

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Фітнес додаток на IOS з підтримкою Apple Watch»

Виконав студент групи ПЗ-20-2
ВОЛИНЕЦЬ Владислав Михайлович

Керівник роботи:
ГРИВЮК Олексій Петрович

Рецензент:
БАКАЛЮК Тетяна Анатоліївна

Житомир – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ВОЛИНЕЦЬ Владислав Михайлоич**

Керівник роботи: **ГРИВЮК Олексій Петрович**

Тема роботи: **«Фітнес додаток на IOS з підтримкою Apple Watch»**,
затверджена Наказом закладу вищої освіти від **«23» лютого 2024 р., №74/с**
Вихідні дані для роботи: **розробка мобільного фітнес додатку зосереджена на забезпеченні ефективної організації тренувань. Додаток має спрощувати керування своїми тренуваннями та вправами. Відслідковувати статистику.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Гривюк О.П	02.03.2024р.	02.03.2024р.
2	Гривюк О.П	02.03.2024р.	02.03.2024р.
3	Гривюк О.П	02.03.2024р.	02.03.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра включає мобільний фітнес додаток та пояснювальну записку. Пояснювальна записка складається з тексту обсягом 58 сторінок друкованого тексту, 24 сторінок додатків, списку використаних джерел на 25 найменувань, що займає 2 сторінки. Робота містить 15 рисунків і 5 таблиць, загальний обсяг роботи становить 82 сторіки.

Мета роботи – створення фітнес додатку на iOS полягає у наданні користувачам зручного і ефективного інструменту для планування тренувань, ведення здорового способу життя та досягнення їх фітнес-цілей.

У процесі роботи було проаналізовано аналогічні мобільні додатки, їх функціонал та недоліки.

Результатом стала розробка мобільного додатку, що покращує спосіб життя, надає змогу якісно керувати своїми тренуваннями та відображати статистику. Має зручний та мінімалістичний інтерфейс

КЛЮЧОВІ СЛОВА: IOS, ФІТНЕС, БАЗА ДАНИХ, КОРИСТУВАЧ, ТРЕНУВАННЯ, СТАТИСТИКА.

					ІПЗ.КР.Б-121-24-ПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата	Фітнес додаток на IOS з підтримкою Apple Watch	Літ.	Арк.	Аркушів
Розроб.		В. М. Волинець						
Керівник		О. П. Гривюк					3	81
Рецензент		Т. А. Вакалюк				Житомирська політехніка, група ІПЗ-20-2		
Зав. каф.		Т. А. Вакалюк						

ABSTRACT

The final bachelor's thesis includes a mobile fitness application and an explanatory note. The explanatory note consists of 58 pages of printed text, 24 pages of appendices, a list of 25 references, which occupies 2 pages. The work contains 18 figures and 5 tables, the total volume of the work is 82 pages.

The purpose of the work is to create a fitness application for iOS is to provide users with a convenient and effective tool for planning workouts, leading a healthy lifestyle and achieving their fitness goals.

In the process, we analyzed similar mobile applications, their functionality and shortcomings.

The result was the development of a mobile application that improves lifestyle, allows you to manage your workouts and display statistics. It has a user-friendly and minimalistic interface

KEYWORDS: IOS, FITNESS, DATABASE, USER, TRAINING, STATISTICS.

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У РЕАЛІЗАЦІЇ МОБІЛЬНОГО ФІТНЕС ДОДАТКУ	10
1.1 Постановка задачі	10
1.2 Аналіз аналогів програмного продукту.....	11
1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	18
Висновки до першого розділу	24
РОЗДІЛ 2. ПРОЄКТУВАННЯ МОБІЛЬНОГО ФІТНЕС ДОДАТКУ	25
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	25
2.2 Розробка бази даних системи	28
2.3 Проєктування та реалізація алгоритмів роботи системи.....	33
2.4 Реалізація функціоналу мобільного фітнес додатку	34
Висновки до другого розділу	39
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ФІТНЕС ДОДАТКОМ	41
3.1 Порядок встановлення та налаштування параметрів додатку	41
3.2 Структура інтерфейсу мобільного фітнес додатку	42
3.3 Тестування додатку	51
Висновки до третього розділу	54
ВИСНОВКИ.....	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	56
ДОДАТКИ	58

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс

AutoLayout – механізм для створення і регулювання інтерфейсу користувача на iOS

ARC – Automatic Reference Counting: Механізм для автоматичного керування пам'яттю в програмах для iOS

CRUD – основні операції з управління даними (Create, Read, Update, Delete)

JSON – JavaScript Object Notation

SDK – набір розробника для побудови додатків

БД – база даних

ПЗ – програмне забезпечення

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми розробки фітнес додатка на OS з підтримкою Apple Watch надзвичайно очевидна в сучасному цифровому віці. Швидкий розвиток технологій мобільних пристроїв і популярність гаджетів, які вимірюють показники здоров'я, таких як Apple Watch, створюють унікальні можливості для інтеграції інформаційних технологій з фітнесом і здоров'ям.

Розробка фітнес додатку на базі iOS з інтеграцією Apple Watch є важливим завданням, оскільки такий продукт може значно поліпшити якість життя користувачів, сприяти підвищенню їхньої фізичної активності та спортивних досягнень. Інтеграція з Apple Watch дозволяє реалізувати інноваційні функції, такі як автоматичне вимірювання пульсу, відстеження калорій спожитих, нагадування про тренування та багато іншого.

На сьогоднішній день спостерігається активний розвиток ринку мобільних додатків для здоров'я та фітнесу. Попит на такі рішення зростає, що свідчить про необхідність створення конкурентоспроможних інноваційних продуктів. Це відкриває широкі можливості для впровадження новаторських технологій та функцій у фітнес додатках, що інтегруються з Apple Watch.

Основною метою розробки фітнес додатка є задоволення потреби у зручному і ефективному способі контролю за фізичною активністю для широкого кола користувачів. Такий додаток може бути особливо корисним для людей, які ведуть активний спосіб життя, спортсменів, а також тих, хто прагне поліпшити своє здоров'я.

Метою цього дослідження є розробка фітнес додатка для платформи IOS з підтримкою Apple Watch.

Завдання, які потрібно виконати для успішної розробки такого додатка, включають:

- **Аналіз вимог:** Ретельно вивчити потреби цільової аудиторії і визначити функціональність, яку має включати додаток. Наприклад, це може

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

включати відстеження активності (пробіжки, прогулянки, велосипедні прогулянки), вправи, моніторинг пульсу, витрату калорій тощо.

- Проектування і архітектура: Розробити структуру додатка, включаючи взаємодію між iOS додатком і додатком для Apple Watch. Важливо визначити, які дані будуть синхронізовані між обома платформами та які функції будуть доступні на Apple Watch.
- Розробка iOS додатка: Виконати програмування інтерфейсу користувача та логіку додатка на платформі iOS, забезпечивши відповідність дизайну Apple і стандартам безпеки.
- Розробка додатка для Apple Watch: Створити додаток для Apple Watch, який буде взаємодіяти з основним додатком на iPhone. Це включає відстеження активності, відображення інформації про тренування, взаємодію з сенсорним екраном та датчиками Apple Watch.
- Тестування і відлагодження: Провести тестування обох додатків для забезпечення їх правильної роботи і відповідності функціональним вимогам. Особливу увагу слід приділити інтеграції між iOS і Apple Watch додатками.
- Впровадження і супровід: Після успішного тестування впровадити додаток в App Store для iOS і забезпечити регулярні оновлення і підтримку функцій. Важливо також розглянути можливості взаємодії з iCloud для зручності синхронізації даних між пристроями користувача.

Об'єктом дослідження для фітнес додатку під платформу iOS з підтримкою Apple Watch є всі аспекти, пов'язані з його розробкою, впровадженням і використанням. Це включає:

Дослідження потреб цільової аудиторії і визначення основних функцій додатку.

- Вивчення інтерфейсу користувача та зручності взаємодії з додатком на платформі iOS і Apple Watch.
- Аналіз вимог до безпеки і захисту даних користувачів.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

- Вивчення технічних можливостей синхронізації даних між iOS і Apple Watch.
- Визначення ефективності функціоналу відстеження фізичних активностей та здоров'я.
- Дослідження можливостей інтеграції з додатками третіх сторін або сервісами.

Предмет дослідження фітнес додатку для платформи iOS з підтримкою Apple Watch полягає в комплексному вивченні всіх аспектів, що стосуються його функціональності, взаємодії з користувачем, технічної реалізації і впливу на користувачів. Основні напрямки дослідження включають аналіз можливостей для відстеження фізичних активностей, моніторингу здоров'я, ефективності інтерфейсу користувача, безпеки даних, оптимізації ресурсів пристроїв і можливостей інтеграції з іншими сервісами та платформами.

