ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка застосунку з програмою тренувань фізичних якостей для різних видів спорту мовою Python»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Олександр ІЛЛЮЧЕНКО
(підпис)

Виконав: здобувач вищої освіти групи ПД-42
Олександр ІЛЛЮЧЕНКО

Керівник: Оксана ЗОЛОТУХІНА
к.т.н., доцент

Рецензент:

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Іллюченко Олександру Сергійовичу

1. Тема кваліфікаційної роботи: «Розробка застосунку з програмою тренувань фізичних якостей для різних видів спорту мовою Python»
Керівник кваліфікаційної роботи к.т.н., доц., доцент кафедри ІПЗ Оксана ЗОЛОТУХІНА,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вхідні дані до кваліфікаційної роботи:
   3.1 Науково-технічна література.
   3.2 Офіційна документація Kivy.
   3.3 Інформація про тренування фізичних якостей у футболі, баскетболі та волейболі.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   1. Огляд та аналіз існуючих додатків з тренуваннями фізичних здібностей у різних видах спорту.

2. Обґрунтування засобів розробки та проектування програмного забезпечення.
3. Реалізація функціональності додатку.
4. Визначення перспектив та можливості до удосконалення додатку.
5. Тестування додатку.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Засоби програмної реалізації.
4. Діаграма варіантів використання.
5. Діаграма класів.
6. Екранні форми.
7. Демонстрація роботи програми.
8. Апробація результатів дослідження.

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Огляд засобів розробки та проектування програмного забезпечення	14.03-20.03.2024	
4	Реалізація функціональності додатку	21.03-17.04.2024	
5	Дослідження перспектив та можливостей до удосконалення додатку	18.04-20.04.2024	
6	Тестування застосунку	21.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Олександр ІЛЛЮЧЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Оксана ЗОЛОТУХІНА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 49 стор., 2 табл., 29 рис., 13 джерел.

Мета роботи – підтримка процесу тренування фізичних якостей у обраному виді спорту засобами мобільного додатку, створеного мовою програмування Python.

Об`єкт дослідження – тренування фізичних якостей у обраному виді спорту.

Предмет дослідження – програмне забезпечення для тренування фізичних якостей у обраному виді спорту.

Короткий зміст роботи: В роботі проаналізовано додатки-аналоги у сфері тренування фізичних якостей. Проаналізовано інструментальні засоби для розробки додатку з тренувань фізичних якостей: Figma, Pycharm, Kivy та мова програмування Python. Розроблено додаток для тренувань та програмно реалізовані ключові функціональні можливості, зокрема: реєстрація, авторизація, вибір категорії спорту, перегляд вправ. Проведено функціональне тестування додатку. В роботі використано Figma для проектування користувацького інтерфейсу, середовище розробки PyCharm, мова програмування Python, фреймворк Kivy для створення користувацького інтерфейсу.

Сферою використання застосунку є тренування фізичних якостей у різних видах спорту.

КЛЮЧОВІ СЛОВА: ТРЕНУВАННЯ ФІЗИЧНИХ ЯКОСТЕЙ, ВИДИ СПОРТУ, МОБІЛЬНИЙ ЗАСТОСУНОК, PYTHON.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - Application Programming Interface

ПЗ - Програмне забезпечення

KV - Kivy Language

UI - User Interface

ВСТУП

У сучасному світі майже кожен користується смартфоном, цей гаджет майже весь день поряд з користувачем. Люди проводять перед екранами в середньому 6 годин 40 хвилин на день, через що є великий інтерес до мобільних додатків. Користувачі бажають мати усе в одному місці, а саме все потрібне для них у додатках в їх смартфонах [1]. Не менш популярним є і розвинення аматорського спорту, на відміну від великої кількості існуючих професійних та напів професійних команд, кількість чемпіонатів аматорського рівня збільшується з кожним сезоном, як наприклад у організації KSL що проводить футбольні турніри аматорського рівня в Україні та Польщі, кількість чемпіонатів що базуються у Києві виросла з 5 до 17 за останні 2 роки [2], а разом з ними також розширюють свої турніри і інші організації, середня кількість гравців у одному чемпіонаті досягає 160 учасників, і лише третина з них має бази тренувань. Враховуючи усі ці фактори, виникає актуальна потреба у розробці програмного забезпечення що надасть альтернативу звичайних тренувань для обраного спорту, і дасть змогу бажаючим тренуватись індивідуально у різних видах спорту, зокрема, у футболі, волейболі та баскетболі. Кожна з цих категорій має свій власний набір вправ, спрямованих на розвиток ключових фізичних якостей, необхідних для успішної гри в обраний спорт. Мобільний додаток забезпечує інтерфейс для користувачів, де вони матимуть доступ до вправ, розроблених професійними тренерами з урахуванням вимог конкретного виду спорту. Кожен вид спорту має свою унікальну категорію вправ, що сприятиме розвитку відповідних фізичних якостей. Наприклад, для футболістів в додатку потрібні вправи на покращення швидкості, витривалості та удару, у волейболістів - вправи на розвиток стрибкових якостей та координації, а у баскетболістів - на зміцнення м'язів та мобільність.

Мета роботи – підтримка процесу тренування фізичних якостей у обраному виді спорту засобами мобільного додатку, створеного мовою програмування Python.

Об`єкт дослідження – тренування фізичних якостей у обраному виді спорту.

Предмет дослідження – програмне забезпечення для тренувань фізичних якостей у обраному виді спорту.

В процесі дослідження виконувались наступні завдання:

- аналіз обраної предметної області в сфері тренувань фізичних якостей;

- порівняння існуючого програмного забезпечення для тренувань у різних видах спорту;

- визначення та моделювання вимог до програмного забезпечення;

- розробка архітектури додатку для тренувань;

- проектування та розробка програмного забезпечення для тренувань;

- тестування розробленого додатку.

Практична значущість додатку полягає у наданні спортсменам можливість тренуватися в будь-який час та будь-якому місці, навіть коли фізичний контакт з тренером або командою не є можливим. Крім того, використання мобільного додатка для тренування дозволить гравцям зберігати свою фізичну форму, навіть якщо вони не можуть регулярно збиратися на тренування з командою через свої роз'єднані графіки зайнятості. Такий додаток стане корисним як для тих, хто лише починає займатися обраним видом спорту, так і для досвідчених спортсменів, які бажають розвивати або підтримувати свою фізичну форму або поліпшити свої навички.

Робота пройшла апробацію на Всеукраїнській науково-технічній конференції «Застосування програмного забезпечення в ІКТ» (24.04.2024, м.Київ, ДУІКТ), Всеукраїнській науково-практичній конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті» (15.05.2024, м.Київ, ДУІКТ). За результатами апробації опубліковано тези доповідей.

1 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ ЗАСОБІВ З ТРЕНУВАННЯМИ ФІЗИЧНИХ ЗДІБНОСТЕЙ У РІЗНИХ ВИДАХ СПОРТУ

1.1 Особливості розробки програмного забезпечення для тренування фізичних якостей у різних видах спорту

Додаток для тренувань фізичних якостей у різних видах спорту - це програма, що надасть користувачам змогу тренувати свої фізичні якості, які використовуються в обраному ними спорті, не прив’язуючись до команд або тренувальних баз, тобто виконувати ці вправи вони зможуть там де їм зручно, та коли їм зручно. Вправи розраховані на індивідуальне виконання що надає змогу користувачу зробити графік тренувань особисто під свій вільний час. Фізичні якості, що тренуються вправами з цього додатку - це ті якості які є основними для кожного гравця у їх виді спорту. Рейтинги що випускаються різними платформами або ж навіть іграми, формуються саме з рейтингів навичок де 5 із 6 навичок можна вдосконалювати фізично, а не тільки з м’ячем. Завдяки цим вправам користувач може вдосконалити такі якості як удар, різні типи швидкості, витривалість, прижки, мобільність і багато чого іншого що є важливим для їх спорту, все це детально розподілено на вправи, як наприклад вправи на швидкість розділяють себе на такі вправи - стартова швидкість, максимальна швидкість, утримання швидкості на дистанції, а для стрибків - це стрибки у довжину та висоту. Все це розподілено по категоріях, таких як - футбол, баскетбол, волейбол. Зроблено це для того щоб користувач не робив все що може здатись потрібним, а робив саме те що є потрібним для обраного виду спорту.

1.2 Аналіз аналогів

Аналіз існуючих додатків для тренувань фізичних якостей та тренувань по видах спорту є важливим етапом у розробці нового додатку. Це дає розуміння, що вже реалізовано у аналогах, та дає інформацію про їхні недоліки та проблеми, щоб використати цю інформацію при розробці. Найбільш популярними додатками у обраній сфері є PlaySport, Turnfest Trainings App та PUMATRAC.

1.2.1 PlaySport

"PlaySport" - це додаток, розроблений компанією PlaySport Holdings Pty Ltd, який спрямований на сприяння активному способу життя через спортивні заходи та фізичну активність. Додаток надає користувачам можливість реєстрації на спортивні заходи, користувачі можуть шукати різноманітні тренування та змагання у таких категоріях, як футбол, баскетбол, теніс, йога тощо, які відбуваються в їхньому регіоні. Додаток надає можливість реєструватися на ці заходи прямо через нього, що надає змогу підібрати для себе тренування з комфортним розташуванням та часом проведення. Користувачі можуть створювати свої заходи у застосунку. вони можуть вказати деталі, такі як дату, час, місце проведення та інші умови. Користувачі можуть залишати відгуки та оцінки щодо організованих спортивних заходів, що допомагає іншим користувачам при виборі такого заходу. Додаток пропонує персоналізовані рекомендації спортивних заходів на основі інтересів та активностей користувача. Користувачі можуть звертатися до інших учасників для отримання порад щодо тренувань, за допомогою чату в додатку, але для цього користувач має додати в контакти іншого користувача, що робить проблемним спілкування з незнайомими користувачами.[3]

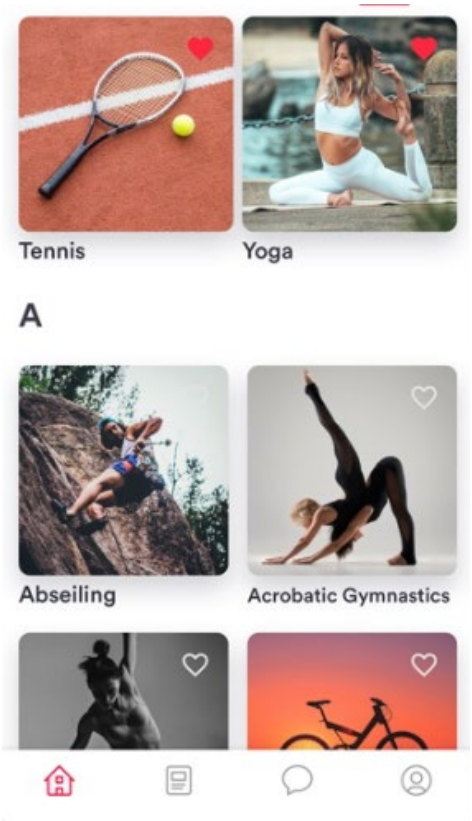


Рис. 1.1 Головний екран PlaySport

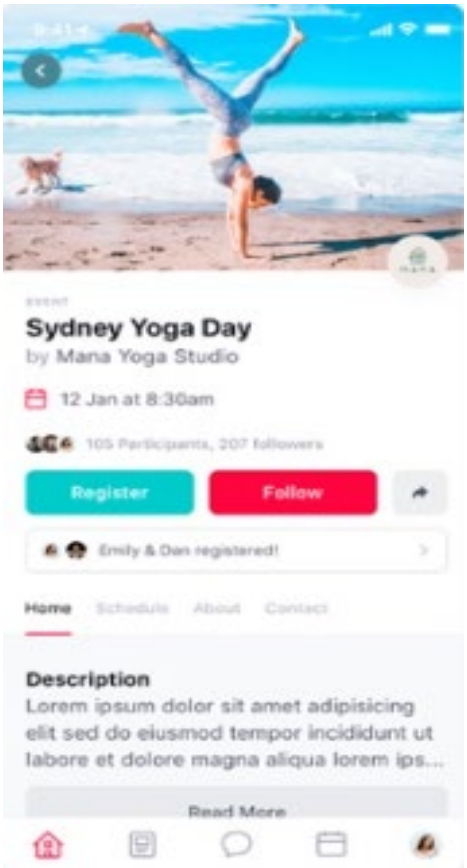


Рис. 1.2 Екран інформації до тренування

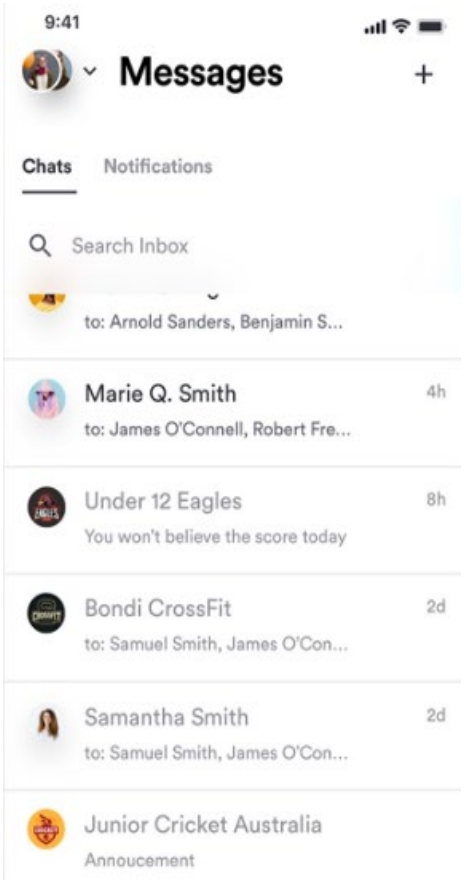


Рис 1.3 Екран чату між користувачами

Переваги:

– широкий вибір категорій;

– можливість спілкуватись з іншими користувачами;

– можливість організації власних тренувань.

Недоліки:

– додаток доступний тільки на iOS;

– проведення тренувань залежить від інших;

– спілкування з іншими користувачами у чаті може бути обмеженим;

1.2.2. Turnfest Trainings App

За допомогою цього додатку користувач може отримати доступ до вправ. У кожній категорії є вправи що вистроєні послідовно, задля забезпечення повноцінного тренування, однак на кожен вид спорту, є лише 2 таких комплексних тренувань. Також є можливість підрахування результатів

тренування та порівняння їх у проміжку однієї тренувальної сесії, зроблено це за допомогою доповнення STV Special Test Training, що було додано з останнім оновленням, також задля комфорту користувачів отримані результати можна конвертувати в оцінку. У додатку доступні вправи до таких видів спорту, як волейбол, хокей, баскетбол та теніс. Додаток доступний лише на німецькій мові, що може спричинити важкість у розумінні вправ. Додаток надає лише назви вправ, зазначаючи що всю інформацію користувач може знайти в інтернеті [4].

Рис. 1.4 Головний екран Turnfest Trainings App

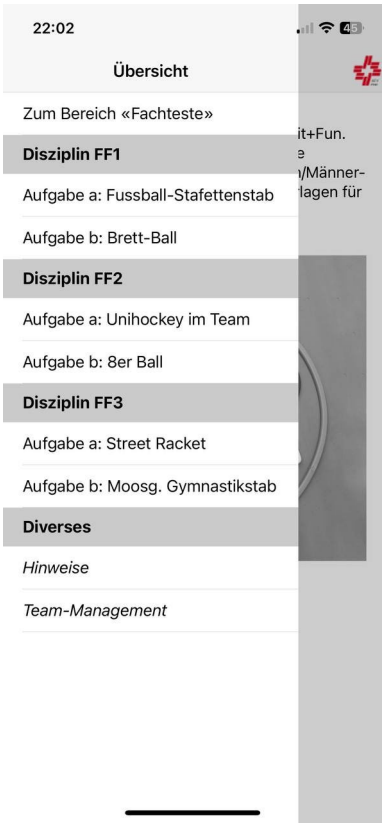


Рис. 1.5 Екран вибору вправ

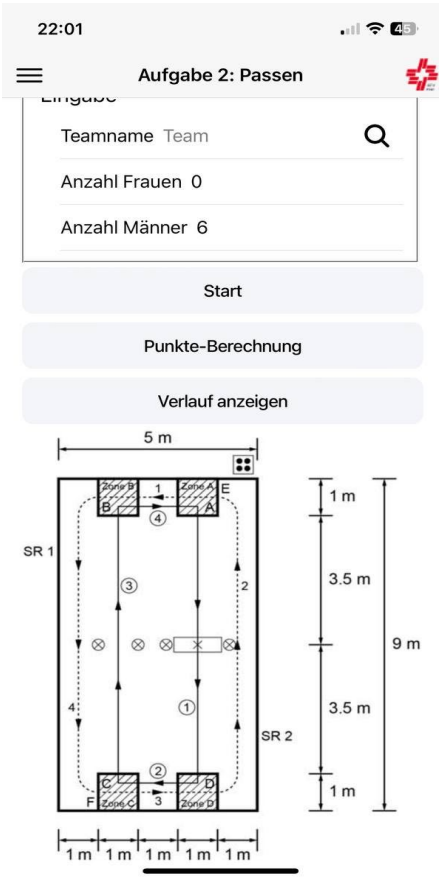


Рис. 1.6 Екран інформації до вправ

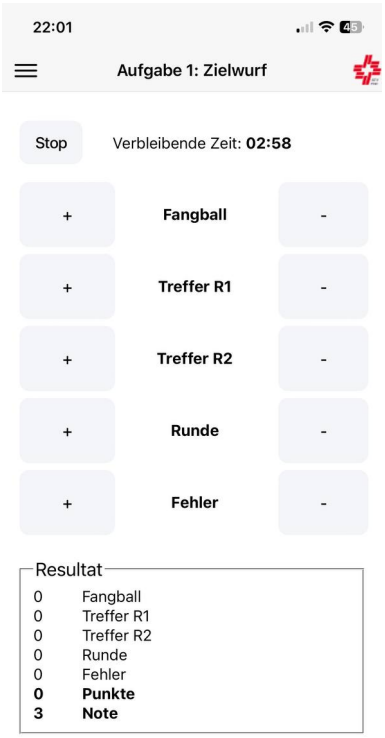


Рис. 1.7 Екран підведення результатів

Рис. 1.8 Конвертація результатів в оцінку

Переваги:

– можливість виконувати вправи як індивідуально так і з кимось;

– можливість ведення підрахунку результатів для порівняння;

– можливість змагань завдяки запису результатів.

