer.data.Human
import com.dedsec_x47.trainer.aiTrainer.data.KeyPoints
import com.dedsec_x47.trainer.aiTrainer.render.Visual

object PushUp {
    // Angle vals for Hammercurl SET1
    //1 -hands UP 2 - DOWN

    private val SHKAngle1 = 160

    private val WESAngle1 = 90
    private val WESAngle2 = 160

    private val angleThreshold = 15

    private var isExeriseStarted: Boolean = false

    private var mediaPlayer: MediaPlayer? = null
    private var count: Int = 0

    //check users position
    private var isUp =
        true                    // initail pos -- triggered when user initializes
UP pos

    fun getPushUpAngles(person: Human, bitmap: Bitmap, surfaceView:
SurfaceView): Int {
        // estimated WES angle - LEFT
        var esWESAngleL = Visual.getAngle(
            listOf<PointF>(
                person.keyPoints[KeyPoints.LEFT_WRIST.position].coordinate,
                person.keyPoints[KeyPoints.LEFT_ELBOW.position].coordinate,
```

```
                person.keyPoints[KeyPoints.LEFT_SHOULDER.position].coordinate
            )
        )

        // estimated WES angle - RIGHT
        var esWESAngleR = Visual.getAngle(
            listOf<PointF>(
                person.keyPoints[KeyPoints.RIGHT_WRIST.position].coordinate,
                person.keyPoints[KeyPoints.RIGHT_ELBOW.position].coordinate,
                person.keyPoints[KeyPoints.RIGHT_SHOULDER.position].coordinate
            )
        )

        // estimated SHK angle - LEFT
        var esSHKAngleL = Visual.getAngle(
            listOf<PointF>(
                person.keyPoints[KeyPoints.LEFT_SHOULDER.position].coordinate,
                person.keyPoints[KeyPoints.LEFT_HIP.position].coordinate,
                person.keyPoints[KeyPoints.LEFT_KNEE.position].coordinate
            )
        )

        // estimated SHK angle - RIGHT
        var esSHKAngleR = Visual.getAngle(
            listOf<PointF>(
                person.keyPoints[KeyPoints.RIGHT_SHOULDER.position].coordinate,
                person.keyPoints[KeyPoints.RIGHT_HIP.position].coordinate,
                person.keyPoints[KeyPoints.RIGHT_KNEE.position].coordinate
            )
        )

        Log.d(
            ContentValues.TAG,
            "ANGLE  L :" + esSHKAngleL.toString() + "  R : " +
esSHKAngleR.toString()
        )

        return checkPosition(
            esWESAngleL,
            esWESAngleR,
            esSHKAngleL,
            esSHKAngleR,
            person,
            bitmap,
            surfaceView
        )
    }

    private fun checkPosition(
        esWESAngleL: Double, esWESAngleR: Double, esSHKAngleL: Double,
esSHKAngleR: Double,
        person: Human, bitmap: Bitmap, surfaceView: SurfaceView
    ): Int {

        var WESCHK = false
        var SHKCHK = false

        // if already up -- listen for down angles
        if (isUp) {
            if (chk(
                    esWESAngleL,
```

```
                    esWESAngleR,
                    esSHKAngleL,
                    esSHKAngleR,
                    person,
                    bitmap,
                    surfaceView
                )
            ) {
            } else {
                WESCHK = (esWESAngleL <= WESAngle1 || esWESAngleR <= WESAngle1)
                SHKCHK = (esSHKAngleL >= SHKAngle1 || esSHKAngleR >= SHKAngle1)
            }
        }
        // else -- listen for up angles
        else {
            if (chk(
                    esWESAngleL,
                    esWESAngleR,
                    esSHKAngleL,
                    esSHKAngleR,
                    person,
                    bitmap,
                    surfaceView
                )
            ) {
            } else {
                WESCHK = (esWESAngleL >= WESAngle2 || esWESAngleR >= WESAngle2)
                SHKCHK = (esSHKAngleL >= SHKAngle1 || esSHKAngleR >= SHKAngle1)
            }
        }

        if (WESCHK && SHKCHK) {
            if (isExeriseStarted) {
                if (mediaPlayer == null) {
                    mediaPlayer = MediaPlayer.create(surfaceView.context,
R.raw.ring)
                }
                if (mediaPlayer != null && !mediaPlayer!!.isPlaying) {
                    mediaPlayer!!.release()
                    mediaPlayer = null
                    mediaPlayer = MediaPlayer.create(surfaceView.context,
R.raw.ring)
                    mediaPlayer!!.start()
                }

                if (isUp) count++
                isUp = !isUp
            } else {
                isExeriseStarted = true
                isUp = false
            }
        }
        return count
    }

    private fun chk(
        esWESAngleL: Double,
        esWESAngleR: Double,
        esSHKAngleL: Double,
        esSHKAngleR: Double,
        person: Human,
```

```
        bitmap: Bitmap,
        surfaceView: SurfaceView
    ): Boolean {
        if (mediaPlayer == null) {
            mediaPlayer = MediaPlayer.create(surfaceView.context, R.raw.ring)
        }
        if (mediaPlayer != null && !mediaPlayer!!.isPlaying) {
            mediaPlayer!!.release()
            mediaPlayer = null
            if (esSHKAngleL <= SHKAngle1 - 10 && esSHKAngleR <= SHKAngle1 - 10)
{
                drawOnImg(surfaceView, bitmap, person, R.raw.straibody, 2, 3, 4,
5)
                return true
            }
        }
        return false
    }

    private fun drawOnImg(
        surfaceView: SurfaceView, bitmap: Bitmap, person: Human, uri: Int,
        no1: Int, no2: Int, no3: Int, no4: Int
    ) {
        mediaPlayer = MediaPlayer.create(surfaceView.context, uri)
        mediaPlayer!!.start()

        /*if (isExeriseStarted) {
            Visual.drawWrongPose(bitmap, surfaceView, person, no1, no2)
            Visual.drawWrongPose(bitmap, surfaceView, person, no3, no4)
        }*/
    }

    fun setCountZero() {
        count = 0
    }
}
```