В роботі було використано **методи** об'єктно орієнтованого проектування, мобільного програмування та дизайну.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У РЕАЛІЗАЦІЇ МОБІЛЬНОГО ФІТНЕС ДОДАТКУ

1.1 Постановка задачі

Мобільні додатки для фітнесу стають все більш популярними серед користувачів iOS, оскільки вони надають можливість відстежувати фізичну активність, керувати здоров'ям та покращувати спосіб життя. Цей проект має на меті розробити та впровадити унікальний фітнес додаток для платформи iOS. В рамках цієї задачі описані вимоги, архітектурні рішення та плани тестування для успішної реалізації додатку.

Додаток повинен забезпечувати:

- Створення користувачем тренувань з вправами.
- Відображення прогресу і досягнень користувача у вигляді графіків та статистики.
- Показувати історію тренувань, які користувач пройшов чи не закінчив.
- Фільтрувати вправи за групою м'язів та статусом завершення.
- Сповіщати користувача про те, що потрібно виконати тренування.

Основні етапи розробки додатку:

- Зробити аналіз конкурентів та виявити сильні та слабкі сторони.
- Проаналізувати потреби користувачів, основний функціонал.
- Розробка UI/UX дизайну з урахуванням сучасних трендів і вимог користувачів.
- Тестування на різних пристроях iOS для перевірки сумісності та оптимізації продуктивності.

Розробка бази даних:

- Створення бази даних з відповідними таблицями для якісного використання додатку.
- Реалізувати зв'язки між таблицями.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2 Аналіз аналогів програмного продукту

У сучасному світі мобільні додатки для фітнесу на платформі iOS займають важливе місце в підтримці активного способу життя та здорового харчування. Швидкий розвиток цього сегменту ринку призвів до появи великої кількості програмних продуктів, які пропонують різноманітні можливості для відстеження фізичної активності, управління дієтою та моніторингу здоров'я.

Аналіз аналогів програмного продукту є ключовим етапом в процесі розробки нового додатку. Він дозволяє отримати усвідомлення про поточний стан ринку, виявити сильні та слабкі сторони конкурентів, а також визначити можливості для удосконалення та відмінності, які можуть зробити новий продукт привабливішим для цільової аудиторії.

Це є важливим інструментом для успішної розробки програмного продукту, оскільки він дозволяє зрозуміти ринкові тенденції, пристосувати стратегію розвитку та забезпечити конкурентні переваги у сфері фітнесу та здоров'я.

Перший мобільний додаток, який варто розглянути, це "Nike Training Club" [1] (рис. 1.1). Цей додаток відомий світовим брендом Nike і є ідеальним інструментом для тих, хто цікавиться фітнесом та здоровим способом життя.

"Nike Training Club" розроблений з урахуванням потреб користувачів будь-якого рівня фізичної підготовки. Він пропонує величезний вибір тренувальних програм, які охоплюють різні аспекти фітнесу, включаючи кардіо, силові тренування, йогу та біг. Кожна програма доступна в різних варіаціях тривалості і складності, що дозволяє користувачам знаходити оптимальний варіант для своїх потреб.

Основні функції "Nike Training Club" включають можливість персоналізації тренувальних програм, адаптацію до індивідуальних фізичних можливостей та цілей. Крім того, додаток синхронізується з Apple Watch, що дозволяє відстежувати прогрес і результати тренувань прямо з вашого годинника.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

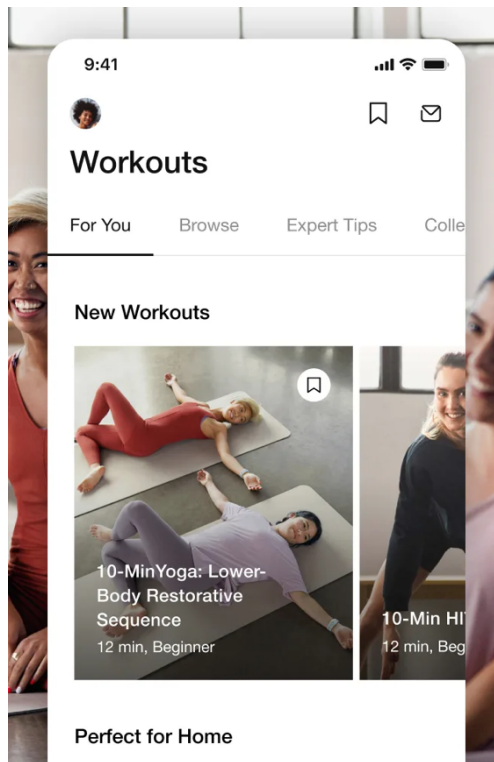


Рисунок 1.1 – Інтерфейс мобільного додаку Nike Training Club

Переваги Nike Training Club:

- Різноманітність тренувань і програм для всіх рівнів підготовки.
- Персоналізація тренувальних програм з урахуванням індивідуальних потреб.
- Синхронізація з Apple Watch для зручного відстеження прогресу.
- Мотиваційна система з віртуальними нагородами та викликами.

Недоліки Nike Training Club:

- Потреба у стабільному з'єднанні з Інтернетом для завантаження нових тренувальних програм та синхронізації даних.
- Деякі додаткові функції доступні лише у платній преміум-версії.

"Nike Training Club" є відмінним вибором для будь-якого, хто прагне займатися фізичною активністю і покращити своє здоров'я. Він поєднує в собі широкий вибір тренувань, високий рівень персоналізації і зручність використання завдяки синхронізації з Apple Watch. Незважаючи на деякі

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

недоліки, "Nike Training Club" заслуговує на увагу через свою ефективність і зручність для користувачів.

Наступний додаток, який було розглянуто – “Fitbit” [2] (рис1.2). "Fitbit" є одним з провідних мобільних додатків для відстеження фізичної активності, розробленим компанією Fitbit для користувачів платформи iOS. Цей додаток дозволяє вести облік активності, сну та харчування, спрямований на поліпшення загального здоров'я і фізичної форми.

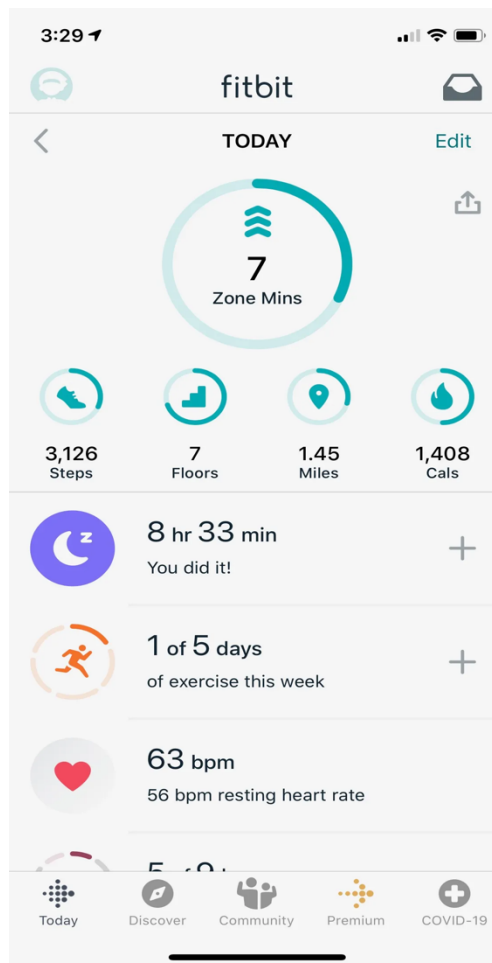


Рисунок 1.2 – Інтерфейс мобільного додаку Fitbit

Переваги:

- Відстеження фізичної активності: "Fitbit" відстежує кількість кроків, відстань, активність і споживані калорії, допомагаючи користувачам контролювати їхню фізичну активність щоденно.

- Моніторинг сну: Додаток аналізує якість сну на основі рухів під час сну, що дозволяє користувачам зрозуміти і покращити свій режим сну.
- Харчування і гідратація: "Fitbit" дозволяє вести облік прийому їжі та води, допомагаючи користувачам зберігати здоровий баланс харчування та гідратації.
- Мотиваційні виклики та досягнення: Додаток надає мотиваційні виклики, віртуальні нагороди і можливість конкурувати з друзями, що стимулює до активності та досягнення цілей.
- Інтеграція з іншими пристроями і додатками: "Fitbit" синхронізується з різними фітнес-трекерами, смарт-вагами та додатками для тренувань, що розширює його можливості та зручність.