Недоліки:

– додаток доступний тільки на німецькій мові;

– всього два тренування на категорію;

– опис вправ відсутній.

1.2.3. PUMATRAC

Pumatrac надає користувачам близько 100 індивідуальних програм тренувань. що містить контент з бігу, боксу, балету, йоги, пілатесу, тренування загальної сили та тренування для двох. Тренування у додатку складають професійні тренери, атлети, боксери, що дає користувачам впевненість у якості контенту. Компанія іноді проводить масові заходи для своїх користувачів, такі як наприклад біг для всіх охочих у зазначеному місці та у зазначений час, що дає користувачам змогу познайомитись з однодумцями, але у додатку про це розміщується лише рекламний банер, з посиланням на сторінку веб-сайту на котрому вже і буде інформація. Усі вправи розділяються на 3 рівні складності, початковий, середній та просунутий рівень тренувань. Також вправи надаються як короткі, до 20 хв, так і довготривалі від 30 хвилин. Вправи розраховані на те, щоб користувач міг виконувати їх без сторонніх атрибутів. До кожної вправи є відео зі спортсменом що її складав, це дає змогу користувачу дивитись на приклад того як саме потрібно виконувати вправу. Також додаток надає календар, в якому користувач може планувати свій графік тренувань. Ще є доступ до своїх плейлистів Spotify та Apple music з можливістю програвання музики не виходячи з додатку, та аудіо асистента, але застосувати це можна лише до керованих пробіжок, яких додано всього по одній до кожного рівня складності. Також у додатку є можливість реєстрації через пошту, facebook,

twitter та apple, але у більшості випадків реєстрація успішно проходить лише через facebook та apple на мобільних пристроях. [5]

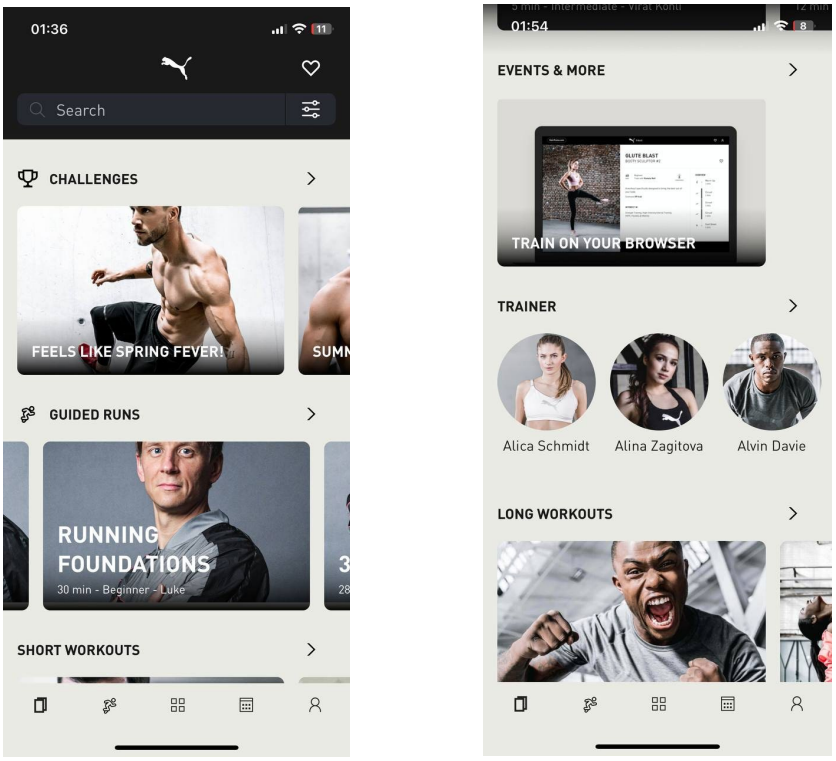


Рис. 1.9 Головний екран Pumatrac в різних режимах

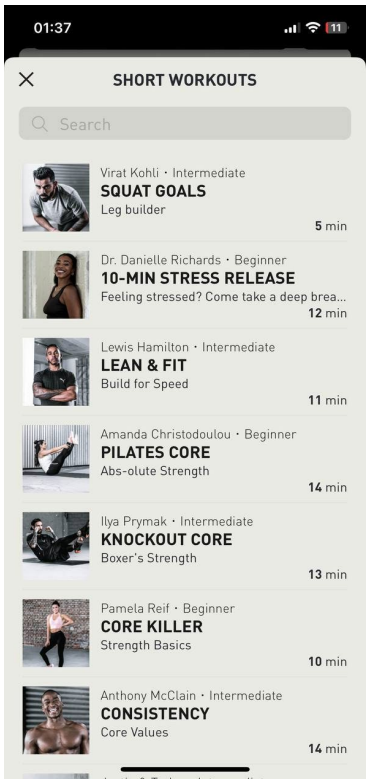


Рис. 1.10 Екран вибору вправ

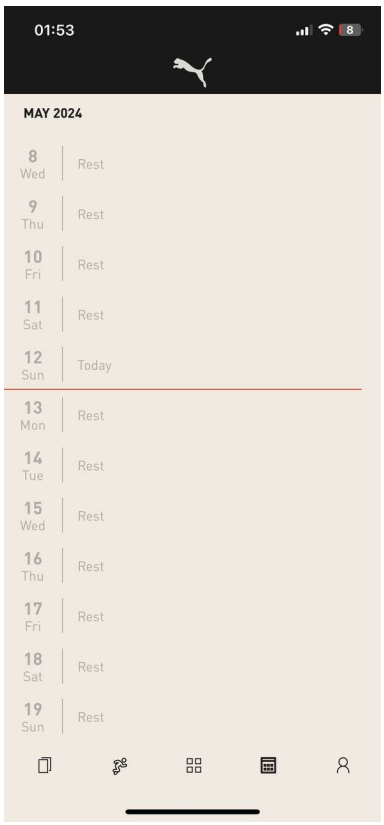


Рис. 1.11 Екран календаря

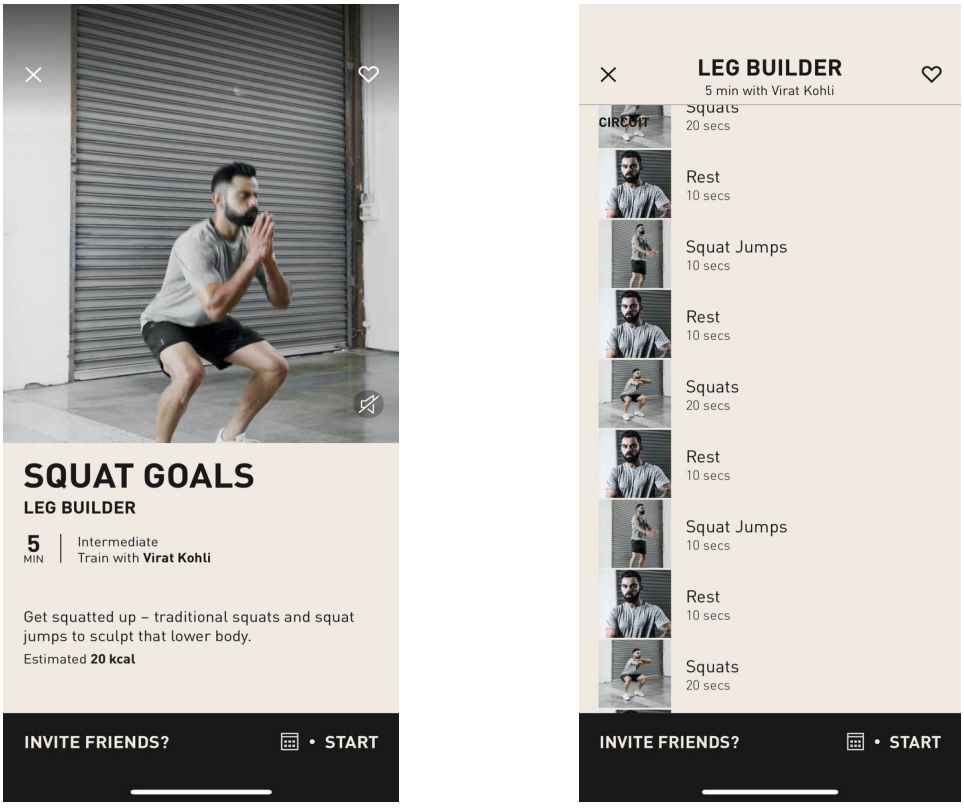


Рис. 1.12 Екрани деталей до вправи

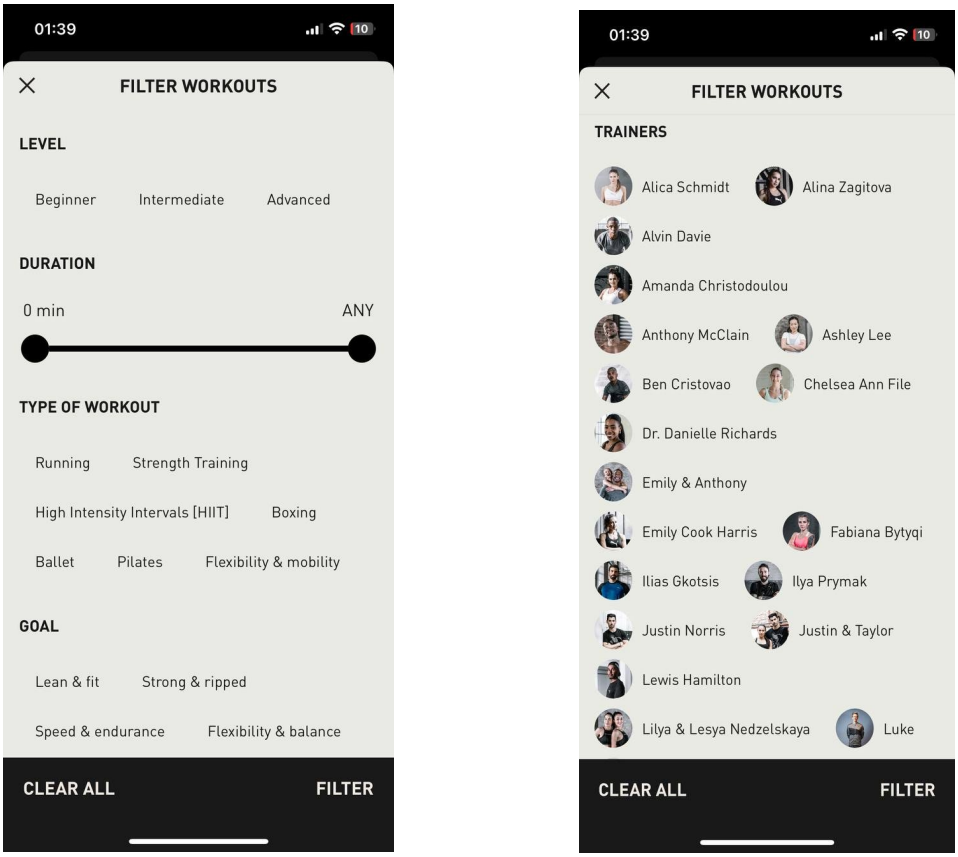


Рис.1.13 Система фільтрації вправ

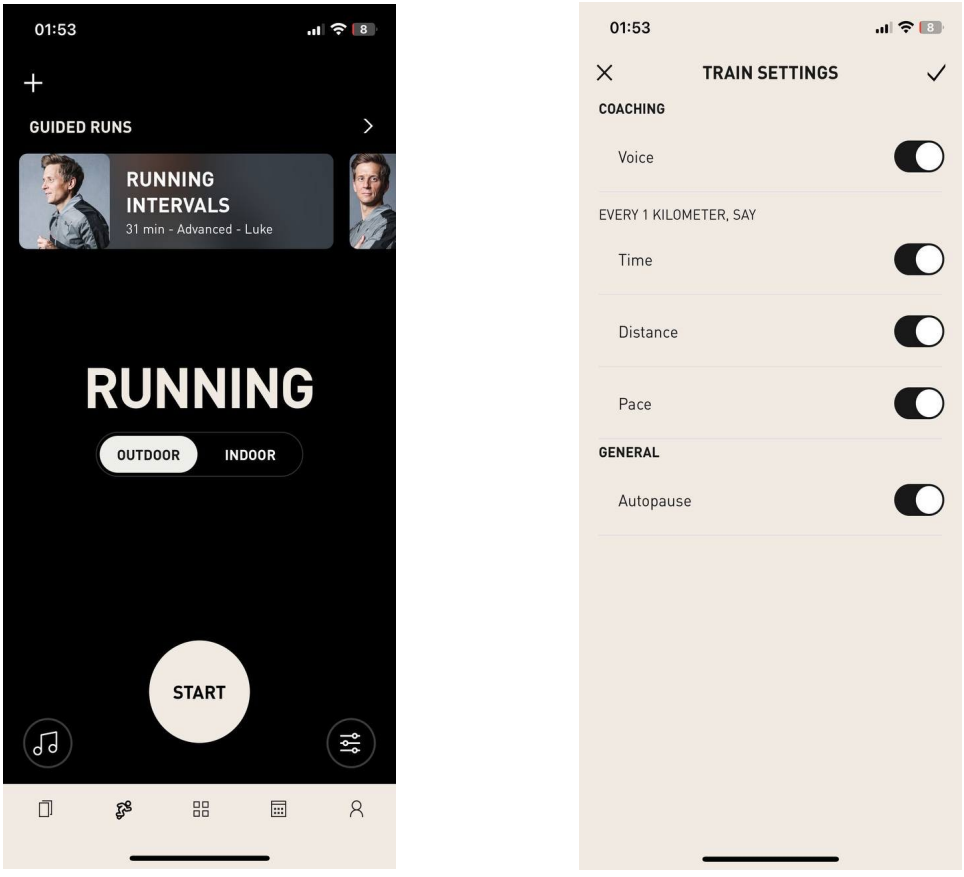


Рис. 1.14 Тренування керованих пробіжок

Переваги:

– широкий вибір вправ;

– різноманітність у виборі тренерів;

– відеоприклад виконання вправ;

– вправи розділені на категорії складності;

– можливість планування тренувань у календарі;

– система фільтрації вправ;

– категорії тренувань по тривалості.

Недоліки:

– проблеми з реєстрацією;

– для перегляду івентів потрібно переходити на сайт;

– у категоріях керованих пробіжок та челенджів всього по одній вправі на рівень складності;

– підключення музики та аудіо помічника доступно тільки для керованих пробіжок.

1.3 Зведений аналіз аналогів

Розглянуті додатки - PlaySport, Turnfest Trainings App та PUMATRAC - мають свої унікальні особливості та переваги у наданні тренувань користувачам.

PlaySport пропонує широкий вибір спортивних заходів та можливість спілкування з іншими користувачами, що сприяє соціальній взаємодії. Однак, обмеженість доступності додатка тільки на iOS, та залежність тренувань від інших учасників може стати перешкодою для багатьох користувачів.

Turnfest Trainings App надає можливість виконувати комплексні тренування з різних видів спорту, але ті факти що додаток лише на німецькій мові та відсутність описів вправ можуть зробити складним використання додатку для користувачів, які не володіють цією мовою. Також додаток пропонує всього 2 тренування на вид спорту, а до деяких потрібні ще й атрибути, це може зупинити користувачів у виборі цього додатку.