Недоліки:

- Необхідність пристрою Fitbit: Для повного використання додатку "Fitbit" необхідно мати спеціальний фітнес-трекер від компанії Fitbit, що може бути обмеженням для деяких користувачів.
- Обмежена безкоштовна версія: Деякі розширені функції, такі як деталізований аналіз сну і звіти за харчування, доступні лише у платній преміум-версії додатку.
- Залежність від синхронізації: Для точного відстеження активності та інших параметрів потрібне постійне з'єднання з Інтернетом і синхронізація з пристроями.

Останній додаток який було розглянуто – це “MyFitnessPal” [3] (рис. 1.3)

Один з популярних додатків для фітнесу на iOS - "MyFitnessPal". Цей додаток розроблений компанією Under Armour і призначений для ведення здорового способу життя, зокрема контролю харчування і фізичної активності.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

"MyFitnessPal" дозволяє користувачам вести щоденник харчування, фіксуючи всі спожите їжу та напої. Додаток містить велику базу даних продуктів і страв, що дозволяє швидко знаходити і додавати необхідні елементи до щоденного журналу. Крім того, користувачі можуть вводити власні страви або використовувати функцію сканування штрих-кодів для швидкого внесення інформації про продукти.

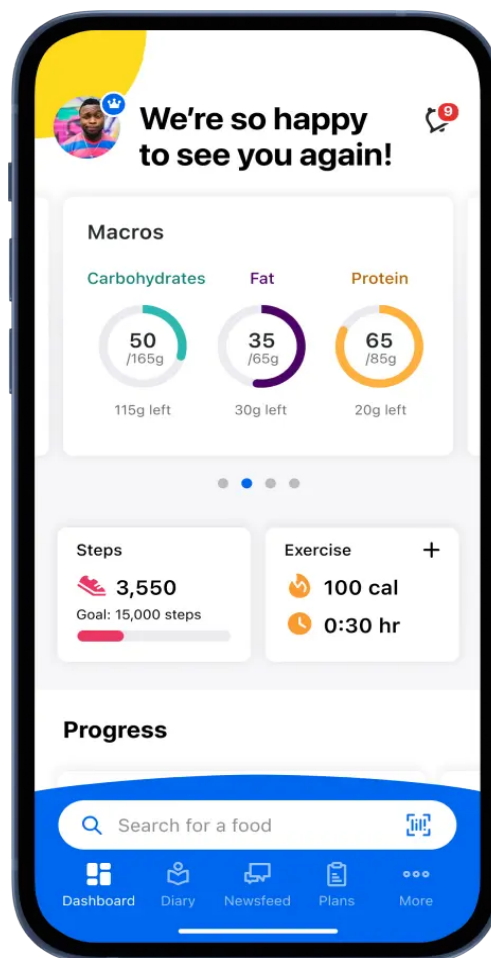


Рисунок 1.3 – Інтерфейс мобільного додаку MyFitnessPal

Переваги:

- Простота використання: Інтуїтивний і легкий у навчанні інтерфейс дозволяє швидко оволодіти всіма функціями додатку.
- Широкий функціонал: "MyFitnessPal" об'єднує у собі інструменти для ведення щоденника харчування і трекінгу фізичної активності,

що робить його універсальним інструментом для підтримки здорового способу життя.

- Велика база даних: Додаток має вражаючу базу даних продуктів і страв, що спрощує внесення і відстеження харчування.
- Спільнота користувачів: "MyFitnessPal" дозволяє користувачам спілкуватися і ділитися своїми досягненнями та цілями, що сприяє мотивації до активного способу життя.

Недоліки:

- Реклама і платні підписки: Деякі функції можуть бути обмежені або доступні лише з платним підпискою, що може бути недоступним для деяких користувачів.
- Не завжди точний трекінг калорій: Як і в більшості додатків, точність трекінгу калорій може бути вплинута на особисті характеристики користувача та умови використання додатку.

Усі вищеописані проєкти націлені покращення фізичних якостей людини та дисципліни. Нижче наведено порівняльну таблицю (табл. 1.1) цих трьох додатків

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

Порівняння мобільних фітнес додатків

Особливості / Додатки	Nike Training Club	Fitbit	MyFitnessPal
Тип додатку	Тренування	Фітнес-трекер, здоров'я і фітнес	Щоденник харчування, фізична активність
Функції	Різноманітні тренування Персоналізовані програми Відеоінструкції з тренерами	Трекінг активності, кроки, сон Моніторинг серцевого ритму Синхронізація з гаджетами Fitbit	Ведення треків харчування Трекінг калорій Ведення щоденника харчування
Спільнота користувачів	Маленька, активна спільнота	Велика спільнота користувачів	Активна спільнота користувачів
Точність трекінгу	Залежить від сенсорів та аналітики	Висока точність вимірювань	Точність може коливатись
Інтеграція з іншими додатками	Можливість інтеграції з деякими іншими додатками	Широка інтеграція з іншими трекерами	Інтеграція з іншими додатками харчування

У порівняльній таблиці були розглянуті три популярні мобільні додатки для фітнесу на платформі iOS: Nike Training Club, Fitbit і MyFitnessPal. Кожен з них має свої унікальні особливості і призначення:

- Nike Training Club - спеціалізується на тренуваннях, надаючи різноманітні програми і відеоінструкції, що робить його ідеальним вибором для тих, хто шукає персоналізовані тренування.
- Fitbit - орієнтований на фітнес-трекінг, здоров'я і фізичну активність, з високою точністю вимірювань та широким спектром функцій, таких як моніторинг серцевого ритму і синхронізація зі смарт-годинниками.
- MyFitnessPal - спрямований на ведення щоденника харчування та трекінг калорій, має велику базу даних продуктів і страв, що спрощує процес контролю харчування.

Кожен з цих додатків може відповідати різним потребам користувачів залежно від їхніх цілей і переваг. Вибір конкретного додатку залежатиме від того, що саме важливіше для конкретного користувача: інтенсивні тренування, стеження за здоров'ям чи контроль харчування.

1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Swift [4] - це сучасна мова програмування, розроблена компанією Apple для створення програмного забезпечення під iOS, macOS, watchOS та tvOS. Вона була представлена в 2014 році як заміна старій мові Objective-C, і вже за короткий час стала надійним і популярним інструментом для розробників.

Основні переваги Swift:

- Простота вивчення та читання коду: Swift пропонує простий і сучасний синтаксис, що робить його доступним для новачків у програмуванні. Код на Swift легше читати і зрозуміти, що сприяє швидшому розвитку проектів.

- **Безпека і надійність:** Swift включає в себе ряд заходів безпеки, які дозволяють виявляти та виправляти помилки програмування на ранніх етапах розробки. Це допомагає покращити якість програмного забезпечення та зменшити його схильність до помилок.
- **Швидкодія:** Swift розроблений з урахуванням оптимізації швидкості виконання. Він працює значно швидше, ніж Objective-C, завдяки вбудованим в мову функціям оптимізації пам'яті та швидкої компіляції.
- **Інтероперабельність з Objective-C:** Swift може взаємодіяти з кодом, написаним на Objective-C, що робить його ідеальним вибором для проектів, що потребують поступового переходу на нову мову без великих зусиль.
- **Активна спільнота та підтримка від Apple:** Swift підтримується і активно розвивається Apple, що забезпечує постійне оновлення і покращення мови разом з виходом нових версій операційних систем.
- **Мультиплатформеність:** Swift може бути використаний для розробки додатків не лише для iOS, але й для macOS, watchOS та tvOS, що спрощує роботу над багатоплатформовими проектами.

Swift є потужним інструментом для розробників, який поєднує в собі сучасний синтаксис, високу швидкодію та надійність, що робить його ідеальним вибором для створення широкого спектру додатків для платформ Apple.

SwiftUI [5] є відносно новою декларативною платформою для розробки інтерфейсів для додатків під iOS, macOS, watchOS та tvOS. Вперше представлена Apple в 2019 році, SwiftUI швидко став популярним інструментом серед розробників завдяки його модерній архітектурі та простоті використання.

Основні переваги SwiftUI:

- **Декларативний підхід:** SwiftUI пропонує декларативний синтаксис для опису інтерфейсу користувача. Розробники описують, як повинен виглядати інтерфейс у кожному стані програми, і SwiftUI

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

автоматично обчислює, як він повинен змінюватися відповідно до змін у даних або станах додатка.

- Швидкість розробки: Завдяки декларативному підходу і вбудованій підтримці live preview, SwiftUI дозволяє швидко створювати і візуалізувати інтерфейси без необхідності великої кількості коду.
- Інтерактивність та анімації: SwiftUI пропонує різноманітні вбудовані анімації та переходи між екранами, що спрощує створення привабливих інтерактивних додатків.
- Інтеграція з Swift і UIKit: SwiftUI може легко інтегруватися з існуючим кодом на Swift і UIKit. Це дає розробникам можливість поступово переходити до нової технології без необхідності переписувати весь код з нуля.
- Спрощена реактивність і стан: SwiftUI автоматично відслідковує зміни в стані додатка і оновлює відповідні інтерфейсні елементи, що спрощує управління даними і станом додатка.
- Підтримка всіх платформ Apple: SwiftUI підтримує розробку додатків для iOS, macOS, watchOS та tvOS, що робить його універсальним інструментом для розробки крос-платформових рішень.

SwiftUI є інноваційною платформою, яка значно спрощує розробку інтерфейсів для додатків на платформах Apple, забезпечуючи зручність, швидкість та можливість інтеграції з іншими технологіями. Використання SwiftUI дозволяє розробникам зосередитися на створенні користувацького досвіду, не заморочуючись технічними деталями, що робить його важливим інструментом для сучасних додатків на iOS та інших платформах Apple.

Combine [6] - це фреймворк, розроблений компанією Apple, який вперше був представлений на конференції WWDC у 2019 році. Він призначений для реалізації реактивного програмування в додатках для iOS, macOS, watchOS та

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

tvOS. Реактивне програмування дозволяє створювати програми, які реагують на зміни даних автоматично і миттєво.

Основні концепції Combine:

- Публікації та підписки (Publishers & Subscribers): Основна ідея Combine полягає в роботі з видачею (publishers), які генерують події, і підписниками (subscribers), які підписуються на ці події і реагують на них. Це спрощує управління потоком даних і сповіщеннями в додатках.
- Оператори та потоки даних: Combine надає багато вбудованих операторів, які дозволяють трансформувати, об'єднувати, фільтрувати та іншим чином обробляти дані в потоках. Це дозволяє зручно керувати потоками даних і реалізувати складні логічні взаємодії.
- Інтеграція з SwiftUI та UIKit: Combine ідеально інтегрується з SwiftUI і UIKit, що дозволяє створювати декларативні інтерфейси, які реагують на зміни в даних без необхідності вручну оновлювати інтерфейс.
- Асинхронність та керування пам'яттю: Combine вбудовує в себе механізми керування пам'яттю та асинхронні операції, що дозволяє уникнути підвісок та витоків пам'яті під час роботи з асинхронними задачами і потоками даних.
- Тестування і відладка: Combine надає інструменти для тестування коду, що використовує фреймворк, а також для відладки потоків даних, що дозволяє підтримувати і покращувати якість програмного забезпечення.

Combine відкриває нові можливості для розробників платформи Apple, дозволяючи створювати сучасні, ефективні і легко підтримувані додатки, які активно взаємодіють з користувачем і середовищем. Вбудована підтримка Combine в SwiftUI та UIKit робить його незамінним інструментом для

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

розробників, які працюють над проектами на платформі iOS та інших платформах Apple.

CoreData [7]- це фреймворк, розроблений компанією Apple, який надає механізм для управління структурованими даними в додатках для iOS, macOS, watchOS та tvOS. Він дозволяє розробникам зберігати, організовувати та маніпулювати даними в програмі, що включає у себе можливості для зберігання даних в SQLite, XML або бінарних файлах.

Основні концепції CoreData:

- **Модель даних (Data Model):** CoreData використовує модель даних, яка описує структуру та взаємозв'язки об'єктів, які зберігаються в базі даних. Модель створюється за допомогою візуального інструменту Xcode, що спрощує процес проектування даних.
- **Об'єктно-орієнтоване програмування:** CoreData надає об'єктно-орієнтований інтерфейс для роботи з даними, що дозволяє використовувати класи і об'єкти для представлення записів бази даних, що робить код більш структурованим і підтримуваним.
- **Управління контекстом (Managed Context):** CoreData використовує контексти для управління життєвим циклом даних. Контексти дозволяють вносити зміни до об'єктів, які пізніше можна зберегти в базі даних або відкатити зміни.
- **Постійне збереження даних:** CoreData автоматично керує збереженням і відновленням даних на диску. Вона використовує SQLite як основний механізм збереження даних, але також підтримує інші формати збереження.
- **Міграція даних:** CoreData підтримує механізми міграції даних, що дозволяють змінювати структуру бази даних без втрати існуючих даних користувача.
- **Інтеграція з іншими технологіями:** CoreData легко інтегрується з іншими технологіями та фреймворками Apple, такими як UIKit,

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

SwiftUI, і CloudKit, що робить його важливим інструментом для розробки додатків зі складною логікою даних.

CoreData є потужним інструментом для управління даними в додатках на платформі iOS, який надає об'єктно-орієнтований підхід до роботи з базами даних. Вона дозволяє розробникам ефективно організовувати і зберігати дані, що спрощує розробку складних додатків і забезпечує високу продуктивність програмного забезпечення.

Xcode [8] - це інтегроване середовище розробки (Integrated Development Environment, IDE), розроблене компанією Apple для створення додатків для iOS, macOS, watchOS та tvOS. Воно включає в себе набір інструментів і сервісів, які допомагають розробникам проектувати, програмувати, тестувати і впроваджувати свої додатки.

Основні функції Xcode:

- Інтерфейс користувача: Xcode має інтуїтивно зрозумілий інтерфейс з різними панелями та вікнами, що дозволяє зручно керувати всіма аспектами розробки додатка.
- Редактор коду: В Xcode вбудований потужний текстовий редактор з підсвічуванням синтаксису, автодоповненням коду, інтеграцією з системою контролю версій і підтримкою розширень для поліпшення продуктивності.
- Інтегроване налагоджування (Debugging): Xcode надає інструменти для налагодження додатків з можливістю стеження за змінними, точками зупинки, стеками викликів та іншими інструментами для виявлення помилок.
- Інструменти дизайну і інтерфейсу: Xcode має вбудовані інструменти для дизайну інтерфейсу користувача, такі як Interface Builder і SwiftUI Preview, що дозволяють візуально створювати і налаштовувати елементи інтерфейсу.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

- Інтеграція з іншими інструментами Apple: Xcode інтегрується з іншими ключовими технологіями Apple, такими як Instruments для профілювання додатків, TestFlight для тестування бета-версій, а також з Apple Developer Portal для управління сертифікатами, ідентифікаторами та профілями випуску.
- Підтримка мов програмування: Xcode підтримує різні мови програмування, такі як Swift, Objective-C, C++, а також Web-технології для розробки веб-прикладів для Safari.
- Інтеграція з Xcode Server: Для автоматизації та контролю версій у великих проектах Xcode підтримує інтеграцію з Xcode Server, що дозволяє автоматизувати збірку, тестування та розгортання додатків.

Xcode є незамінним інструментом для розробників, які створюють додатки для платформ Apple [9]. Він об'єднує всі необхідні інструменти та сервіси для ефективної розробки, тестування і впровадження додатків, що робить його основним вибором для програмістів, що працюють у середовищі iOS та macOS.

Висновки до першого розділу

Після проведення детального аналізу предметної області були визначені ключові вимоги для створення мобільного фітнес-додатку для iOS. Основні завдання включають створення тренувань, можливість додавання вправ та сетів, а також відслідковування статистики і історії тренувань. Під час дослідження були проаналізовані різноманітні аналогічні продукти з різним функціоналом і концепціями роботи.

На основі зібраних вимог було сформульовано необхідні технології для реалізації додатку, зокрема, мову програмування Swift, фреймворки SwiftUI для розробки інтерфейсу та CoreData [10] для управління базою даних. Цей підхід дозволяє забезпечити ефективну розробку, легку підтримку і високу продуктивність програмного забезпечення на платформі iOS.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ МОБІЛЬНОГО ФІТНЕС ДОДАТКУ

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Перед тим, як розпочати розробку мобільного додатку, вирішальне значення має чітке визначення його функціоналу. Це передбачає глибоке розуміння потреб користувачів та стратегічних цілей бізнесу.

Детальна проробка сценаріїв використання (або прецедентів) допоможе у цьому. Сценарії описують типові дії користувачів з додатком, ґрунтуючись на яких, можна чітко окреслити його функціональні вимоги.

Аналіз прецедентів використання дає низку переваг:

- Виявлення потреб користувачів: Чітке розуміння, що саме хочуть отримати користувачі від додатку.
- Визначення ключових функцій: Визначення, які функції є найважливішими та найнеобхіднішими для користувачів.
- Оптимізація бази даних: Створення бази даних, які ідеально відповідають потребам функціоналу додатку.