PUMATRAC надає користувачам приблизно 100 тренувань, які розроблені професійними тренерами та атлетами, а також можливістю планування тренувань у календарі, також до вправ є як текстовий так і відео опис. Однак, деякі недоліки, такі як проблеми з реєстрацією та обмеженість підключення музики та аудіо помічника, можуть вплинути на зручність використання додатку.

Результати зведеного аналізу застосунків для організації тренувань наведено в таблиці 1.1.

Таблиця 1.1

Результати зведеного аналізу застосунків для організації тренувань

Характеристика	PlaySport	Turnfest Trainings App	PUMATRAC
Вибір категорій	так	так	так
Мовна доступність	всі доступні системою	лише німецька	всі доступні системою
Платформи	iOS	iOS, Android	iOS, Android
Відео до вправ	ні	ні	так
Різноманітність тренувань	залежно від користувачів	8	близько 100
Опис вправ	ні	ні	так
Час на виконання	до всього тренування	немає	до кожної вправи
Залежність від атрибутів	залежить від організаторів	так	ні

2 ЗАСОБИ РОЗРОБКИ ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Дизайн та прототипування інтерфейсу з використанням Figma

Для дизайну інтерфейсу додатку вибір пав на онлайн сервіс Figma. Це безкоштовний онлайн сервіс для дизайну та прототипування створюваного продукту. Figma використовується для розробки різноманітних продуктів, від мобільних додатків і веб-сайтів до графічного дизайну та дизайну ігор. Він став популярним інструментом для дизайнерів та розробників завдяки зручному інтерфейсу, та можливостям. Figma пропонує можливості для створення макетів та прототипів, дозволяючи швидко втілювати ідеї. З Figma можна зручно працювати над усіма аспектами дизайну, створювати різні екрани, додавати взаємодіючі елементи, та в цілому можливість наповнення для користувача є інтуїтивно зрозумілою, так як користувач із запропонованих віджетів може прототипувати додаток як конструктор. Цей інструмент надає усі необхідні засоби для того, щоб довести інтерфейс до потрібного вигляду, та підготувати прототип для розробки. Також Figma дає змогу взаємодіяти з плагінами інших ресурсів, кастомними віджетами від ком'юніті, або ж навіть експорту створеного користувачем (рис.2.1).

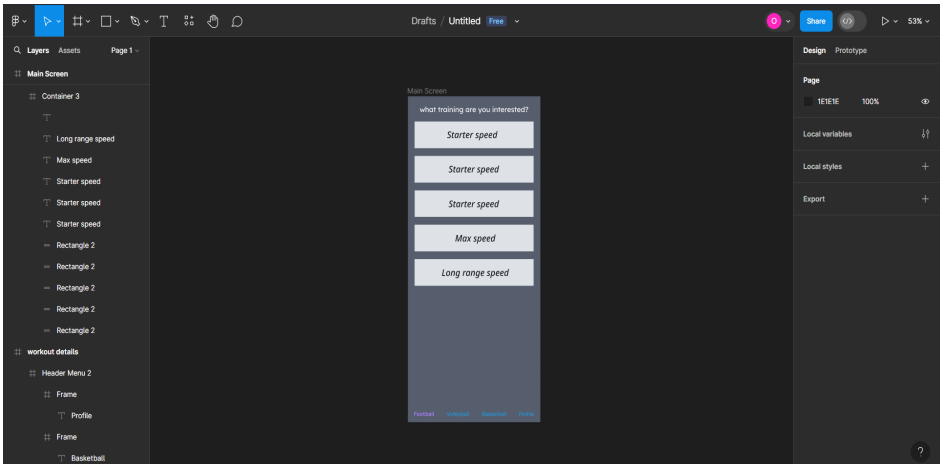


Рис. 2.1 Сторінка онлайн-сервісу Figma

2.2 Обґрунтування вибору середовища розробки PyCharm

Для розробки програмного забезпечення дуже важливо мати ефективні інструменти для того, щоб робота була якісною та продуктивною. І одним з таких інструментів є PyCharm - це інтегроване середовище розробки від JetBrains, створене спеціально для мови програмування Python. PyCharm допомагає швидко та зручно писати код, надаючи розширені можливості редагування, перевірки синтаксису та автодоповнення. Крім того, PyCharm інтегрується з віртуальними середовищами Python. Віртуальне середовище дозволяє мати окреме середовище для кожного проекту, що надає змогу встановлювати різні версії Python та пакети залежностей, не змішуючи їх з іншими проектами на пристрої. Це дозволяє уникнути конфліктів між версіями пакетів та забезпечити чисте середовище для кожного проекту, що допомагає зберігати проекти відокремлено один від одного. Однією з ключових переваг PyCharm є можливість ефективно відлагоджувати код та працювати з системами контролю версій, наприклад Git, що надає змогу працювати з репозиторіями Git, додавати, комітувати та пушити зміни прямо з середовища розробки PyCharm.

2.3 Обґрунтування вибору фреймворку Kivy

Для розробки інтерфейсу додатку був обраний Kivy. Kivy - це фреймворк для програмування на мові Python, який спрямований на розробку мобільних додатків та іншого програмного забезпечення. Він відкритий для використання та підтримує мультисенсорні можливості. Однією з його головних переваг є кросплатформеність, це означає, що він може працювати на різних операційних системах, включаючи Android, iOS, Windows, Linux і macOS. Крім того, Kivy має вбудований модуль для роботи з фотографіями та відео, що спрощує розробку додатків, які використовують ці медіа-ресурси. Проте, деякі більш досвідчені розробники вказують на обмеженість засобів стилізації та розмітки у Kivy порівняно з іншими фреймворками.[6-7]

2.3.1 Мова розмітки KV

Мова розмітки KV є інструментом, створеним спеціально для фреймворку Kivy, який дозволяє розробникам описувати візуальні елементи та їх взаємодію з іншими елементами за допомогою декларативного синтаксису. Основні переваги мови розмітки KV включають можливість розділення логіки програми від представлення інтерфейсу, що полегшує розробку та обслуговування коду. Крім того, вона надає розширені можливості для встановлення властивостей елементів та їх взаємодії, а також стилізації інтерфейсу за допомогою простих правил, схожих на CSS. Мова розмітки KV допомагає розробникам швидко та ефективно створювати інтерфейси користувача у фреймворку Kivy, забезпечуючи зрозумілий та легко редагований код (рис.2.2). [6]

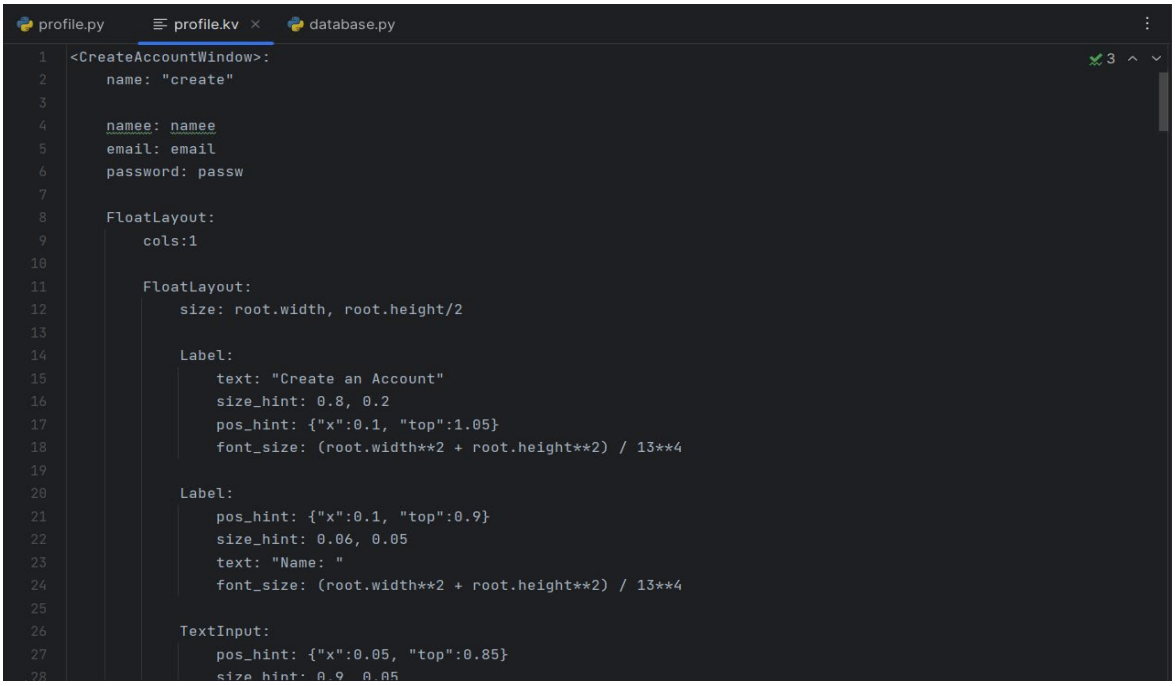


Рис. 2.2 Приклад коду мовою розмітки KV

2.4 Обґрунтування вибору мови програмування Python

Python - це високорівнева інтерпретована мова програмування, що відзначається своєю простотою, легкістю використання та зрозумілим кодом. Мова використовується в різних сферах, таких як - веб-розробка, машинне

навчання, штучний інтелект, data science, автоматизація, геймдев та мобільна розробка. Мова Python відрізняється зрозумілим синтаксисом, що схожий з англійською мовою. Python підтримує різні парадигми програмування, включаючи об'єктно-орієнтоване, функціональне та структурне програмування. Мова підтримується на різних операційних системах, що робить її доступною для розробників усіх платформ. Python є однією з найпопулярніших мов програмування завдяки своїй простоті вивчення та використання, а також активній спільноті розробників.[8]

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ЗАСТОСУНКУ З ПРОГРАМОЮ ТРЕНУВАНЬ ФІЗИЧНИХ ЯКОСТЕЙ ДЛЯ РІЗНИХ ВИДІВ СПОРТУ

3.1 Розробка архітектури застосунку

Загальна архітектура застосунку складається з наступних компонентів (рис.3.1):

- інтерфейс користувача(UI);
- логіка додатку(ядро);
- база даних.

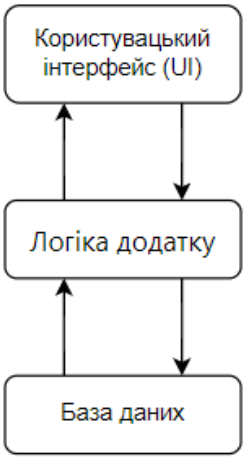


Рис. 3.1 Схема архітектури застосунку

Користувацький інтерфейс (UI).

Цей блок включає в себе всі елементи, що відображаються на екрані, такі як кнопки, текст, кнопка прокручування, поля введення та навіть кольори. Вони дозволяють користувачам взаємодіяти з додатком, виконуючи такі дії, як перехід між категоріями, перегляд вправ, створення облікового запису, перегляд профілю тощо.

Логіка додатку (ядро).

Цей блок містить основні функції програми, такі як управління тренуваннями, керування профілем та аутентифікація користувача. Він відповідає за перевірку даних, оновлення інформації та інші дії.

Доступ до даних (база даних).

Цей блок відповідає за зберігання даних. Він зберігає інформацію користувача, включаючи електронну пошту, пароль, ім’я, дату створення та дані про тренування і вправи, включаючи категорію, тип вправи, час на виконання та опис вправи.

Потік даних.

Стрілками визначений напрям потоку даних між різними блоками. Наприклад, дії користувача на UI викликають запити до ядра, який у свою чергу взаємодіє з базою даних, для отримання та збереження даних.

3.2 Моделювання вимог користувачів. Діаграма варіантів використання

Діаграма варіантів використання дозволяє уявити типи ролей та їх взаємодію із системою, зображує функціональні вимоги з точки зору користувача. Вимоги можуть описуватись текстом або у вигляді діаграми. Діаграми варіантів використання розробляються на стадії проектування системи та призначені для простого пояснення роботи системи. [9]

У контексті розроблюваної системи, актором виступає користувач, який взаємодіє з додатком (рис.3.2). Це може бути як людина, яка використовує інтерфейс програми, так і інша система, що інтегрується з нею, але для нашого випадку зосередимось на користувачі як на людині та основному акторі, оскільки саме він буде безпосередньо взаємодіяти з функціоналом програми. Враховуючи специфіку додатку, введення інших акторів наразі не передбачено. У діаграмі актор буде відображений саме як користувач.

Прецедент буде відображати, що саме має бути на екрані користувача у результаті взаємодії.

Система, відображена у вигляді прямокутної фігури та має назву у верхній лівій частині цієї фігури, є загальним відображенням того що створюється, у даному випадку, додаток для тренувань.

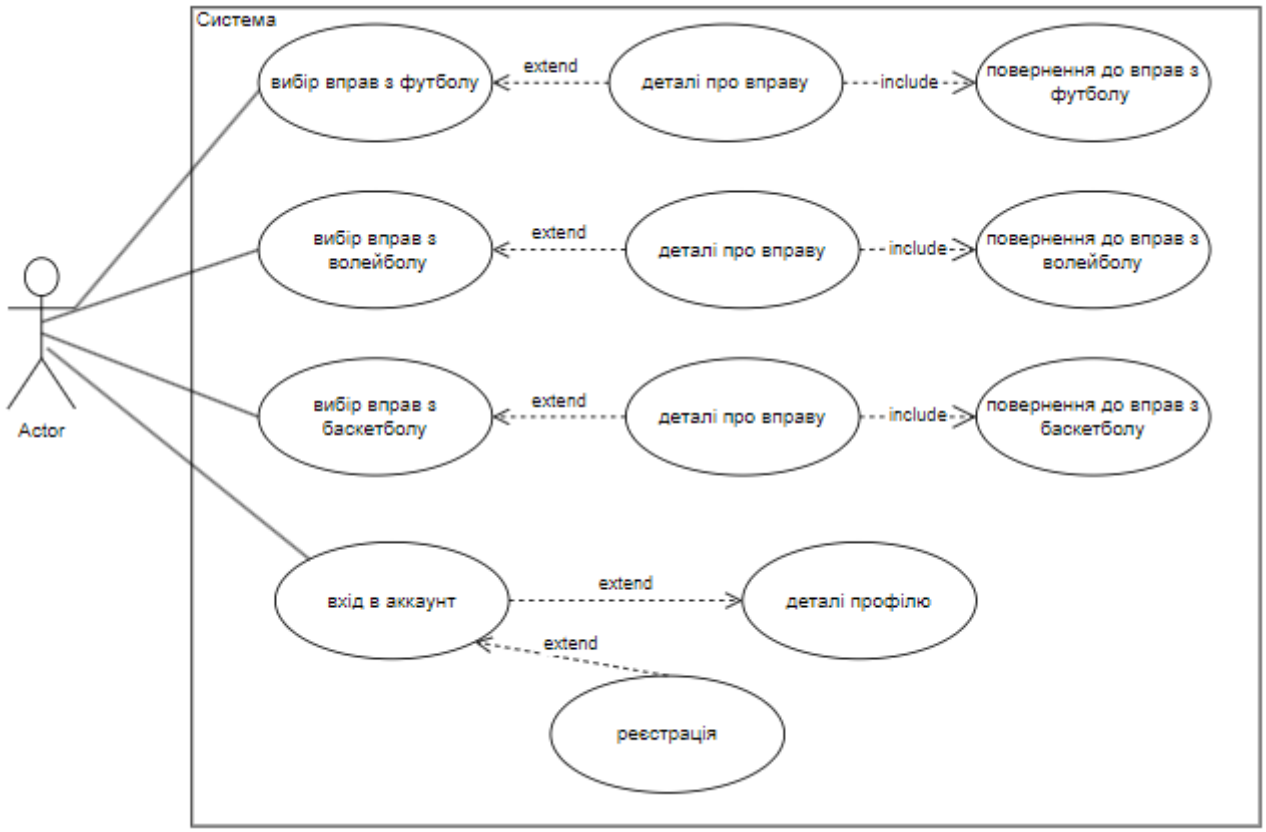


Рис. 3.2 Діаграма варіантів використання

Далі наведено опис прецедентів.

Як початковий екран для користувача є екран з вправами з футболу.

Варіанти дій на екрані зі списком вправ, а саме футбол, баскетбол, волейбол:

– перегляд списку вправ;

– перейти до опису вправи, натиснувши на назву обраної;

– перейти до екрану зі списком вправ за допомогою вибору категорії у навігаційному меню;

– прокручування списку вправ вверх та вниз за допомогою відповідної кнопки прокрутки у правій частині екрану, або за допомогою слайду у відповідному напрямку у області списку вправ.

Варіанти використання екрану з описом вправи:

– перегляд детального опису вправи(тип, категорія, час виконання, опис);

– перейти назад до екрану зі списком вправ, натиснувши відповідну кнопку;

– прокручування опису вправи вверх та вниз за допомогою відповідної кнопки прокрутки у правій частині екрану, або за допомогою слайду у відповідному напрямку у області списку вправ.

Екран входу в аккаунт:

– введення даних, а саме пошти та паролю у зазначені поля для введення;

– авторизація, натиснувши на відповідну кнопку після введення даних;

– перехід до екрану реєстрації;

– перейти до екрану зі списком вправ за допомогою вибору категорії у навігаційному меню.