Створення бази даних:

Після розробки прецедентів можна перейти до створення бази даних та її налаштування. Це допоможе:

- Визначити, які дані необхідно зберігати: Чітко допоможе визначити, яка інформація необхідна для підтримки функціоналу додатку.
- Забезпечити ефективність: Створення бази даних, яка оптимізована для зберігання та доступу до інформації.

Переваги чітко визначеного функціоналу:

- Зменшення неузгодженостей. Мінімізування ймовірності виникнення проблем між функціоналом додатку та його базою даних.
- Забезпечення простої та ефективної взаємодії між клієнтом та додатком.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

Функціональні вимоги

Для мобільного додатку було визначено наступні функціональні вимоги:

- Відстуність автентифікації
- Створення тренування
- Створення вправ для тренувань
- Додавання сетів та кількість підходів до вправ
- Можливість перегляду історії тренувань
- Фільтрація тренувань
- Відображення статистики тренувань
- Відправлення локальних сповіщень

Також було визначено наступні нефункціональні вимоги:

- Продуктивність: Додаток повинен бути швидким і малоперервним у роботі навіть при інтенсивному використанні. Наприклад, час завантаження тренувань та відкликання статистики повинен бути мінімальним.
- Надійність: Додаток повинен працювати стабільно без частих збоїв або витоків пам'яті, особливо під час тривалого використання або під час переходу між різними екранами.
- Безпека: Забезпечення конфіденційності та цілісності даних користувачів є критично важливим. Додаток повинен використовувати сучасні методи шифрування та захисту даних, особливо коли йдеться про особисту інформацію користувачів.
- Сумісність: Додаток має бути сумісним із різними версіями iOS та різними моделями iPhone та iPad. Важливо перевіряти сумісність з новими випусками операційної системи та вчасно внести необхідні зміни.
- Інтерфейс користувача: Дизайн інтерфейсу повинен бути зрозумілим, ергономічним і зручним для користувачів. Він має

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

враховувати особливості платформи iOS та відповідати рекомендаціям Apple щодо дизайну.

- **Доступність:** Додаток має бути доступним для широкого кола користувачів, включаючи людей з обмеженими можливостями. Наприклад, повинна бути підтримка VoiceOver для незрячих користувачів.

Прецедент «Створення тренування»

Опис: Цей прецедент описує процес створення тренування

Основні кроки:

1. Зайти на вкладку “Тренування”
2. Натиснути кнопку +
3. Заповнити форму по створенні тренування
4. Зберегти форму
5. Користувач створив тренування і може продовжити додавання вправ до нього

Прецедент «Додавання вправи»

Опис: Цей прецедент описує процес додавання вправи до існуючого тренування

Основні кроки:

1. Натиснути на тренування, яке користувач створив
2. Натиснути кнопку +
3. Заповнити форму по додаванні вправи
4. Вправа з’явиться у переліку вправ для вказаного тренування

Прецедент «Додавання підходів »

Опис: Цей прецедент описує процес додавання підходів та ваги до вправи

Основні кроки:

1. Натиснути на вправу
2. Натиснути кнопку +
3. Заповнити форму по додаванні підходів та ваги
4. Підходи та вага до вправи з’явиться у переліку підходів

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

Прецедент «Фільтрація тренувань»

Опис: Цей прецедент описує процес, за допомогою якого користувач може фільтрувати свої вправи в історії вправ

Основні кроки:

1. Зайти на вкладку “Історія”
2. Натиснути кнопку “Фільтрація”
3. Обрати спосіб за яким потрібно відфільтрувати вправи
4. Покажуться усі вправи за вказаним фільтром

Прецедент «Статистика»

Опис: Цей прецедент описує процес, за допомогою якого користувач може переглянути свою статистику по вправам

Основні кроки:

1. Натиснути кнопку з іконкою графіка
2. Зайти на вкладку “Історія”
3. Відобразиться кругова діаграма
4. Покажеться статистика за весь час у вигляді кругової діаграми

2.2 Розробка бази даних системи

CoreData [11] є потужним інструментом для розробки та управління базами даних у додатках для платформ Apple, зокрема для iOS, macOS, watchOS та tvOS. Цей фреймворк надає зручні інструменти для роботи з об'єктно-орієнтованою моделлю даних і забезпечує ефективне зберігання, організацію та маніпулювання інформацією.

Основні концепції CoreData

- Модель даних (Data Model) [12]: Основною частиною розробки баз даних з використанням CoreData є створення моделі даних. Модель описує структуру даних, що буде зберігатися в базі. Вона включає сутності (Entities), атрибути (Attributes) та відносини (Relationships) між ними. Наприклад, якщо в додатку потрібно зберігати інформацію про користувачів та їх тренування, модель

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

може містити сутності "User" з атрибутами, такими як ім'я, вік та "Workout" з атрибутами, такими як тип тренування і тривалість.

- **NSManagedObject:** NSManagedObject [13] є базовим класом, який CoreData використовує для представлення об'єктів даних в вашій програмі. Він автоматично згенерований за моделлю даних і містить властивості та методи для доступу до атрибутів та відносин даних.
- **NSManagedObjectContext:** NSManagedObjectContext відповідає за управління об'єктами в базі даних. Він виконує роль зберігання, вилучення та зміни даних, що зберігаються. Кожен контекст має свою власну область дії, і ви можете створювати багато контекстів для різних потоків виконання вашого додатку.
- **NSPersistentStoreCoordinator:** NSPersistentStoreCoordinator відповідає за координацію збереження даних у вказаних місцях, таких як SQLite-файли [14] або в пам'яті. Він управляє підключеннями до різних сховищ даних та забезпечує цілісність даних між ними.

Процес розробки баз даних з CoreData

- **Створення моделі даних:** Перший крок у розробці бази даних з використанням CoreData - створення моделі даних. Це можна зробити за допомогою інструмента Xcode, що надає графічний інтерфейс для визначення сутностей, атрибутів і відносин між ними. Модель зберігається у файлі з розширенням .xcdatamodeld.
- **Генерація класів NSManagedObject [15]:** Після створення моделі CoreData автоматично генерує класи NSManagedObject для кожної сутності. Ці класи вже мають зазначені атрибути і відносини, що відображають структуру даних, визначену у моделі.

- Інтеграція з додатком: Після створення моделі даних і згенерованих класів NSManagedObject [16] необхідно інтегрувати CoreData в ваш додаток. Це включає ініціалізацію NSPersistentStoreCoordinator, створення NSManagedObjectContext та налаштування параметрів зберігання.
- Операції з даними: Після налаштування CoreData дозволяє вам здійснювати різноманітні операції з даними, такі як додавання нових записів, зчитування, оновлення та видалення існуючих. Ці операції виконуються через NSManagedObjectContext [17], що дозволяє управляти змінами та забезпечувати цілісність даних.

Переваги використання CoreData

- Інтеграція з Xcode [18]: CoreData інтегрується з Xcode, що спрощує розробку та налагодження додатків.
- Ефективне зберігання та доступ: Вона надає ефективний механізм зберігання даних з оптимальним доступом до них.
- Міграція даних: Підтримує механізм міграції даних для змін структури бази даних під час розвитку додатка.

CoreData є потужним інструментом для розробки баз даних у додатках для платформ Apple [19], забезпечуючи зручний інтерфейс для роботи з об'єктно-орієнтованою моделлю даних. Цей фреймворк дозволяє програмістам ефективно управляти даними, забезпечуючи високу продуктивність та надійність додатків.

Нижче приводиться опис усіх таблиць, які потрібні для реалізації додатку (табл. 2.1 – 2.4)

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.1

Опис таблиці Workout

Поле	Тип	Опис
id	UUID	Унікальний ідентифікатор
name	String	Назва тренування
notes	String	Додаткові нотатки
lastTrainedAt	Date	Дата останнього тренування
createdAt	Date	Дата створення тренування
bodyPart	String	Група м'язів
updatedAt	Date	Дата оновлення тренування
exercises	[Exercise]	Вправи
histories	[WorkoutHistory]	Історія

Таблиця «Workout». Відповідає за тренування користувача. Містить в собі вправи та історію тренувань.

Таблиця 2.2

Опис таблиці Exercise

Поле	Тип	Опис
id	UUID	Унікальний ідентифікатор
name	String	Назва вправи
notes	String	Додаткові нотатки
createdAt	Date	Дата створення вправи
updatedAt	Date	Дата оновлення вправи
exerciseSets	[ExerciseSet]	Підходи вправи
workout	Workout	Тренування

Таблиця «Exercise» пов'язана з таблицею Workout, так як у них зв'язок один до багатьох, тобто одне тренування має багато вправ.