Екран реєстрації:

– введення даних, а саме ім’я, пошта та пароль у зазначені поля для введення;

– реєстрація, натиснувши на відповідну кнопку після введення даних;

– перехід до екрану входу в аккаунт;

– перейти до екрану зі списком вправ за допомогою вибору категорії у навігаційному меню.

Екран деталей профілю(перехід при умові авторизованого користувача):

– перегляд інформації про аккаунт, а саме ім’я, пошта та дата реєстрації;

– вихід з акаунту, натиснувши на відповідну кнопку;

– перейти до екрану зі списком вправ за допомогою вибору категорії у навігаційному меню.

3.3 Діаграма класів

Розробка програмного забезпечення вимагає ретельного планування та проектування структури додатку. Одним із важливих етапів цього процесу є створення діаграми класів, яка дозволяє візуалізувати взаємозв'язки між різними класами системи та зрозуміти їхні атрибути та методи. Ця діаграма допомагає розробникам ефективно планувати і реалізовувати архітектуру додатку.

На рис. 3.3 представлена діаграма класів для додатку, який включає екрани реєстрації, авторизації, перегляду профілю, перегляду списку вправ та детальної інформації про вправу. Діаграма класів відображає структуру додатку та взаємозв'язки між різними компонентами.

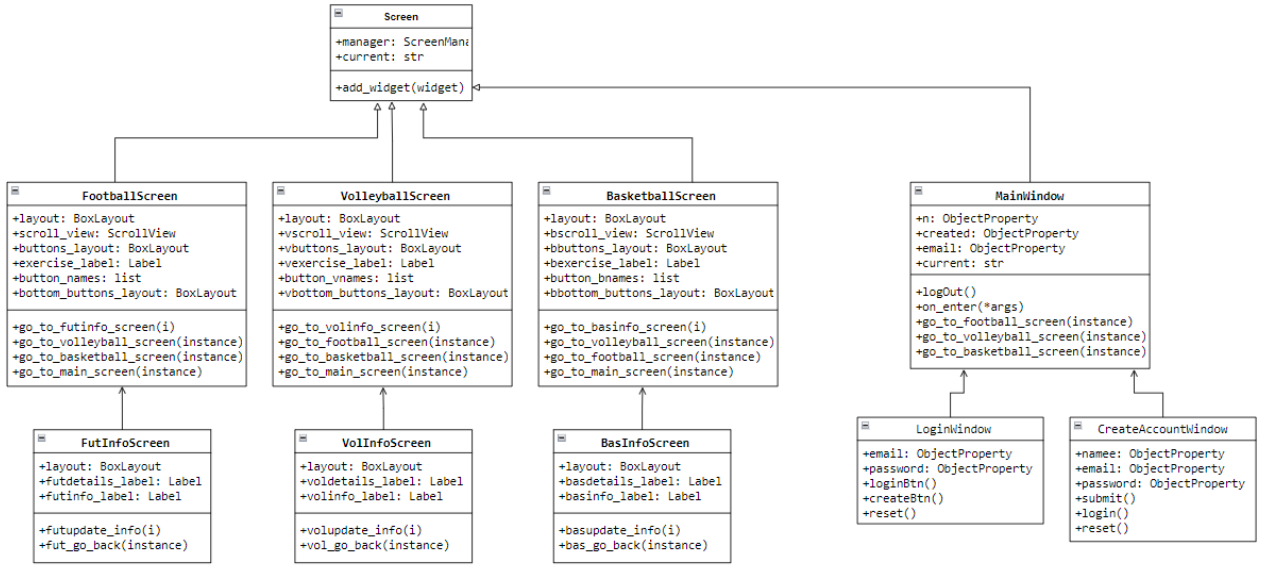


Рис. 3.3 Діаграма класів

Screen:

- Базовий клас для всіх екранів, який містить менеджер екранів (ScreenManager) та методи для додавання віджетів.

FootballScreen:

- Спадковий клас від Screen.
- Відповідає за відображення екрану з вправами для футболу.

- Має методи для переходу до екрану з деталями вправ, екрану волейболу, баскетболу та екрану профілю.

FutInfoScreen:

- Спадковий клас від Screen.
- Відповідає за відображення екрану з деталями вибраної вправи.
- Має методи для оновлення інформації про вправу та повернення до попереднього екрану.

VolleyballScreen:

- Спадковий клас від Screen.
- Відповідає за відображення екрану з вправами для волейболу.
- Має методи для переходу до екрану з деталями вправ, екрану футболу, баскетболу та екрану профілю.

VolInfoScreen:

- Спадковий клас від Screen.
- Відповідає за відображення екрану з деталями вибраної вправи для волейболу.
- Має методи для оновлення інформації про вправу та повернення до попереднього екрану.

BasketballScreen:

- Спадковий клас від Screen.
- Відповідає за відображення екрану з вправами для баскетболу.
- Має методи для переходу до екрану з деталями вправ, екрану волейболу, баскетболу та екрану профілю.

BasInfoScreen:

- Спадковий клас від Screen.
- Відповідає за відображення екрану з деталями вибраної вправи для баскетболу.
- Має методи для оновлення інформації про вправу та повернення до попереднього екрану.

CreateAccountWindow:

- Спадковий клас від Screen.
- Відповідає за створення акаунту користувача.
- Має методи для подання форми (submit), переходу до екрану логіну (login) та скидання форми (reset).

  LoginWindow:
- Спадковий клас від Screen.
- Відповідає за логін користувача.
- Має методи для логіну (loginBtn), переходу до створення акаунту (createBtn) та скидання форми (reset).

  MainWindow:
- Спадковий клас від Screen.
- Відповідає за відображення екрану деталей профілю.
- Має методи для виходу з акаунту (logOut), завантаження даних при вході на екран (on_enter), та переходу до екрану з вправами для футболу (go_to_football_screen).

3.4 Проектування інтерфейсу

На цій стадії розробки увага зосереджена на створенні зручного та естетично привабливого інтерфейсу, який надасть інтуїтивно зрозумілу навігацію та взаємодію. Елементи що були використані під час проектування інтерфейсу користувача у Figma:

– екран;

– контейнер;

– текстове поле;

– текстове поле для вводу тексту;

– рамка;

– прямокутник.

За допомогою цих елементів було візуалізовано наступні компоненти що будуть використовуватись у додатку:

– Вікно;

– Кнопки;

– Навігаційне меню;

– Контейнери;

– Текстові поля;

– Поля для вводу тексту;

– Кольори для всіх вище зазначених елементів.

Колірна схема додатку - це сучасна та мінімалістична комбінація кольорів. У додатку було використано сірий, фіолетовий та білий кольори, використовувались вони у різних відтінках відповідно до елементів.

Опис кольорів використаних у додатку:

Вікно – світло-сірий колір використовується як фоновий колір вікна додатку.

Кнопки зі списку вправ - для кнопок із назвою вправ був обраний темно-сірий колір.

Навігаційне меню:

– для навігаційного меню був обраний фіолетовий колір;

– при вибраній категорії колір насичений з темним відтінком;

– для не обраної категорії колір є тьмяним, насиченість фіолетового менша, а відтінок переходить ближче до сірого.

Кнопки екрану профілю, логін, реєстрація, вихід з акаунту, перехід між логіном та реєстрацією, були зроблені у одному кольорі – фіолетовий колір, що є трохи світлішим від кольору кнопки обраної категорії.

Текст:

– кольором тексту був обраний білий, він гарного поєднується з іншими використаними кольорами;

– також використовувався сірий колір тексту, на не обраних кнопках навігаційного меню.

Кнопка повернення до списку вправ – фіолетовий колір, що є трохи світлішим від кольору кнопки обраної категорії.

Дизайн графічного інтерфейсу розроблений із зручністю та інтуїтивною зрозумілістю в його основі. Елементи інтерфейсу розташовані так, щоб користувачі швидко знаходили що їм потрібно та безперешкодно взаємодіяли.

На рис. 3.4-3.5 наведено скріншоти змодельованого інтерфейсу у сервісі Figma.

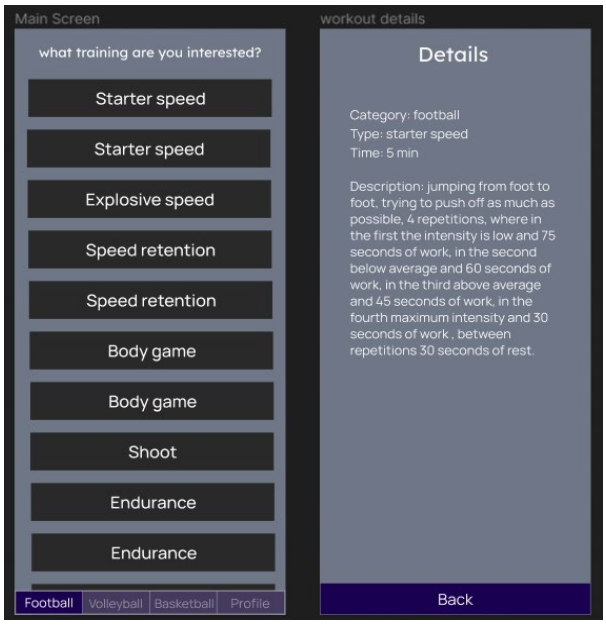


Рис. 3.4 Головний екран додатку та екран детальної інформації про вправу

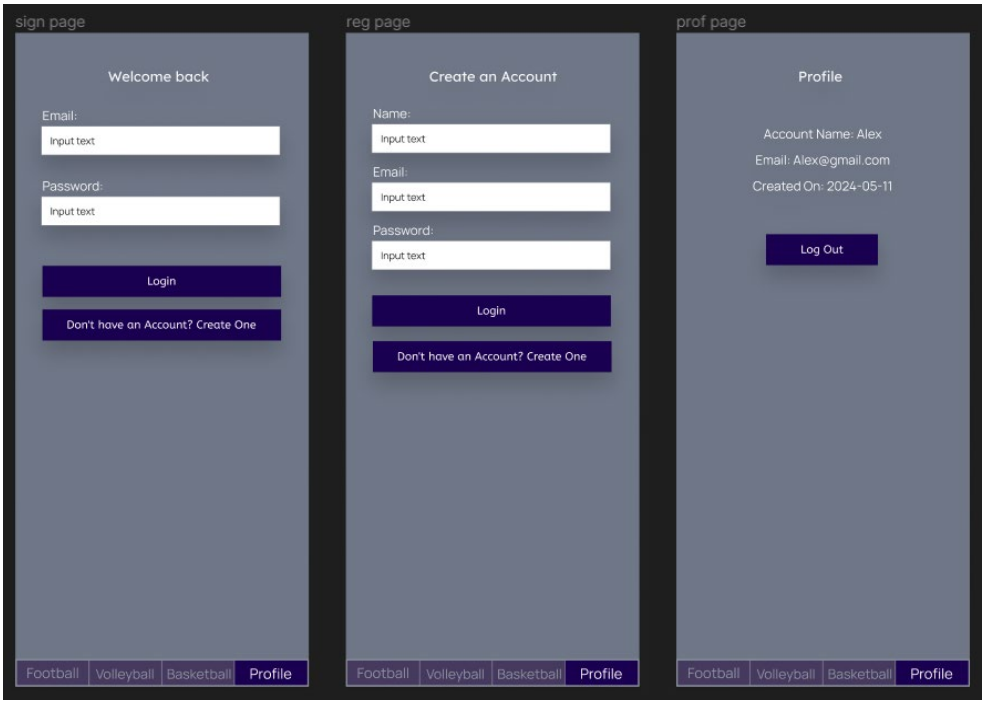


Рис. 3.5 Екрани реєстрації, авторизації та деталей профілю

3.5 Розробка екрану зі списком вправ

При розробці екрану зі списком вправ, було розроблено наступне:

- Три екрани що відповідають за категорії спорту, а саме футболу, баскетболу та волейболу. В Python, зокрема в бібліотеці Kivy, Screen є класом, що використовується для створення екранів у графічному інтерфейсі користувача. Класи FootballScreen, BasketballScreen та VolleyballScreen успадковують клас Screen, тобто вони є підкласом, які розширюють функціональність базового класу Screen. Це надає можливість визначити унікальну логіку та елементи інтерфейсу для кожного з екранів.
- Був створений вертикальний контейнер для кнопок списку вправ.
- До кожного з екранів був доданий список вправ, який представлений у вигляді вертикального списку кнопок, назви на яких відображають типи тренувань відповідно до обраної категорії. Кількість таких кнопок генерується відповідно до кількості тренувань у категорії.
- Був доданий метод, що виконує перехід до екрану детальної інформації про вправу, а викликається він при натисканні на одну з вправ.
- Для забезпечення зручного перегляду на екранах використовується функція прокрутки, яка дозволяє користувачу прокручувати екран у випадку, якщо завдань або матеріалу опису багато і вони не вміщуються в екран, зроблено це було для того щоб розмір елементів не змінювався через кількість елементів. Розміри ScrollView встановлюються таким чином, щоб він був розміром відповідно до вмісту і займав весь доступний простір екрану.
- Додано текстовий лейбл ‘Select your desired workout’ та розміщено у верхній частині екрану над кнопками вправ, для того щоб відзначити що на цьому екрані користувач може обрати вправу.

На рис. 3.6-3.8 наведено результат розробки екрану зі списком вправ.

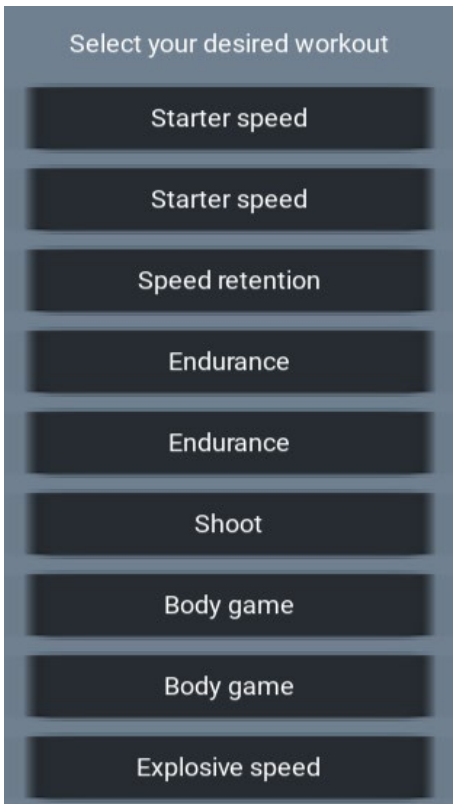


Рис. 3.6 Вибір вправ з футболу

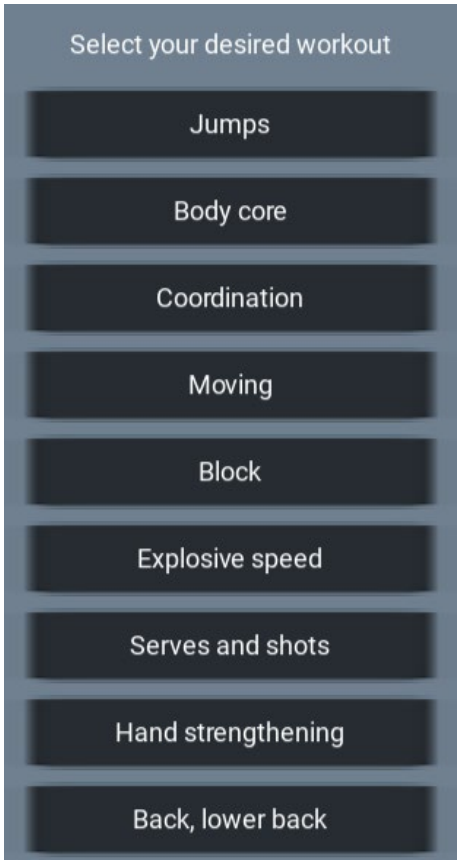


Рис. 3.7 Вибір вправ з волейболу

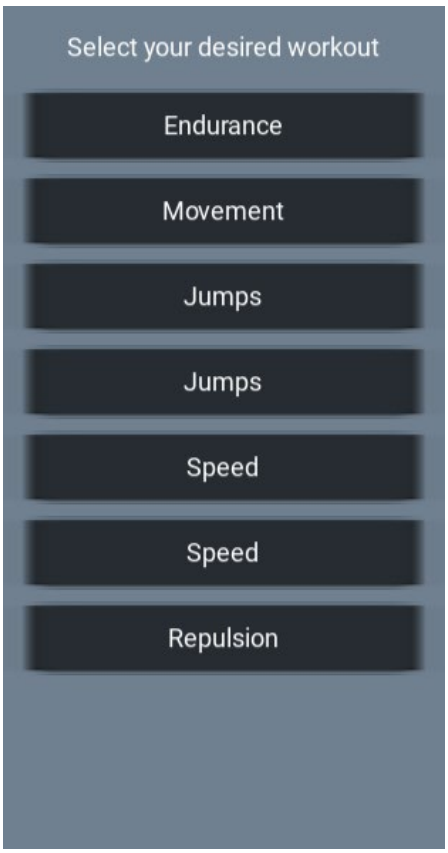


Рис 3.8 Вибір вправ з баскетболу

3.6 Розробка екрану з описом вправ

При розробці екрану з описом вправ, було розроблено наступне:

- Три екрани що відповідають за відображення деталей вправи у категоріях спорту, а саме футболу, баскетболу та волейболу. Названі ці класи були FutInfoScreen, VolInfoScreen та BasInfoScreen.
- Було додано текстовий лейбл з написом ‘Details’ та розміщено у верхній частині екрану, зроблено це для того, щоб показати користувачу що на цьому екрані деталі до вправи.
- Текстові лейбли для позначення даних що тут, а саме, про вид спорту, тип вправи, час на виконання та детальний опис вправи. Розміщені трохи нижче від ‘Details’ наступні текстові лейбли ‘Category’, ‘Type’, ‘Time’, та трохи нижче лейбл ‘Description’, зроблено для того щоб загальні дані та опис знаходились не у купі.