Таблиця 2.3

Опис таблиці ExerciseSet

Поле	Тип	Опис
id	UUID	Унікальний ідентифікатор
name	String	Назва підходу
notes	String	Додаткові нотатки
createdAt	Date	Дата створення підходу
updatedAt	Date	Дата оновлення підходу
exercise	Exercise	Вправа
duration	Int	Тривалість
weight	Double	Вага

Таблиця «ExerciseSet» пов'язана з таблицею Exercise, так як у них зв'язок багато до одного, тобто багато підходів в одній вправі.

Таблиця 2.4

Опис таблиці WorkoutHistory

Поле	Тип	Опис
id	UUID	Унікальний ідентифікатор
status	Bool	Статус тренування
finishedAt	Date	Дата закінчення тренування
duration	Int	Тривалість
workout	Workout	Тренування

2.3 Проєктування та реалізація алгоритмів роботи системи

Коли користувач входить у додаток, він опиняється на екрані “Тренування”, де він зможе створити тренування, натиснувши кнопку “+” та відкривши форму додавання вправи, після чого він може натиснути на тільки що створене тренування та додати вправи для нього. Після створення вправи, користувач зможе додавати підходи та кількість повторень.

Після того, як користувач додав вправи та підходи до них, в нього буде можливість розпочати тренування натиснувши кнопку “Почати тренування”. На екрані самого тренування користувач зможе відстежувати вправи та відмічати їх виконаними. Крім того, від потреби зможе налаштувати таймер для виконання вправи чи паузи між вправами, чи підходами.

Користувач має можливість переглядати свою історію тренувань, перейшовши на екран “Історія”. На тому ж екрані в нього є можливість відфільтрувати вправи та подивитись свою загальну статистику по групам м’язів та загальний час виконання тренувань по кожній групі м’язів.

Для завершення, користувач може на екрані “Налаштування” увімкнути сповіщення та налаштувати час їх надходження. Також є можливість увімкнути автоматичний таймер для відпочинку після виконання підходу.

Це забезпечує користувачеві зручність у створенні, виконанні та відстеженні його тренувальних програм, а також аналізі результатів і підвищенні ефективності тренувань. Це сприяє підвищенню ефективності тренувань та стимулює до постійного покращення результатів.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 2.3 – Діаграма активності користувача

2.4 Реалізація функціоналу мобільного фітнес додатку

Для початку розглянемо реалізацію локальних сповіщень. Для цього створимо окремий файл “Network Helper” [20].

Створимо функцію, `registerForPushNotification`, яка виконує процес реєстрації додатка для отримання локальних пуш-сповіщень на пристрої.

```

static private func registerForPushNotification(type: LocalNotificationType) {
    let userNotification = UNUserNotificationCenter.current()
    userNotification.requestAuthorization(options: [.sound, .badge,
        .alert]) { (status, error) in
        if status && error == nil {
            addLocalNoification(type: type)
        }
    }
}

```

Наступним кроком буде реалізація налаштування локального сповіщення на певний час

```
static private func configureLocalNotification(at hour: Int?, minute: Int?) {
    invalidateLocalNotification(with: [kLocalNotificationIdentifier])
    let centre = UNUserNotificationCenter.current()
    let notificationContent = UNMutableNotificationContent()
    notificationContent.title = "Привіт, \(AppSettings.userName().isEmpty
? kCommonUserName : AppSettings.userName())!"
    notificationContent.body = "Прийшов час тренуватись 🏋️"
    notificationContent.sound = UNNotificationSound.default
    var dateComponent = DateComponents()
    dateComponent.hour = hour
    dateComponent.minute = minute
    let triggerAt = UNCalendarNotificationTrigger(dateMatching:
dateComponent, repeats: true)
    let request = UNNotificationRequest(identifier:
kLocalNotificationIdentifier, content: notificationContent, trigger: triggerAt)
    centre.add(request) { (error) in
        if error != nil {
            print(error ?? "")
        }
    }
}
```

Викликається функція `invalidateLocalNotification`, яка скасовує попереднє локальне сповіщення з вказаним ідентифікатором (`kLocalNotificationIdentifier`). Це може бути корисним для оновлення налаштувань сповіщень або скасування старого сповіщення перед налаштуванням нового.

Отримання поточного центру користувачьких сповіщень (`UNUserNotificationCenter.current()`). Створення вмісту сповіщення (`UNMutableNotificationContent`). Налаштування звуку сповіщення (`UNNotificationSound.default`).

Створюється об'єкт `dateComponent`, який містить годину і хвилину, коли має відбутись сповіщення.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

UNCalendarNotificationTrigger [21] - об'єкт тригера, що викликає сповіщення в задані моменти часу.

Створення запиту на сповіщення (UNNotificationRequest). Створюється об'єкт request для запиту на сповіщення з унікальним ідентифікатором (kLocalNotificationIdentifier), вмістом notificationContent та тригером triggerAt.

Цей процес важливий для додатків, що використовують сповіщення для підтримки користувачів у їхніх активаціях і нагадуваннях

Створення екрану налаштувань SettingsView, де можна вмикати сповіщення:

```
import SwiftUI
struct SettingsView: View {
    @EnvironmentObject var appSettings: AppSettings
    var body: some View {
        NavigationView {
            Form {
```

Створення секції для увімкнення локальних сповіщень для початку тренування на сьогоднішній день для користувача:

```
                Section {
                    ZStack {
                        RoundedRectangle(cornerRadius: 0)
                            .fill(Color.clear)
                        Toggle(isOn:
$appSettings.enabledReminder.animation()) {
                            Text("Нагадування про тренування")
                        }
                    }
                }
```

Показ додаткового поля для ручного налаштування часу сповіщення:

```
                if appSettings.enabledReminder {
                    DatePicker("Нагадування щодня о", selection:
$appSettings.notificationTime, displayedComponents: .hourAndMinute)
                        .foregroundColor(.secondary)
                        .accentColor(appSettings.themeColorView())
                }
            }
```

Створення секції для увімкнення автоматичного показу таймера після виконаного підходу:

```
Section(header: Text("Таймер відпочинку запуститься автоматично, щойно ви позначите підхід як завершений")) {  
    Toggle(isOn:  
$appSettings.shouldAutoStartRestTimer.animation()) {  
        Text("Таймер автоматичного відпочинку")  
    }  
}
```

Показ додаткового поля для ручного налаштування тривалості таймеру для відпочинку:

```
if appSettings.shouldAutoStartRestTimer {  
    Stepper(value: $appSettings.workoutTimerInterval) {  
        Text("Тривалість відпочинку")  
        .foregroundColor(.secondary)  
        Text("\$(Int(appSettings.workoutTimerInterval))  
Секунди.")  
        .modifier(CustomInfoTextStyle(appSettings:  
appSettings))  
    }  
}  
}  
}  
}.navigationBarTitle("Налаштування")  
}  
}
```

@EnvironmentObject var appSettings: AppSettings: Використовується для доступу до інстанції AppSettings [22], яка зберігає налаштування додатка і може бути змінена з будь-якої частини додатка. NavigationView [23]: Обгортає вміст екрану, дозволяючи навігацію між сторінками.

Функціонал створення тренування для користувача методом saveWorkout

```
func saveWorkout() {  
    if workoutToEdit != nil { // Update workout flow  
        workoutToEdit?.name = self.name  
        workoutToEdit?.notes = self.notes  
    }  
}
```

```

        workoutToEdit?.bodyPart
BodyParts.allCases[bodyPartIndex].rawValue
    } else { // New workout flow
        let newWorkout = Workout(context: managedObjectContext)
        newWorkout.name = self.name
        newWorkout.notes = self.notes
        newWorkout.bodyPart = BodyParts.allCases[bodyPartIndex].rawValue
        newWorkout.id = UUID()
        newWorkout.createdAt = Date()
        newWorkout.updatedAt = Date()
    }
    if managedObjectContext.hasChanges {
        do {
            try managedObjectContext.save()
            dismissView()
        } catch {
            print(error)
        }
    }
}

```

Якщо `workoutToEdit` не є `nil`, то це означає, що тренування редагується. В цьому випадку оновлюються властивості тренування (`name`, `notes`, `bodyPart`) з новими значеннями.

Якщо `workoutToEdit` є `nil`, створюється новий об'єкт `Workout` в контексті `managedObjectContext`. Додаються інформація про назву тренування (`name`), нотатки (`notes`), частину тіла (`bodyPart`) і унікальний ідентифікатор (`id`). Також встановлюються дата створення (`createdAt`) і дата оновлення (`updatedAt`), а також прапорець улюбленого тренування (`isFavourite`).

Після внесення змін у контекст `managedObjectContext` [24], перевіряється наявність змін (`hasChanges`). Якщо зміни є, вони зберігаються у базі даних `Core Data` за допомогою методу `save()`. Після успішного збереження викликається функція `dismissView()` для закриття поточного відображення.

Ця функція демонструє ефективний підхід до роботи з базою даних Core Data в SwiftUI [25], забезпечуючи збереження і оновлення даних тренування згідно з потребами додатка.

Висновки до другого розділу

Другий розділ проектування фітнес додатку був спрямований на розробку прецедентів використання та визначення функціональних вимог, які повинен виконувати додаток. Основні кроки включали аналіз сценаріїв взаємодії користувачів з додатком, встановлення вимог до функціоналу та ідентифікацію необхідного функціоналу для задоволення потреб користувачів і бізнес-цілей. В результаті цього етапу було сформульовано наступні кроки:

Прецеденти використання:

- Опрацьовані сценарії використання, які описують взаємодію користувачів з додатком, такі як створення і редагування тренувань, відстеження прогресу, налаштування нагадувань тощо.

Функціональні вимоги:

- Визначені основні функції, які повинен виконувати додаток, такі як управління тренувальними програмами, відстеження статистики тренувань, налаштування нагадувань та інші.

Нефункціональні вимоги:

- Встановлені нефункціональні вимоги, такі як швидкодія, надійність, безпека даних, зручність інтерфейсу користувача, підтримка різних версій операційних систем тощо.

Реалізація бази даних:

- За допомогою CoreData було створено ефективну базу даних з відповідними таблицями, що відображають структуру додатку. Це дозволило зберігати, оновлювати і видаляти дані тренувань, користувачів та іншу необхідну інформацію.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

В результаті цього етапу проектування було створено базу для подальшої реалізації функціоналу додатку, що відповідає визначеним вимогам і прецедентам використання. Цей підхід забезпечує високу ефективність, зручність у використанні та масштабованість додатку в майбутньому

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ФІТНЕС ДОДАТКОМ

3.1 Порядок встановлення та налаштування параметрів додатку

Діаграма розгортання (Deployment Diagram) в контексті iOS додатків є важливим інструментом для візуалізації та розуміння того, як різні компоненти додатку фізично розташовані і взаємодіють один з одним у середовищі розгортання. Особливо важливою є така діаграма для розробників, архітекторів та тих, хто відповідає за інфраструктуру, оскільки вона допомагає врахувати аспекти масштабування, надійності та ефективності додатку.

Структура діаграми розгортання iOS додатку

Вузли (Nodes):

- iOS пристрої: Вони представляють реальні пристрої, на яких встановлюється iOS додаток, такі як iPhone, iPad або iPod Touch.

Сервери:

- Фізичні або віртуальні сервери, які можуть використовуватися для зберігання даних, обробки бізнес-логіки або для надання додаткам додаткових функцій через мережу.

Артефакти (Artifacts):

- iOS додаток: Артефакт представляє сам додаток, який включає виконуваний код, ресурси (такі як зображення, звуки тощо) і всі необхідні файли.

Бібліотеки:

- Артефакти, які можуть бути використані додатком для доступу до певних функцій або служб, наприклад, сторонні бібліотеки для роботи з мережею, базами даних тощо.

Конфігураційні файли:

- Налаштування додатку, які можуть включати параметри підключення до серверів, API ключі, налаштування взаємодії з користувачем тощо.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

Клієнт-серверна взаємодія:

- iOS пристрої взаємодіють з серверами через мережеві протоколи, такі як HTTP, HTTPS, WebSocket тощо.

Взаємодія між серверами:

- Якщо додаток включає багато серверних компонентів, діаграма може показувати, як вони обмінюються даними між собою.

Захист і безпека:

- Діаграма може також відображати заходи безпеки, такі як використання SSL для захищеної передачі даних, автентифікацію користувачів і контроль доступу до різних ресурсів.

Масштабування і надійність:

- Важливо показати, як додаток може масштабуватися при збільшенні обсягів трафіку або даних, а також як забезпечується надійність (наприклад, резервне копіювання даних, відновлення після збоїв тощо).

3.2 Структура інтерфейсу мобільного фітнес додатку

При запуску додатку користувача вітає головний екран, з вибраною вкладкою “Тренування” на якому розміщені 3 нижні кнопки меню та показуються усі тренування користувача. Приклад наведено нижче (рис. 3.1)

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

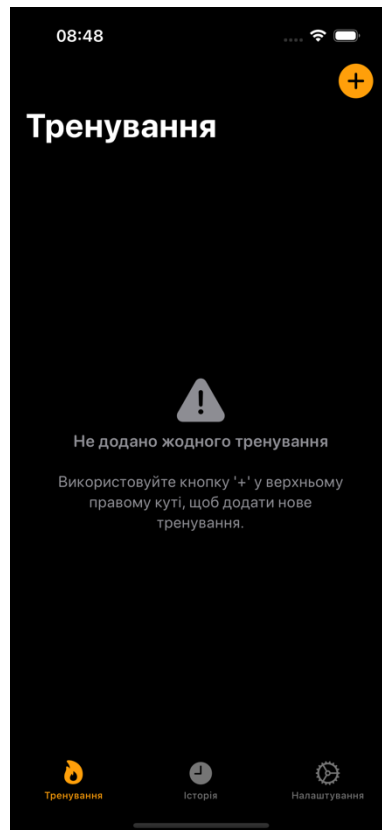


Рисунок 2.1 – Головний екран, екран з тренуваннями

На головному екрані, тобто екрані тренувань можна побачити іконку додавання тренування, у разі, якщо тренувань немає у користувача буде відображатись елемент дизайну, який показує, що список тренувань порожній та що потрібно зробити, щоб додати тренування

Користувач має змогу перейти на інші вкладки, або розпочати додавання тренувань до свого списку натиснувши кнопку “+”. Відкриється вікно форми, яку потрібно заповнити для створення тренування. Приклад нижче (рис. 3.2)

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

Рисунок 3.2 – Форма для створення тренування

Після того як користувач заповнив форму, обрав частину тіла на якою буде працювати, в користувача створюється вправа в списку усіх тренувань звідки користувач може перейти на тренування та додати вправи до нього. Приклад нижче списку тренувань (рис. 3.3)

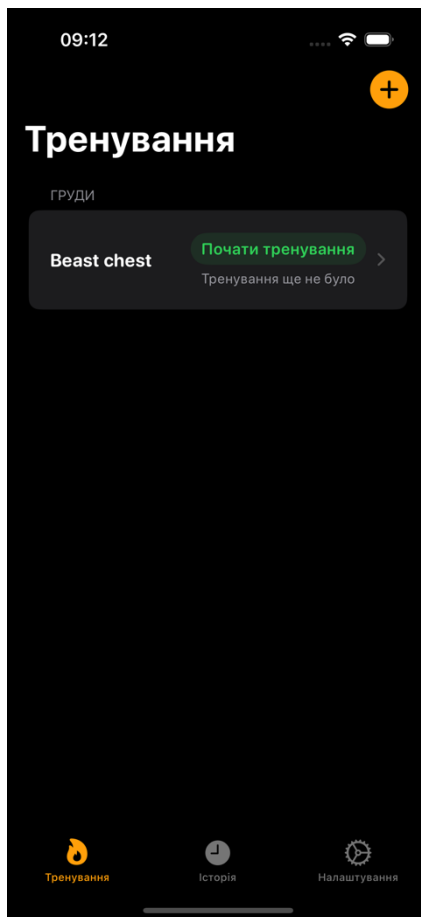


Рисунок 3.3 – Список усіх тренувань

Далі користувач може перейти на тренування та почати додавати вправи до нього через форму. Нижче наведено приклад (рис. 3.4)

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

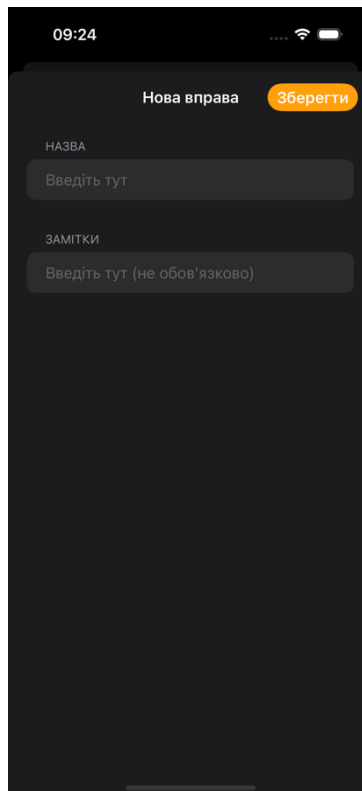


Рисунок 3.4 – Форма створення вправи

Після успішного додавання вправи, вправа з'явиться у переліку усіх вправ, де користувач зможе редагувати порядок вправ, або видалити вправу. Також є можливість перейти на вправу та додати до вправи підходи та кількість повторень з вказаною вагою. Нижче наведено приклад (рис. 3.5)