- До кожного із зазначених вище лейблів додаються дані після двокрапки, а саме вони додаються у поля для виводу даних про вид спорту, тип вправи, час на виконання та детальний опис вправи. Якщо даних забагато, то робиться перенос на наступну строку, особливо використовується під час виводу даних пояснення вправи.
- Було створено кнопку ‘Back’, та розміщено у нижній частині екрану, а саме прикріплено до нижнього краю екрану. Кнопка виконує функцію повернення до екрану зі списком вправ, до того з якого було відкрито екран. Реалізовано за допомогою того, що до кожної категорії є свій окремий екран деталей і відповідно окрема кнопка ‘Back’, тобто якщо користувач відкриває вправу з футболу, то з екрану футболу(FootballScreen) робиться перехід до екрану деталей(FutInfoScreen), а кнопка ‘Back’ повертає назад до FootballScreen.

На рис.3.9 наведено результат розробки екрану з описом вправ.

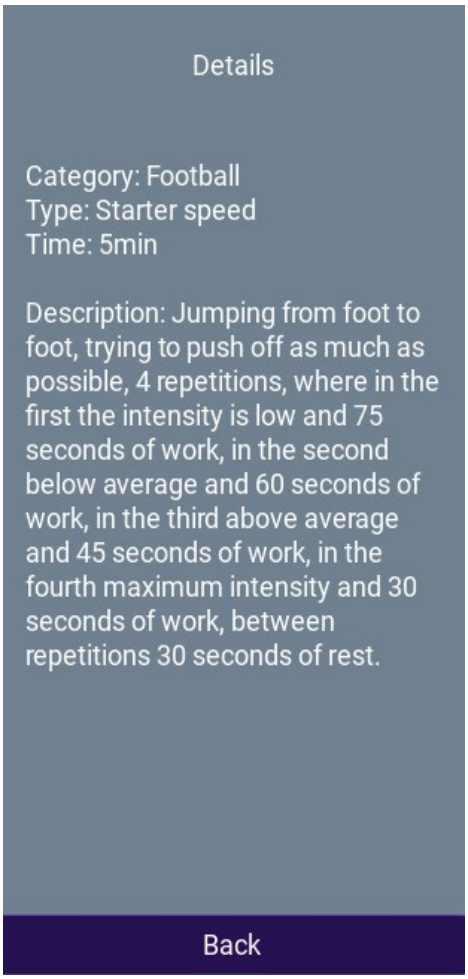


Рис. 3.9 Деталі до вправи

3.7 Розробка екрану профілю

На цьому етапі було створено три екрани, а саме екран реєстрації, авторизації та екран з деталями авторизованого профілю. До цих екранів був використаний окремий kv file для розмітки та дизайну, на відміну від екранів списку вправ та деталей до вправи, де ці налаштування були у коді відповідного екрану.

3.7.1 Реєстрація

При розробці екрану реєстрації, було розроблено наступне:

- Екран що відповідає за створення нового облікового запису, а саме CreateAccountWindow.
- Додано текстовий лейбл ‘Create an Account’, та прикріплено до верхньої частини екрану, з невеликим відступом від верхнього краю.
- Додано три текстових лейбли ‘Name’, ‘Email’, ‘Password’ для позначення що потрібно ввести у текстове поле.
- Додано три поля для введення тексту, та розміщено під кожен з лейблів відповідно.
- Оголошено змінні ‘namee’, ‘email’ і ‘password’, відповідно до кожного з полів для введення, та оголошено як об'єктні властивості, ініціалізовано значенням None. ObjectProperty - це тип властивості в Kivy, який використовується для зберігання посилання на будь-який об'єкт.
- Додана кнопка ‘Submit’ та розміщена трохи нижче полів вводу тексту.
- Додана кнопка ‘Already have an Account? Log In’ та розміщена трохи нижче кнопки ‘Submit’, по натисканню виконує перехід до екрану реєстрації.
- Доданий метод submit, що спрацьовує під час натискання кнопки ‘Submit’, для того щоб перевіряти, чи всі поля заповнені користувачем, а також правильність формату електронної пошти. Якщо умови виконуються, викликається метод для додавання нового користувача, після чого відбувається

перехід до екрану реєстрації. Якщо ж умови не виконані, виводиться повідомлення про неправильно введені дані.

- Додано спливаюче вікно, на якому виводиться повідомлення при невиконанні умов методу submit.
- Додано метод reset для очищення всіх текстових полів.

На рис. 3.10 наведено результат розробки екрану реєстрація.

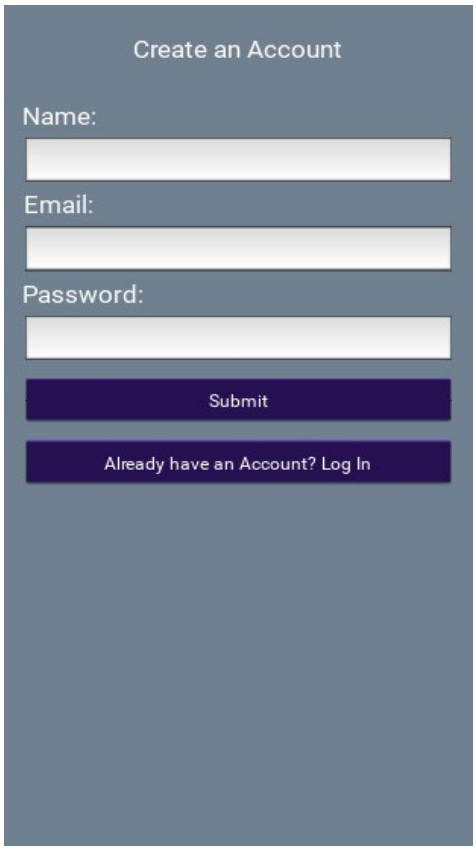


Рис. 3.10 Реєстрація профілю

3.7.2 Авторизація

При розробці екрану авторизації, було розроблено наступне:

- Екран що відповідає за вхід в систему, а саме LoginWindow.
- Додано текстовий лейбл ‘Welcome back’, який прикріплено до верхньої частини екрану з невеликим відступом від верхнього краю.
- Додано два текстових лейбли ‘Email’ та ‘Password’ для позначення, що потрібно ввести у відповідні текстові поля.

- Додано два текстових поля для введення електронної пошти та пароля, які розміщено під відповідними лейблами.
- Оголошено змінні email та password, відповідно до кожного з полів для введення, як об'єктні властивості та ініціалізовано значенням None.
- Додана кнопка ‘Login’ та розміщена трохи нижче полів вводу тексту. При натисканні виконує метод loginBtn.
- Додана кнопка ‘Don't have an Account? Create One’ та розміщена трохи нижче кнопки ‘Login’. При натисканні виконує метод createBtn, що здійснює перехід до екрану створення облікового запису.
- Доданий метод loginBtn, що спрацьовує при натисканні кнопки ‘Login’. Перевіряє правильність введених даних за допомогою методу validate. Якщо умови виконуються, встановлюється поточний користувач і здійснюється перехід до екрану інформації профілю. Якщо умови не виконані, викликається метод invalidLogin для відображення повідомлення про невдалу спробу входу.
- Доданий метод createBtn, що спрацьовує при натисканні кнопки ‘Create Account’. Він здійснює перехід до екрану створення облікового запису.
- Доданий метод reset використовується для очищення текстових полів електронної пошти та пароля.

На рис.3.11 наведено результат розробки екрану авторизація.

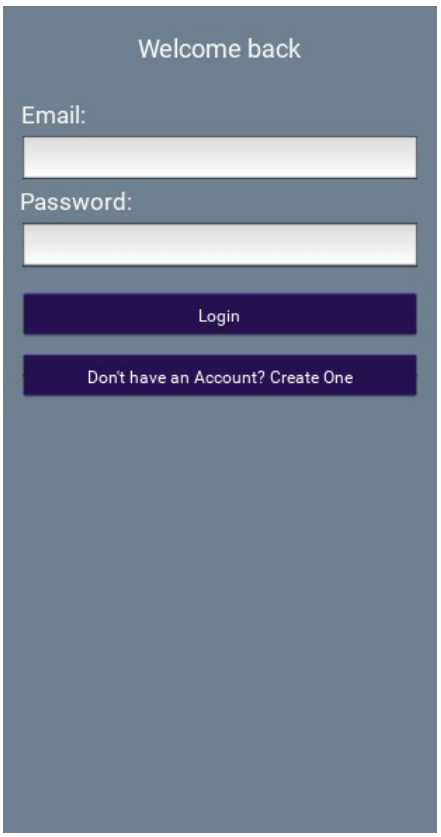


Рис. 3.11 Авторизація профілю

3.7.3 Інформація профілю

При розробці екрану інформації профілю, було розроблено наступне:

● Екран, що відповідає за відображення основної інформації про обліковий запис користувача, а саме MainWindow.

● Додано три текстових лейбли для відображення імені акаунту, електронної пошти та дати створення облікового запису. Ці лейбли будуть оновлюватися при вході користувача.

● Додано кнопку ‘Log Out’ для виходу з облікового запису, та розміщену в сердині екрану, під даними.

● Оголошено змінні n, created і email, які відповідають за лейбли для відображення імені акаунту, дати створення акаунту та електронної пошти відповідно. Всі змінні оголошені як об'єктні властивості та ініціалізовані значенням None.

● Оголошено змінну current, яка зберігає електронну пошту поточного користувача.

● Додано метод logOut, що спрацьовує при натисканні кнопки ‘Log Out’ і здійснює перехід до екрану входу.

● Додано метод on_enter, що спрацьовує при вході на екран. Він отримує дані про користувача з бази даних, а саме пароль, ім'я та дату створення облікового запису та оновлює відповідні лейбли з інформацією про користувача.

На рис.3.12 наведено результат розробки екрану інформація профілю.

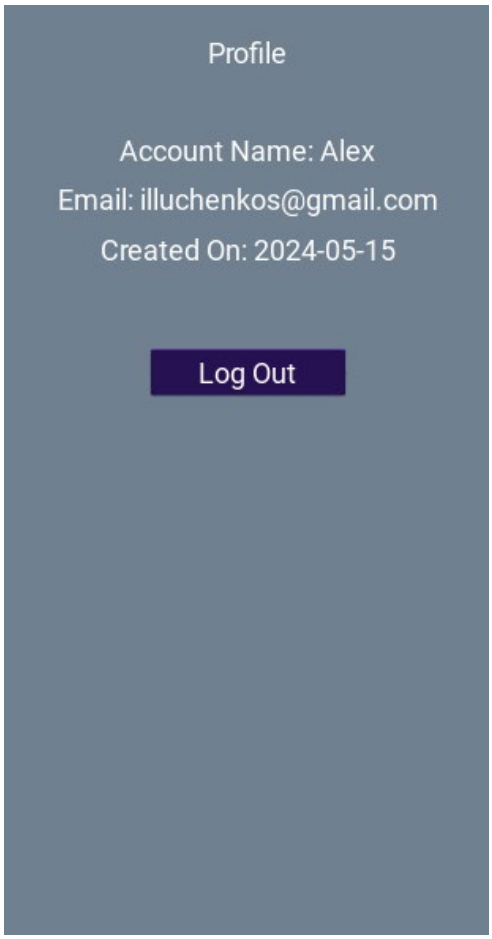


Рис. 3.12 Інформація профілю

Це створює повноцінну систему входу та реєстрації користувачів з можливістю перегляду інформації про користувача та переходу між цими екранами

3.8 Навігація між екранами

На цьому етапі було створено навігаційне меню, а також налаштована навігація між екранами вправ з футболу,волейболу, баскетболу та профілю.

3.8.1 Навігаційне меню

Навігаційне меню, це меню що складається з чотирьох кнопок ‘Football’, ‘Volleyball’, ‘Basketball’, ‘Profile’, зроблено воно для того щоб користувач міг по натиску на кожну з цих кнопок переходити до відповідної категорії.

Розроблено було наступне:

- Створено горизонтальний BoxLayout, який містить всі кнопки.
- Визначено список bottom_button_names, який містить назви всіх кнопок, які будуть додані внизу екрану, а саме ‘Football’, ‘Volleyball’, ‘Basketball’, ‘Profile’.
- Для кожної назви з списку створюється кнопка Button з певними параметрами стилю та розміру. Кожна кнопка має однаковий розмір, шрифт Roboto, курсивний стиль та специфічний колір фону і тексту. Для кнопки обраного екрану колір фону та тексту відрізняється.
- При натисканні на кнопку ‘Football’ виконується метод go_to_football_screen, що переміщує користувача на екран футболу.
- При натисканні на кнопку ‘Volleyball’ виконується метод go_to_volleyball_screen, що переміщує користувача на екран волейболу.
- При натисканні на кнопку ‘Basketball’ виконується метод go_to_basketball_screen, що переміщує користувача на екран баскетболу.
- При натисканні на кнопку ‘Profile’ виконується метод go_to_login_screen, що переміщує користувача на екран профілю.

На рис.3.13 наведено результат інтеграції цього меню до екрану.

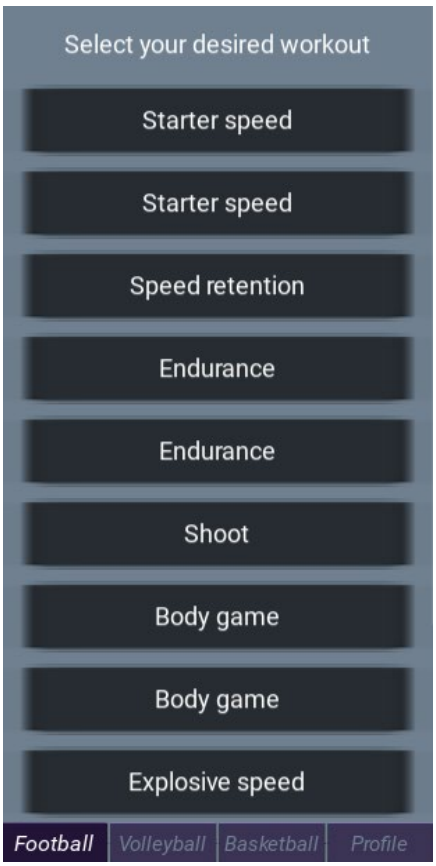


Рис.3.13 Екран футболу з доданим навігаційним меню

3.8.2 Screen manager

Клас ScreenManager, це основний клас, який використовується для керування переходами між різними екранами. За замовчуванням менеджер показує лише один екран одночасно. Атрибут current визначає поточний активний екран.

Розроблено було наступне:

- Для екрану футболу, метод go_to_futinfo_screen приймає параметр i, змінює поточний екран на futinfo та викликає метод futupdate_info екрану futinfo, передаючи параметр i.
- Для екрану волейболу, метод go_to_volinfo_screen приймає параметр i, змінює поточний екран на volinfo та викликає метод volupdate_info екрану volinfo, передаючи параметр i.

- Для екрану баскетболу, метод go_to_basinfo_screen приймає параметр i, змінює поточний екран на basinfo та викликає метод basupdate_info екрану basinfo, передаючи параметр i.
- Метод go_to_football_screen приймає параметр instance (віджет, який викликав метод), та Змінює поточний екран на football.
- Метод go_to_volleyball_screen приймає параметр instance (віджет, який викликав метод), та Змінює поточний екран на volleyball.
- Метод go_to_basketball_screen приймає параметр instance (віджет, який викликав метод), та Змінює поточний екран на basketball.
- Метод go_to_login_screen приймає параметр instance (віджет, який викликав метод), та Змінює поточний екран на profile.

3.9 Перспективи та можливості удосконалення додатку. Створення програми тренування користувачем

Ідея до удосконалення за допомогою створення програми тренувань користувачем полягає у тому, що користувач зможе створювати тренування із вправ що надаються у додатку. Для цього потрібно додати до додатку можливість створення плейлистів, та додавання до них вправ. Це надасть користувачам змогу робити собі програму тренувань і якщо користувач тренує наприклад швидкість і використовує для цього вправи на розвинення стартової швидкості, утримання швидкості на дистанції та витривалість, то він зможе додати всі вправи до цього плейлисту і відповідно назвати його. Це надасть користувачам зручності, особливо буде використовуватися тими, хто працює над покращенням чогось конкретного, і постійно виконує одні й ті самі вправи, так як вони зможуть відкрити саме ці вправи замість того щоб відкривати їх зі списку категорії.

3.10 Можливості використання в застосунку штучного інтелекту

Ідея полягає у використанні штучного інтелекту для аналізу тренувань користувачів, надання їм рекомендацій щодо покращення тренувальних програм та автоматичного формування персоналізованих тренувальних комплексів на основі їх інтересів. Для цього потрібна інтеграція моделі штучного інтелекту у додаток. Завдяки легкій інтеграції через API, можна використовувати функціонал моделі без необхідності змінювати архітектуру додатку. Дотримуючись наданої документації, можна швидко й ефективно інтегрувати модель у додаток.

У додатку для тренувань можна використати механізм рекомендацій. Зробити це можна шляхом аналізу відвіданих користувачами тренувань. Штучний інтелект буде використовувати дані про відвідані тренування користувачем, та завдяки цим даним аналізувати які саме вправи цікавлять користувача і надати йому рекомендації із інших вправ схожих типів. Цей алгоритм може бути навчений з часом, та пізніше наводити рекомендації не тільки за параметрами вправ, а ще й використовуючи інформацію про те які вправи виконували інші користувачі, у парі з розглянутою вправою.

Також можна використовувати штучний інтелект для надання користувачам комплексних тренувань. Комплексне тренування - це набір з різних вправ у одному тренуванні. Завдяки штучному інтелекту можна аналізувати останні відвідані користувачами вправи, та на основі цього пропонувати персоналізовані комплексні тренування із набору найчастіше відвідуваних вправ. Для аналізу можна використовувати не тільки дані про вправи, а також і інформацію про час проведений за їх виконанням. Це надасть інформацію про те, скільки часу користувач може виділити на проведення тренування, що допоможе обмежити тривалість комплексного тренування до імовірно виділеного користувачем часу на нього.

Також у додатку для тренувань, можна використовувати штучний інтелект для проведення аналізу дій користувачів у додатку, це може знадобитись для

розуміння того що більш затребувано і що навпаки не використовується. Ці дані можна використовувати для того щоб зрозуміти які саме вправи та категорії користуються попитом, що можна додати або змінити для того щоб зберегти зацікавленість з боку користувачів.

4 ТЕСТУВАННЯ ДОДАТКУ

На цьому етапі розробки було зроблено тестування додатку, для того щоб перевірити, чи не виникне помилок у роботі додатку, під час користування користувачем.

Для тестування функціональності додатку були застосовані методи "чорного ящика". Цей метод фокусується на перевірці функціональних можливостей програми відповідно до вимог, без аналізу внутрішнього коду. [10] У тест-кейсах використовуються вхідні дані, характерні для дій звичайного користувача, а результати порівнюються з очікуваними. Кожен тест-кейс включає детальний опис дій, що потрібно виконати, та тестові дані, які задовольняють вимогам покриття функціональних можливостей програми.

Структура кожного тест-кейсу включає наступне:

- Test Case ID - номер тест-кейсу;
- Test Case Title - назва тест-кейсу;
- Test Data - дані, що потрібні для виконання тесту;
- Expected Result - очікуємий результат тестування;
- Status - позначення результату тестування(Успішно або Невдало).

В таблиці 4.1. наведені тест-кейси для функціонального тестування додатку.

Таблиця 4.1

Тест-кейси та результати функціонального тестування

Test Case ID	Test Case Title	Test Steps	Test Data	Expected Result	Status
1	Перехід до екрану вправ для футболу	1. Натиснути кнопку "Football" в меню внизу екрану.	-	Відкривається екран з вправами для футболу.	Успішно
2	Перехід до екрану вправ для баскетболу	1. Натиснути кнопку "Basketball" в меню внизу екрану.	-	Відкривається екран з вправами для баскетболу.	Успішно
3	Перехід до екрану вправ для волейболу	1. Натиснути кнопку "Volleyball" в меню внизу екрану.	-	Відкривається екран з вправами для волейболу.	Успішно
4	Прокрутка списку вправ вверх-вниз	1. Відкрити екран з вправами для футболу. 2.Прокрутити список вверх. 3.Прокрутити список вниз.	-	Список вправ коректно прокручується вверх та вниз.	Успішно
5	Прокрутка списку вправ вверх-вниз	1. Відкрити екран з вправами для баскетболу. 2.Прокрутити список вверх. 3.Прокрутити список вниз.	-	Список вправ коректно прокручується вверх та вниз.	Успішно

Продовження таблиці 4.1

Test Case ID	Test Case Title	Test Steps	Test Data	Expected Result	Status
6	Прокрутка списку вправ вверх-вниз	1. Відкрити екран з вправами для волейболу. 2.Прокрутити список вверх. 3.Прокрутити список вниз.	-	Список вправ коректно прокручується вверх та вниз.	Успішно
7	Відкриття детальної інформації про вправу	1. Відкрити екран з вправами для футболу. 2.Натиснути на будь-яку вправу зі списку.	-	Відкривається екран з детальною інформацією про вибрану вправу.	Успішно
8	Відкриття детальної інформації про вправу	1. Відкрити екран з вправами для волейболу. 2.Натиснути на будь-яку вправу зі списку.	-	Відкривається екран з детальною інформацією про вибрану вправу.	Успішно
9	Відкриття детальної інформації про вправу	1. Відкрити екран з вправами для баскетболу. 2.Натиснути на будь-яку вправу зі списку.	-	Відкривається екран з детальною інформацією про вибрану вправу.	Успішно
10	Повернення назад до категорії з екрану детальної інформації	1. Відкрити екран з детальною інформацією про вправу. Натиснути кнопку "Back".	-	Користувач повертається до екрану зі списком вправ.	Успішно
11	Перехід до екрану профілю	1. Натиснути кнопку "Profile" в меню внизу екрану.	-	Відкривається екран входу в профіль.	Успішно

Продовження таблиці 4.1

Test Case ID	Test Case Title	Test Steps	Test Data	Expected Result	Status
12	Перехід до екрану реєстрації	1. Натиснути кнопку "Profile". 2.Натиснути кнопку "Don't have an Account? Create One".	-	Відкривається екран реєстрації профілю.	Успішно
13	Реєстрація з пустими полями	1. Відкрити екран реєстрації. 2.Натиснути кнопку "Submit" з пустими полями.	-	Показується повідомлення про необхідність заповнити всі поля.	Успішно
14	Реєстрація з некоректною електронною поштою	1. Відкрити екран реєстрації. Ввести некоректну електронну пошту (наприклад, "alex.com"). Натиснути кнопку "Submit".	Некоректна електронна пошта: "alex.com"	Показується повідомлення про необхідність введення коректної електронної пошти.	Успішно
15	Успішна реєстрація	1. Відкрити екран реєстрації. Ввести коректну електронну пошту та інші дані. Натиснути кнопку "Submit".	Коректна електронна пошта та інші дані	Успішна реєстрація та перехід до екрану деталей профілю.	Успішно

Продовження таблиці 4.1

Test Case ID	Test Case Title	Test Steps	Test Data	Expected Result	Status
16	Авторизація з некоректними даними	1. Відкрити екран входу. 2.Ввести некоректну електронну пошту або пароль. 3.Натиснути кнопку "Login".	Некоректна електронна пошта або пароль	Показується повідомлення про некоректні дані для входу.	Успішно
17	Успішна авторизація	1. Відкрити екран входу. 2.Ввести коректну електронну пошту та пароль. 3.Натиснути кнопку "Login".	Коректна електронна пошта та пароль	Успішна авторизація та перехід до екрану деталей профілю.	Успішно
18	Вихід з профілю	1. Авторизуватись у профілі. 2.Натиснути кнопку "Log Out".	-	Вихід з профілю та повернення до екрану входу.	Успішно
19	Переміщення між екранами категорій спорту	1. Натиснути кнопку "Football" в меню внизу екрану. 2.Натиснути кнопку "Basketball" в меню внизу екрану. 3.Натиснути кнопку "Volleyball" в меню внизу екрану.	-	Відповідні екрани з вправами відкриваються коректно.	Успішно

Продовження таблиці 4.1

Test Case ID	Test Case Title	Test Steps	Test Data	Expected Result	Status
20	Переміщення до екрану профілю (Неавторизований користувач)	1. Натиснути кнопку "Profile" в меню внизу екрану.	-	Відкривається екран входу профілю.	Успішно
21	Переміщення до екрану профілю (Авторизований користувач)	1. Авторизуватись у профілі. 2.Натиснути кнопку "Profile" в меню внизу екрану.	-	Відкривається екран деталей профілю.	Успішно

Завдяки проведеним тестуванням, додаток було перевірено на відповідність функціональним вимогам та забезпечено його відповідність розробленим вимогам.

ВИСНОВКИ

1. Проведено аналіз обраної предметної області в сфері тренувань фізичних якостей. Визначено ключові потреби користувачів.

2. Проведено аналіз додатків-аналогів у сфері тренувань фізичних якостей, було визначено їх ключові функції, переваги та недоліки, які були використані при розробці додатку для тренувань.

3. Проведено аналіз засобів розробки та проектування програмного забезпечення. Було використано їх при розробці додатку для тренувань.

4. Визначено основні вимоги та проведено їх моделювання з використанням діаграми варіантів використання.

5. Розроблено додаток для тренувань фізичних якостей у різних видах спорту, який надає можливість користувачам переглядати та обирати вправи і тренуватись тренуватись завдяки опису до них.

6. Досліджено перспективи та можливості до удосконалення додатку з тренувань фізичних якостей, вказано варіанти до покращення в майбутньому.

7. Створено тестові сценарії, які охоплюють розроблені функціональні вимоги, і проведено перевірку коректності роботи програмного забезпечення.

8. Робота пройшла апробацію:

Іллюченко О.С., Золотухіна О.А. Порівняння Python фреймворків для розробки інтерфейсу мобільного додатку. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ». Збірник тез. К.: ДУІКТ, 2024р. С.427-429.

Іллюченко О.С., Золотухіна О.А. Розгляд можливостей застосування штучного інтелекту в мобільному додатку для тренувань фізичних якостей у різних видах спорту для аналізу запитів користувачів і надання рекомендацій. Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. К.: ДУІКТ, 2024 р. С.53-54.

ПЕРЕЛІК ПОСИЛАНЬ

1. Revealing average screen time statistics for 2024. *Backlinko*. URL: https://backlinko.com/screen-time-statistics.
2. KSL Ukraine. Офіційний сайт. *KSL Ukraine. Офіційний сайт*. URL: https://ksl.co.ua/.
3. PlaySport by PlaySport Holdings Pty Ltd. *AppAdvice*. URL: https://appadvice.com/app/playsport/1317036368.
4. Die Fachtest Trainings-App. *STV FSG - Schweizerischer Turnverband - STV*. URL: https://www.stv-fsg.ch/de/sportarten/turnen-erwachsene/fachteste/die-fachtest-trainings-app-self.html.
5. Pumatrac. *PUMATRAC*. URL: https://pumatrac.puma.com/.
6. Kivy: cross-platform python framework for NUI. *Kivy: Cross-platform Python Framework for GUI apps Development*. URL: https://kivy.org/.
7. Python for mobile app development in 2023 – kivy vs. beeware. *AI-enhanced digital product development · Monterail*. URL: https://www.monterail.com/blog/python-for-mobile-app-development.
8. Особливості та переваги Python. *SpaceLab*. URL: https://spacelab.ua/articles/osobennosti-i-preimushestva-python/.
9. Каграманова Ю. Як будувати uml-діаграми. *dou*. URL: https://dou.ua/forums/topic/40575/.
10. Рівні і методи тестування ПЗ. *Avada-media*. URL: https://avada-media.ua/ua/services/urovni-i-metody-testirovaniya-po/.
11. Welcome to python.org. *Python.org*. URL: https://www.python.org/.
12. Іллюченко О.С., Золотухіна О.А. Порівняння Python фреймворків для розробки інтерфейсу мобільного додатку. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ». Збірник тез. К.: ДУІКТ, 2024р. С.427-429.

13. Іллюченко О.С., Золотухіна О.А. Розгляд можливостей застосування штучного інтелекту в мобільному додатку для тренувань фізичних якостей у різних видах спорту для аналізу запитів користувачів і надання рекомендацій. Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. К.: ДУІКТ, 2024р. С.53-54.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«Розробка додатку з програмою тренувань фізичних якостей, для різних видів спорту мовою Python»

Виконав студент 4 курсу
групи ПД-42
Іллюченко Олександр Сергійович
Керівник роботи
К.т.н., доц., доцент кафедри ІПЗ Золотухіна Оксана Анатоліївна

Київ – 2024

1

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – підтримка процесу тренування фізичних якостей у обраному виді спорту засобами мобільного додатку, створеного мовою програмування Python.

Об`єкт дослідження – тренування фізичних якостей у обраному виді спорту.

Предмет дослідження – програмне забезпечення для тренування фізичних якостей у обраному виді спорту.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Аналіз обраної предметної області в сфері тренувань фізичних якостей.
2. Порівняння існуючого програмного забезпечення, для тренувань фізичних якостей у різних видах спорту.
3. Визначення та моделювання вимог до програмного забезпечення для тренувань.
4. Розробка архітектури додатку для тренувань.
5. Проектування та розробка програмного забезпечення для тренувань.
6. Тестування розробленого додатку для тренувань.

3

АНАЛІЗ АНАЛОГІВ

Характеристика	PlaySport	Turnfest Trainings App	PUMATRAC	Розроблений додаток
Вибір категорій	так	так	так	так
Локалізація	англійська мова	німецька мова	7 мов	визначається мовою системи
Платформи	ios	ios, android	ios, android	ios, android
Різноманітність тренувань	залежно від користувачів	8	близько 100	35
Опис вправ	ні	ні	так	так
Час на виконання	до всього тренування	немає	до кожної вправи	до кожної вправи
Розділення вправ по типу	ні	ні	так	так
Регіональна доступність	8 країн	усі країни	усі країни	усі країни
Можливість роботи в офлайн режимі	ні	так	ні	так
Розділення вправ по видам спорту	ні	ні	ні	так

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

1. Реєстрація користувачів
2. Авторизація користувачів
3. Відображення деталей профілю
4. Вихід з аккаунту
5. Навігація між екранами
6. Відображення списку тренувань
7. Відображення деталей тренування
8. Прокрутка екрану

Нефункціональні вимоги:

1. Кросплатформеність
2. Адаптивність екрану
3. Масштабованість
4. Додавання функціоналу без змін в архітектурі додатку
5. Швидкість реакції не більше 2 секунд

5

ЗАСОБИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

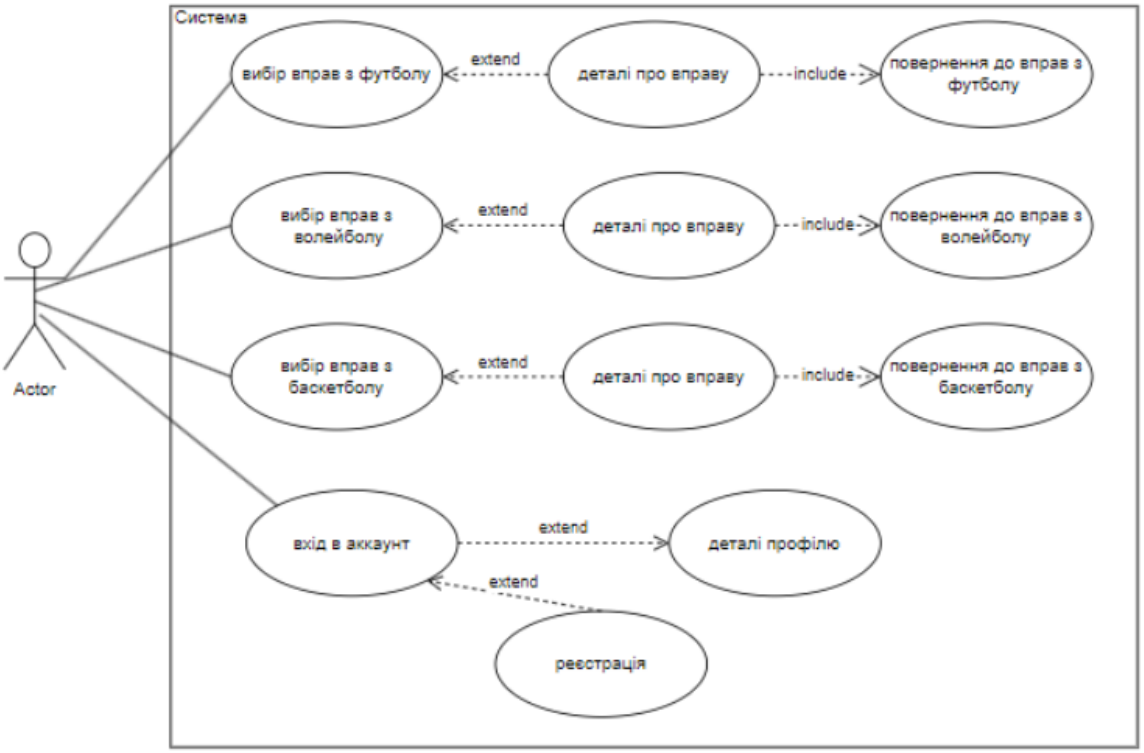
Figma

Kivy

PyCharm

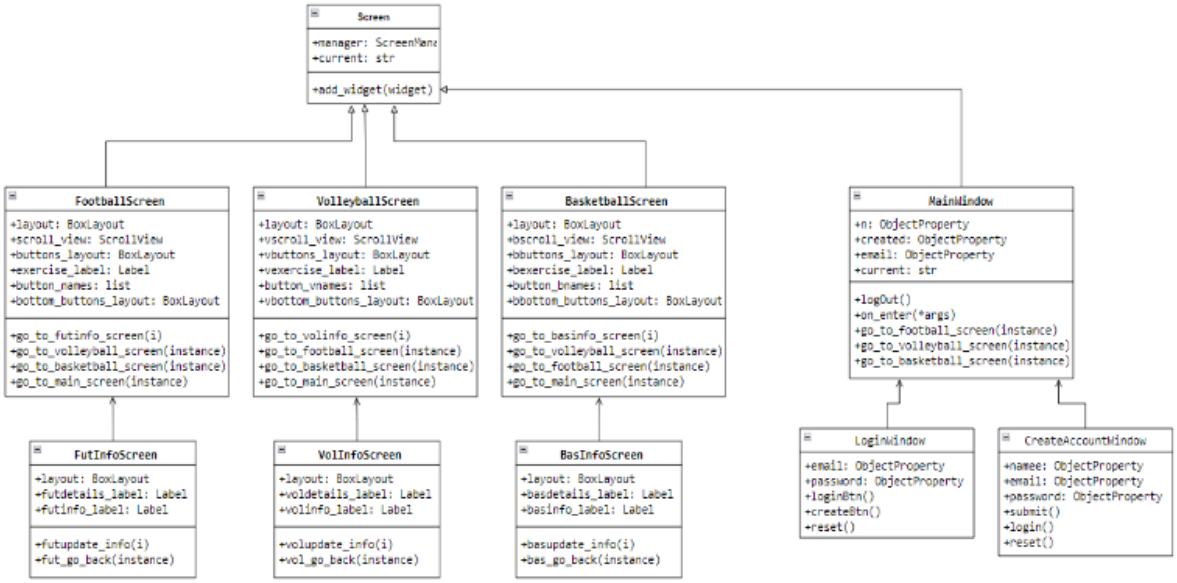
6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



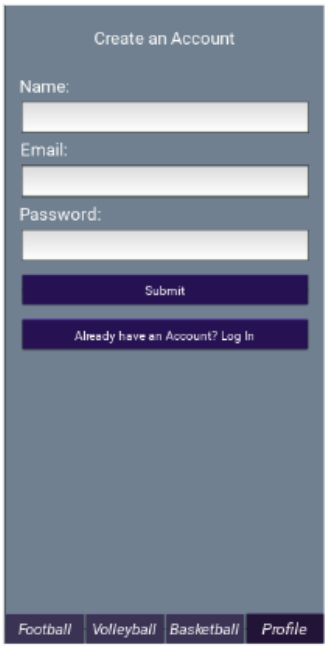
7

ДІАГРАМА КЛАСІВ

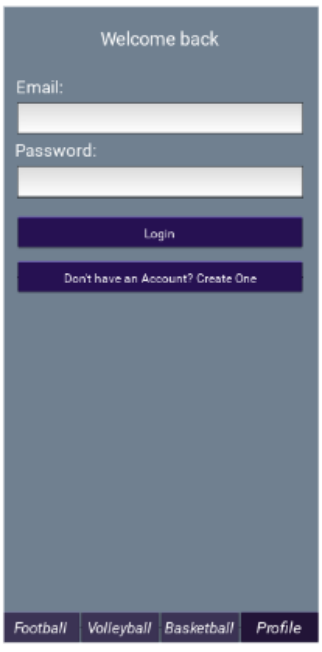


8

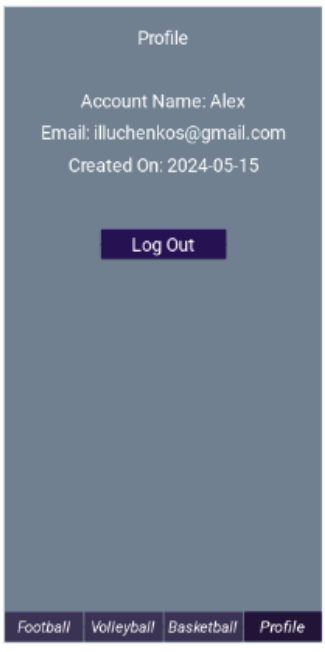
ЕКРАННІ ФОРМИ



Реєстрація



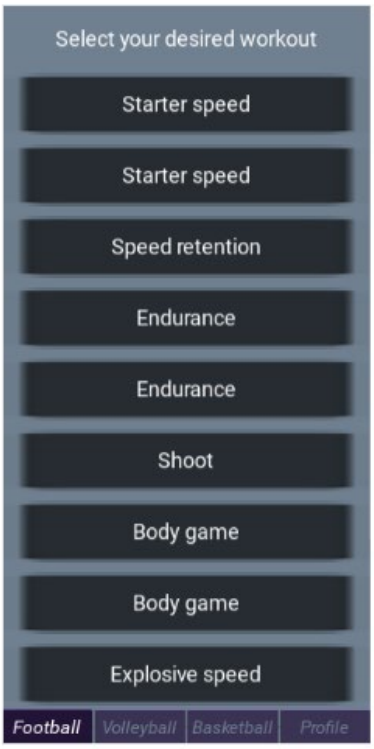
Авторизація



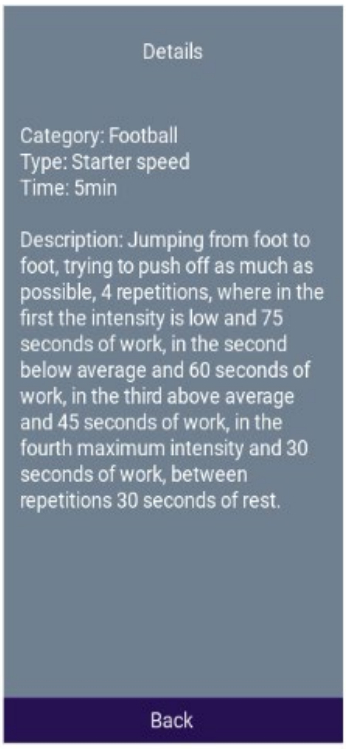
Деталі профілю

9

ЕКРАННІ ФОРМИ



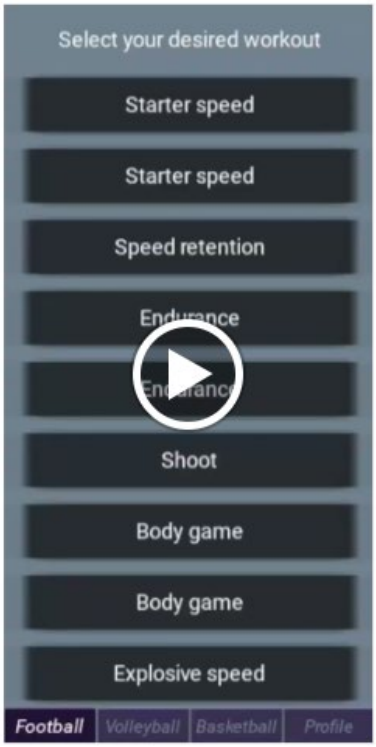
Вибір вправи



Деталі до вправи

10

ВІДЕО ДЕМОНСТРАЦІЯ РОБОТИ



11

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Іллюченко О.С. Порівняння Python фреймворків для розробки інтерфейсу мобільного додатку/Золотухіна О.А., Іллюченко О.С.// Матеріали всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в ІКТ». Збірник тез. 24.04.2024, м. Київ. К.:ДУІКТ - Подано до друку.

2. Іллюченко О.С. Розгляд можливостей застосування штучного інтелекту в мобільному додатку для тренувань фізичних якостей у різних видах спорту для аналізу запитів користувачів і надання рекомендацій./Золотухіна О.А., Іллюченко О.С.// Матеріали Всеукраїнської науково-практичної конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. 15.05.2024, м. Київ. К.:ДУІКТ - Подано до друку.

ВИСНОВКИ

1. Проведено аналіз обраної предметної області в сфері тренувань фізичних якостей. Визначено ключові потреби користувачів.

2. Проведено аналіз додатків-аналогів у сфері тренувань фізичних якостей, було визначено їх ключові функції, переваги та недоліки, які були використані при розробці додатку для тренувань.

3. Проведено аналіз засобів розробки та проектування програмного забезпечення. Було використано їх при розробці додатку для тренувань.

4. Визначено основні вимоги та проведено їх моделювання з використанням діаграми варіантів використання.

5. Розроблено додаток для тренувань фізичних якостей у різних видах спорту, який надає можливість користувачам переглядати та обирати вправи і тренуватись тренуватись завдяки опису до них.

6. Досліджено перспективи та можливості до удосконалення додатку з тренувань фізичних якостей, вказано варіанти до покращення в майбутньому.

7. Створені тестові сценарії, які охоплюють основні функціональні вимоги, і проведено перевірку коректної роботи програмного забезпечення.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

main.py

```
import kivy
from kivy.app import App
from kivy.uix.button import Button
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.label import Label
from kivy.uix.screenmanager import ScreenManager,
Screen
from kivy.core.window import Window
from kivy.uix.scrollview import ScrollView
from kivy.lang import Builder
from kivy.properties import ObjectProperty
from kivy.uix.popup import Popup
from database import DataBase

Window.clearcolor = (0.439, 0.502, 0.565, 1)

class FootballScreen(Screen):
   def __init__(self, **kwargs):
      super(FootballScreen, self).__init__(**kwargs)
      self.layout = BoxLayout(orientation='vertical')
      self.add_widget(self.layout)
      scroll_view = ScrollView(size_hint=(1, None),
size=(Window.width, Window.height))
      self.layout.add_widget(scroll_view)
      self.buttons_layout =
BoxLayout(orientation='vertical', size_hint_y=None,
spacing=10)
      self.buttons_layout.bind(minimum_height=self.bu
ttons_layout.setter('height'))

      exercise_label = Label(text="Select your desired
workout", size_hint_y=None, height=40)
      self.buttons_layout.add_widget(exercise_label)

      self.load_workouts()

      scroll_view.add_widget(self.buttons_layout)

      bottom_buttons_layout =
BoxLayout(orientation='horizontal', size_hint_y=None,
height=31)

      bottom_button_names = ["Football", "Volleyball",
"Basketball", "Profile"]

      for name in bottom_button_names:
         if name == "Football":
            bottom_button = Button(text=name,
size_hint=(None, None), size=(78.1, 31),
font_size=16,
                                   font_name='Roboto',
italic=True, background_color=(0.392, 0.231, 0.624,
1))
            bottom_button.bind(on_press=self.go_to_vol
leyball_screen)
         elif name == "Volleyball":
            bottom_button = Button(text=name,
size_hint=(None, None), size=(78.1, 31),
font_size=15,
                                   font_name='Roboto',
italic=True, background_color=(0.675, 0.58, 0.957, 1),
                                   color=(0.439, 0.502, 0.565,
1))
            bottom_button.bind(on_press=self.go_to_vol
leyball_screen)
         elif name == "Basketball":  # Додано умову
для кнопки "Basketball"
            bottom_button = Button(text=name,
size_hint=(None, None), size=(78.1, 31),
font_size=15,
                                   font_name='Roboto',
italic=True, background_color=(0.675, 0.58, 0.957, 1),
                                   color=(0.439, 0.502, 0.565,
1))
            bottom_button.bind(on_press=self.go_to_bas
ketball_screen)  # Доданий обробник події
         else:
            bottom_button = Button(text=name,
size_hint=(None, None), size=(78.1, 31),
font_size=15,
                                   font_name='Roboto',
italic=True, background_color=(0.675, 0.58, 0.957, 1),
                                   color=(0.439, 0.502, 0.565,
1))

         bottom_buttons_layout.add_widget(bottom_but
ton)

      self.layout.add_widget(bottom_buttons_layout)

   def load_workouts(self):
      db = DataBase()
      workouts = db.get_workouts("Football")
      for workout in workouts:
         workout_id, sport, type, time, description =
workout
         button = Button(text=type, size_hint=(None,
None), size=(312.9, 50), border=(-1, -1, -1, -1))
         button.background_color = (0.439, 0.502,
0.565, 1)
         button.bind(on_press=lambda instance,
i=workout_id: self.go_to_futinfo_screen(i))
         self.buttons_layout.add_widget(button)
      db.close()

   def go_to_futinfo_screen(self, workout_id):
      self.manager.current = 'futinfo'
      self.manager.get_screen('futinfo').futupdate_info(
workout_id)

   def go_to_volleyball_screen(self, instance):
```

```
        self.manager.current = 'volleyball'

    def go_to_basketball_screen(self, instance):
        self.manager.current = 'basketball'


class FutInfoScreen(Screen):
    def __init__(self, **kwargs):
        super(FutInfoScreen, self).__init__(**kwargs)
        self.layout = BoxLayout(orientation='vertical')
        self.add_widget(self.layout)

        self.futdetails_label = Label(text='Details',
size_hint_y=None, height=-190, halign='left')
        self.layout.add_widget(self.futdetails_label)
        self.futinfo_label = Label(size_hint_y=None,
height=640, valign='top', halign='left',
                                   padding=(60, 0), text_size=(400,
None))
        self.layout.add_widget(self.futinfo_label)
        button_layout =
BoxLayout(orientation='horizontal', size_hint_y=None,
height=40)
        back_button = Button(text='Back')
        back_button.background_color = (0.427, 0.192,
0.929, 1)
        back_button.corner_radius = (15)
        back_button.bind(on_press=self.fut_go_back)
        button_layout.add_widget(back_button)
        self.layout.add_widget(button_layout)

    def futupdate_info(self, workout_id):
        db = DataBase()
        workout = db.get_workout_by_id(workout_id)
        if workout:
            _, sport, type, time, description = workout
            futinfo = f"Category: {sport}\nType:
{type}\nTime: {time}\n\nDescription: {description}"
        else:
            futinfo = "Will be added soon"
        db.close()
        self.futinfo_label.text = futinfo

    def fut_go_back(self, instance):
        self.manager.current = 'football'


class VolleyballScreen(Screen):
    def __init__(self, **kwargs):
        super(VolleyballScreen, self).__init__(**kwargs)
        self.layout = BoxLayout(orientation='vertical')
        self.add_widget(self.layout)

        vscroll_view = ScrollView(size_hint=(1, None),
size=(Window.width, Window.height))
        self.layout.add_widget(vscroll_view)
        self.vbuttons_layout =
BoxLayout(orientation='vertical', size_hint_y=None,
spacing=10)
        self.vbuttons_layout.bind(minimum_height=self.v
buttons_layout.setter('height'))

        vexercise_label = Label(text="Select your desired
workout", size_hint_y=None, height=40)
        self.vbuttons_layout.add_widget(vexercise_label)

        self.load_workouts()

        vscroll_view.add_widget(self.vbuttons_layout)

        vbottom_buttons_layout =
BoxLayout(orientation='horizontal', size_hint_y=None,
height=31)

        vbottom_button_names = ["Football",
"Volleyball", "Basketball", "Profile"]

        for name in vbottom_button_names:
            if name == "Volleyball":
                vbottom_button = Button(text=name,
size_hint=(None, None), size=(78.1, 31),
font_size=15,
                                        font_name='Roboto',
italic=True, background_color=(0.392, 0.231, 0.624,
1))
            elif name == "Football":
                vbottom_button = Button(text=name,
size_hint=(None, None), size=(78.1, 31),
font_size=15,
                                        font_name='Roboto',
italic=True, background_color=(0.675, 0.58, 0.957, 1),
                                        color=(0.439, 0.502, 0.565,
1))
                vbottom_button.bind(on_press=self.go_to_fo
otball_screen)  # Доданий обробник події
            elif name == "Basketball":
                vbottom_button = Button(text=name,
size_hint=(None, None), size=(78.1, 31),
font_size=15,
                                        font_name='Roboto',
italic=True, background_color=(0.675, 0.58, 0.957, 1),
                                        color=(0.439, 0.502, 0.565,
1))
                vbottom_button.bind(on_press=self.go_to_ba
sketball_screen)  # Доданий обробник події
            else:
                vbottom_button = Button(text=name,
size_hint=(None, None), size=(78.1, 31),
font_size=15,
                                        font_name='Roboto',
italic=True, background_color=(0.675, 0.58, 0.957, 1),
                                        color=(0.439, 0.502, 0.565,
1))
                vbottom_button.bind(on_press=self.go_to_m
ain_screen)

            vbottom_buttons_layout.add_widget(vbottom_
button)

        self.layout.add_widget(vbottom_buttons_layout)

    def load_workouts(self):
        db = DataBase()
        workouts = db.get_workouts("Volleyball")
```

```
        for workout in workouts:
            workout_id, sport, type, time, description =
workout
            vbutton = Button(text=type, size_hint=(None,
None), size=(312.9, 50), border=(-1, -1, -1, -1))
            vbutton.background_color = (0.439, 0.502,
0.565, 1)
            vbutton.bind(on_press=lambda instance,
i=workout_id: self.go_to_volinfo_screen(i))
            self.vbuttons_layout.add_widget(vbutton)
        db.close()

    def go_to_volinfo_screen(self, workout_id):
        self.manager.current = 'volinfo'
        self.manager.get_screen('volinfo').volupdate_info(
workout_id)

    def go_to_football_screen(self, instance):
        self.manager.current = 'football'

    def go_to_basketball_screen(self, instance):
        self.manager.current = 'basketball'

    def go_to_main_screen(self, instance):
        self.manager.current = 'main'


class VolInfoScreen(Screen):
    def __init__(self, **kwargs):
        super(VolInfoScreen, self).__init__(**kwargs)
        self.layout = BoxLayout(orientation='vertical')
        self.add_widget(self.layout)
        self.voldetails_label = Label(text='Details',
size_hint_y=None, height=-190, halign='left')
        self.layout.add_widget(self.voldetails_label)
        self.volinfo_label = Label(size_hint_y=None,
height=640, valign='top', halign='left',
                                   padding=(60, 0), text_size=(400,
None))
        self.layout.add_widget(self.volinfo_label)

        button_layout =
BoxLayout(orientation='horizontal', size_hint_y=None,
height=40)
        back_button = Button(text='Back')
        back_button.background_color = (0.427, 0.192,
0.929, 1)
        back_button.corner_radius = (15)
        back_button.bind(on_press=self.vol_go_back)
        button_layout.add_widget(back_button)
        self.layout.add_widget(button_layout)

    def volupdate_info(self, workout_id):
        db = DataBase()
        workout = db.get_workout_by_id(workout_id)
        if workout:
            _, sport, type, time, description = workout
            volinfo = f"Category: {sport}\nType:
{type}\nTime: {time}\n\nDescription: {description}"
        else:
            volinfo = "Will be added soon"
        db.close()
        self.volinfo_label.text = volinfo

    def vol_go_back(self, instance):
        self.manager.current = 'volleyball'


class BasketballScreen(Screen):
    def __init__(self, **kwargs):
        super(BasketballScreen, self).__init__(**kwargs)
        self.layout = BoxLayout(orientation='vertical')
        self.add_widget(self.layout)

        bscroll_view = ScrollView(size_hint=(1, None),
size=(Window.width, Window.height))
        self.layout.add_widget(bscroll_view)

        self.bbuttons_layout =
BoxLayout(orientation='vertical', size_hint_y=None,
spacing=10)
        self.bbuttons_layout.bind(minimum_height=self.b
buttons_layout.setter('height'))

        bexercise_label = Label(text="Select your desired
workout", size_hint_y=None, height=40)
        self.bbuttons_layout.add_widget(bexercise_label)

        self.load_workouts()

        bscroll_view.add_widget(self.bbuttons_layout)

        bbottom_buttons_layout =
BoxLayout(orientation='horizontal', size_hint_y=None,
height=31)

        bbottom_button_names = ["Football",
"Volleyball", "Basketball", "Profile"]

        for name in bbottom_button_names:
            if name == "Basketball":
                bbottom_button = Button(text=name,
size_hint=(None, None), size=(78.1, 31),
font_size=15,
                                        font_name='Roboto',
italic=True, background_color=(0.392, 0.231, 0.624,
1))
            elif name == "Football":
                bbottom_button = Button(text=name,
size_hint=(None, None), size=(78.1, 31),
font_size=15,
                                        font_name='Roboto',
italic=True, background_color=(0.675, 0.58, 0.957, 1),
                                        color=(0.439, 0.502, 0.565,
1))
                bbottom_button.bind(on_press=self.go_to_fo
otball_screen)
            elif name == "Volleyball":
                bbottom_button = Button(text=name,
size_hint=(None, None), size=(78.1, 31),
font_size=15,
                                        font_name='Roboto',
italic=True, background_color=(0.675, 0.58, 0.957, 1),
```

```
                        color=(0.439, 0.502, 0.565,
1))
            bbottom_button.bind(on_press=self.go_to_v
olleyball_screen)
        else:
            bbottom_button = Button(text=name,
size_hint=(None, None), size=(78.1, 31),
font_size=15,
                        font_name='Roboto',
italic=True, background_color=(0.675, 0.58, 0.957, 1),
                        color=(0.439, 0.502, 0.565,
1))
            bbottom_button.bind(on_press=self.go_to_lo
gin_screen)

        bbottom_buttons_layout.add_widget(bbottom_
button)

    self.layout.add_widget(bbottom_buttons_layout)

  def load_workouts(self):
    db = DataBase()
    workouts = db.get_workouts("Basketball")
    for workout in workouts:
      workout_id, sport, type, time, description =
workout
      bbutton = Button(text=type, size_hint=(None,
None), size=(312.9, 50), border=(-1, -1, -1, -1))
      bbutton.background_color = (0.439, 0.502,
0.565, 1)
      bbutton.bind(on_press=lambda instance,
i=workout_id: self.go_to_basinfo_screen(i))
      self.bbuttons_layout.add_widget(bbutton)
    db.close()

  def go_to_basinfo_screen(self, workout_id):
    self.manager.current = 'basinfo'
    self.manager.get_screen('basinfo').basupdate_info
(workout_id)

  def go_to_football_screen(self, instance):
    self.manager.current = 'football'

  def go_to_volleyball_screen(self, instance):
    self.manager.current = 'volleyball'

  def go_to_basketball_screen(self, instance):
    self.manager.current = 'basketball'

  def go_to_login_screen(self, instance):
    self.manager.current = 'login'


class BasInfoScreen(Screen):
  def __init__(self, **kwargs):
    super(BasInfoScreen, self).__init__(**kwargs)
    self.layout = BoxLayout(orientation='vertical')
    self.add_widget(self.layout)

    self.basdetails_label = Label(text='Details',
size_hint_y=None, height=-190, halign='left')
    self.layout.add_widget(self.basdetails_label)

    self.basinfo_label = Label(size_hint_y=None,
height=640, valign='top', halign='left',
                padding=(60, 0), text_size=(400,
None))
    self.layout.add_widget(self.basinfo_label)

    button_layout =
BoxLayout(orientation='horizontal', size_hint_y=None,
height=40)
    back_button = Button(text='Back')
    back_button.background_color = (0.427, 0.192,
0.929, 1)
    back_button.corner_radius = (15)
    back_button.bind(on_press=self.bas_go_back)
    button_layout.add_widget(back_button)
    self.layout.add_widget(button_layout)

  def basupdate_info(self, workout_id):
    db = DataBase()
    workout = db.get_workout_by_id(workout_id)
    if workout:
      _, sport, type, time, description = workout
      basinfo = f"Category: {sport}\nType:
{type}\nTime: {time}\n\nDescription: {description}"
    else:
      basinfo = "Will be added soon"
    db.close()
    self.basinfo_label.text = basinfo

  def bas_go_back(self, instance):
    self.manager.current = 'basketball'


class DataBase:
  def __init__(self, database):
    self.connection = sqlite3.connect(database)
    self.cursor = self.connection.cursor()

  def add_user(self, name, email, password,
registration_date):
    self.cursor.execute("INSERT INTO profiles
(name, email, password, registration_date) VALUES
(?, ?, ?, ?)", (name, email, password,
registration_date))
    self.connection.commit()

  def validate(self, email, password):
    self.cursor.execute("SELECT * FROM profiles
WHERE email = ? AND password = ?", (email,
password))
    return self.cursor.fetchone() is not None

  def get_user(self, email):
    self.cursor.execute("SELECT * FROM profiles
WHERE email = ?", (email,))
    return self.cursor.fetchone()

class CreateAccountWindow(Screen):
  namee = ObjectProperty(None)
  email = ObjectProperty(None)
  password = ObjectProperty(None)
```

```
    def submit(self):
        if self.namee.text != "" and self.email.text != ""
and self.email.text.count("@") == 1 and
self.email.text.count(".") > 0:
            if self.password != "":
                db.add_user(self.namee.text, self.email.text,
self.password.text, "2024-05-30")  # Припустимо, що
дата реєстрації - це завжди поточна дата

                self.reset()

                self.manager.current = "login"
            else:
                invalidForm()
        else:
            invalidForm()

    def login(self):
        self.reset()
        self.manager.current = "login"

    def reset(self):
        self.email.text = ""
        self.password.text = ""
        self.namee.text = ""

class LoginWindow(Screen):
    email = ObjectProperty(None)
    password = ObjectProperty(None)

    def loginBtn(self):
        if db.validate(self.email.text, self.password.text):
            MainWindow.current = self.email.text
            self.reset()
            self.manager.current = "main"
        else:
            invalidLogin()

    def createBtn(self):
        self.reset()
        self.manager.current = "create"

    def reset(self):
        self.email.text = ""
        self.password.text = ""

class MainWindow(Screen):
    n = ObjectProperty(None)
    created = ObjectProperty(None)
    email = ObjectProperty(None)
    current = ""

    def logOut(self):
        self.manager.current = "login"

    def on_enter(self, *args):
        password, name, created =
db.get_user(self.current)
        self.n.text = "Account Name: " + name
        self.email.text = "Email: " + self.current
        self.created.text = "Registration Date: " + created

class WindowManager(ScreenManager):
    pass

def invalidLogin():
    pop = Popup(title='Invalid Login',
content=Label(text='Invalid username or password.'),
            size_hint=(None, None), size=(300, 150))
    pop.open()

def invalidForm():
    pop = Popup(title='Invalid Form', size_hint=(None,
None), size=(300, 150))
    content = Label(text='Please fill in all inputs with
valid information.')
    content.font_size = 12
    pop.content = content
    pop.open()

db = DataBase("profiles.db")
kv = Builder.load_file("profile.kv")

class MyApp(App):
    def build(self):
        screen_manager = ScreenManager()
        screen_manager.add_widget(FootballScreen(nam
e='football'))
        screen_manager.add_widget(FutInfoScreen(name
='futinfo'))
        screen_manager.add_widget(VolleyballScreen(na
me='volleyball'))
        screen_manager.add_widget(VolInfoScreen(name
='volinfo'))
        screen_manager.add_widget(BasketballScreen(na
me='basketball'))
        screen_manager.add_widget(BasInfoScreen(name
='basinfo'))
        screen_manager.add_widget(LoginWindow(name
='login'))
        screen_manager.add_widget(CreateAccountWind
ow(name='create'))
        screen_manager.add_widget(MainWindow(name
='main'))

        return screen_manager


if __name__ == '__main__':
    MyApp().run()

profile.kv
<CreateAccountWindow>:
    name: "create"
    namee: namee
    email: email
    password: passw

    FloatLayout:
        Label:
            text: "Create an Account"
            size_hint: 0.8, 0.2
            pos_hint: {"x":0.1, "top":1.05}
```

```
        font_size: (root.width**2 + root.height**2) /
13**4

    Label:
        pos_hint: {"x":0.1, "top":0.9}
        size_hint: 0.06, 0.05
        text: "Name: "
        font_size: (root.width**2 + root.height**2) /
13**4

    TextInput:
        pos_hint: {"x":0.05, "top":0.85}
        size_hint: 0.9, 0.05
        id: namee
        multiline: False
        font_size: (root.width**2 + root.height**2) /
13**4

    Label:
        pos_hint: {"x":0.1, "top":0.8}
        size_hint: 0.06, 0.05
        text: "Email: "
        font_size: (root.width**2 + root.height**2) /
13**4

    TextInput:
        pos_hint: {"x":0.05, "top":0.75}
        size_hint: 0.9, 0.05
        id: email
        multiline: False
        font_size: (root.width**2 + root.height**2) /
13**4

    Label:
        pos_hint: {"x":0.1, "top":0.7}
        size_hint: 0.16, 0.05
        text: "Password: "
        font_size: (root.width**2 + root.height**2) /
13**4

    TextInput:
        pos_hint: {"x":0.05, "top":0.65}
        size_hint: 0.9, 0.05
        id: passw
        multiline: False
        password: True
        font_size: (root.width**2 + root.height**2) /
13**4

    Button:
        pos_hint:{"x":0.05,"y":0.53}
        size_hint: 0.9, 0.05
        background_color: (0.427, 0.192, 0.929, 1)
        text: "Submit"
        font_size: (root.width**2 + root.height**2) /
14**4
        on_release:
            root.manager.transition.direction = "left"
            root.submit()

    Button:
        pos_hint:{"x":0.05,"y":0.46}
        size_hint: 0.9, 0.05
        background_color: (0.427, 0.192, 0.929, 1)
        font_size: (root.width**2 + root.height**2) /
14**4
        text: "Already have an Account? Log In"
        on_release:
            root.manager.transition.direction = "left"
            root.login()

<LoginWindow>:
    name: "login"
    email: email
    password: password

    FloatLayout:

        Label:
            text: "Welcome back"
            size_hint: 0.8, 0.2
            pos_hint: {"x":0.1, "top":1.05}

        Label:
            text:"Email: "
            font_size: (root.width**2 + root.height**2) /
13**4
            pos_hint: {"x":0.1, "top":0.9}
            size_hint: 0.06, 0.05

        TextInput:
            id: email
            font_size: (root.width**2 + root.height**2) /
13**4
            multiline: False
            pos_hint: {"x": 0.05 , "top":0.85}
            size_hint: 0.9, 0.05

        Label:
            text:"Password: "
            font_size: (root.width**2 + root.height**2) /
13**4
            pos_hint: {"x":0.1, "top":0.8}
            size_hint: 0.16, 0.05

        TextInput:
            id: password
            font_size: (root.width**2 + root.height**2) /
13**4
            multiline: False
            password: True
            pos_hint: {"x": 0.05, "top":0.75}
            size_hint: 0.9, 0.05

        Button:
            pos_hint:{"x":0.05,"y":0.62}
            size_hint: 0.9, 0.05
            background_color: (0.427, 0.192, 0.929, 1)
            font_size: (root.width**2 + root.height**2) /
14**4
            text: "Login"
            on_release:
                root.manager.transition.direction = "up"
                root.loginBtn()
```

```
        Button:
            pos_hint:{"x":0.05,"y":0.55}
            size_hint: 0.9, 0.05
            background_color: (0.427, 0.192, 0.929, 1)
            font_size: (root.width**2 + root.height**2) /
14**4
            text: "Don't have an Account? Create One"
            on_release:
                root.manager.transition.direction = "right"
                root.createBtn()

<MainWindow>:
    n: n
    email: email
    created:created

    FloatLayout:
        Label:
            text: "Profile"
            size_hint: 0.8, 0.2
            pos_hint: {"x":0.1, "top":1.05}

        Label:
            id: n
            text: "Account Name: "
            pos_hint: {"x":0.1, "top":0.95}
            size_hint: 0.8, 0.2

        Label:
            id: email
            text: "Email: "
            pos_hint:{"x": 0.1, "top":0.9}
            size_hint:0.8, 0.2

        Label:
            id: created
            text: "Registration Date: "
            pos_hint:{"x": 0.1, "top":0.85}
            size_hint:0.8, 0.2
            text: "Created: "

        Button:
            pos_hint:{"x":0.3, "y": 0.6}
            size_hint:0.4,0.05
            background_color: (0.427, 0.192, 0.929, 1)
            text: "Log Out"
            on_release:
                app.root.current = "login"
                root.manager.transition.direction = "down"

        Button:
            pos_hint:{"x":0,"y":0}
            size_hint: 0.25, 0.05
            background_color: (0.675, 0.58, 0.957, 1)
            font_size: 15
            font_name: 'Roboto'
            italic: True
            text: "Football"
            on_release:
                root.manager.transition.direction = "right"
                app.root.current = "football"

        Button:
            pos_hint:{"x":0.25,"y":0}
            size_hint: 0.25, 0.05
            background_color: (0.675, 0.58, 0.957, 1)
            font_size: 15
            font_name: 'Roboto'
            italic: True
            text: "Volleyball"

        Button:
            pos_hint:{"x":0.5,"y":0}
            size_hint: 0.25, 0.05
            background_color: (0.675, 0.58, 0.957, 1)
            font_size: 15
            font_name: 'Roboto'
            italic: True
            text: "Basketball"

        Button:
            pos_hint:{"x":0.75,"y":0}
            size_hint: 0.25, 0.05
            background_color: (0.392, 0.231, 0.624, 1)
            font_size: 16
            font_name: 'Roboto'
            italic: True
            text: "Profile"
```