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

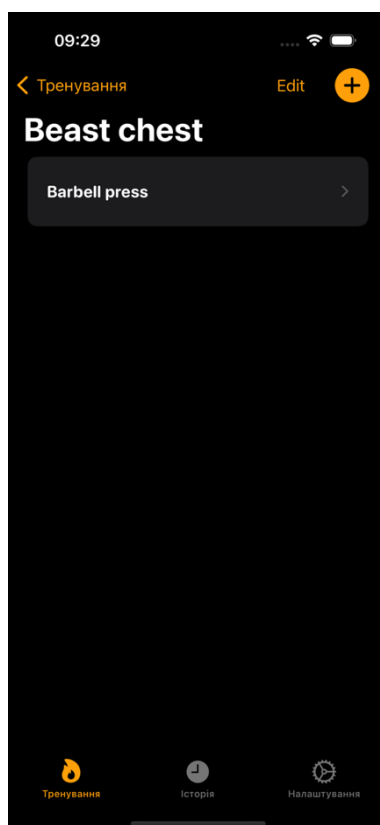


Рисунок 3.5– Список усіх вправ

Після того, як користувач перейшов на вправу, відкривається можливість додавання підходів та повторень через форму. Нижче наведено приклад (рис. 3.6)

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

09:49

Новий підхід **Зберегти**

НАЗВА
Введіть тут

ЗАМІТКИ
Введіть тут (не обов'язково)

ВАГА (7 КГ)
Минуле: 0 кг

ПОВТОРЕНЬ (10 РАЗ)
Минуле: 0 раз

Рисунок 3.6 – Додавання підходів через форму

Після того, як користувач заповнив форму – вправа з'явиться у переліку усіх вправ, де користувач зможе видалити будь-який підхід свайпом вбік, або оновити швидко дані підхода через кнопку “Оновити”. Приклад наведено нижче (рис. 3.7)

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

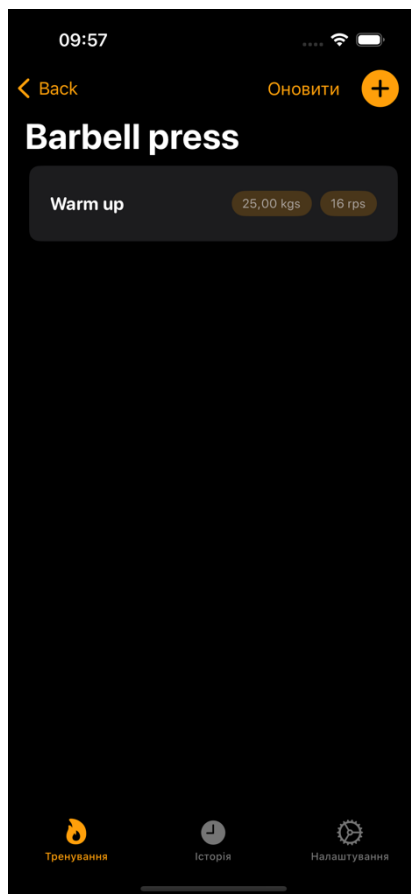


Рисунок 3.7 – Список вправ, які належать вказаній вправі

Після того, як користувач сторив тренування, додав вправи до нього там підходи, можна почати тренування та почнеться таймер, де користувач зможе помічати вправи, які він вже виконав і у разі виконання усіх вправ, користувач може завершити тренування. Приклад наведено нижче (рис. 3.8)

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

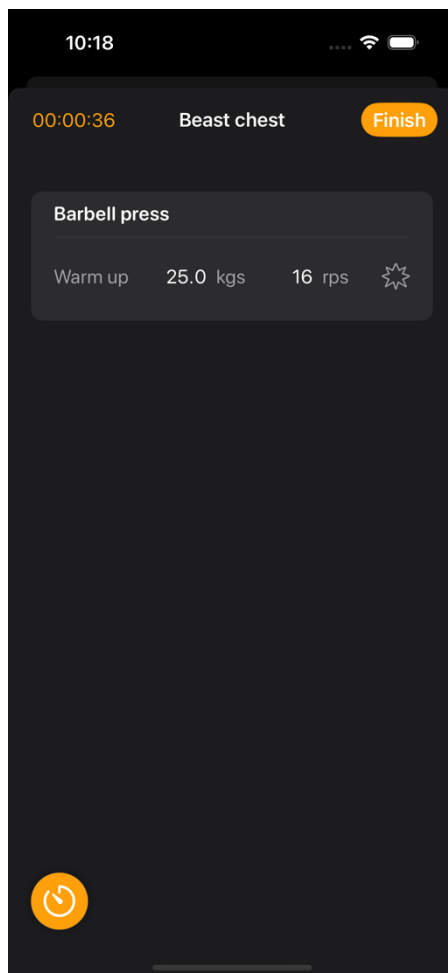


Рисунок 3.8 – Екран тренування

На екрані “Історія” користувач може подивитись свою історію вправ, або переглянути статистику виконаних тренувань. Приклад наведено нижче (рис. 3.9)

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

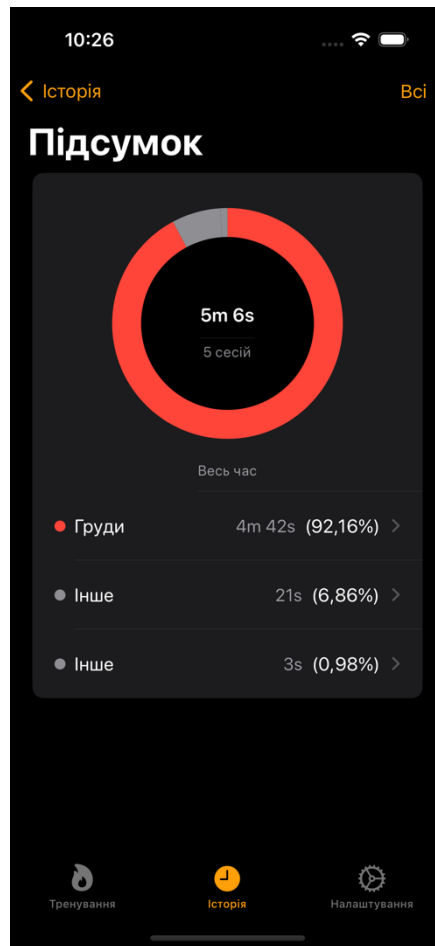


Рисунок 3.9 – Статистика історій тренувань

3.3 Тестування додатку

Для мобільного фітнес додатку на iOS тестування платформи має критичне значення з кількох ключових причин:

- **Забезпечення оптимальної продуктивності:** Мобільні фітнес додатки мають працювати швидко і ефективно, особливо на платформі iOS, яка відома своїм високим рівнем оптимізації. Тестування допомагає виявити й усунути будь-які проблеми з продуктивністю, такі як затримки чи зависання, що можуть виникнути через недоліки в програмному коді чи неефективне використання ресурсів.
- **Забезпечення стабільності і надійності:** Користувачі iOS очікують від додатків бездоганної роботи. Тестування дозволяє виявити й

усунути помилки, які можуть призвести до випадкових збоїв або витоків пам'яті. Це особливо важливо для фітнес додатків, де непередбачувані збої можуть призвести до втрати даних або негативного впливу на тренування користувача.

- **Забезпечення безпеки даних:** Мобільні фітнес додатки збирають особисті дані користувачів про їхні тренування і фізичну активність. Тестування допомагає виявити потенційні вразливості і забезпечити адекватний рівень захисту даних, щоб уникнути їхнього несанкціонованого доступу чи втрати.
- **Забезпечення зручного користувацького досвіду:** iOS відома своєю ергономічністю та користувацькою дружністю. Тестування дозволяє впевнитися, що фітнес додаток має інтуїтивно зрозуміле і привабливе інтерфейсне рішення, що підвищує ймовірність його прийняття та використання користувачами.
- **Забезпечення сумісності з різними пристроями і версіями iOS:** Різні моделі iPhone і iPad мають свої характеристики та версії операційної системи iOS. Тестування дозволяє переконатися, що додаток працює стабільно на різних пристроях і підтримується різними версіями iOS без проблем.

Я буду використовувати ручне тестування додатку, через те що це більш ефективніше та гнучкіше. Це дає змогу швидко у реальному часі перевірити додаток на некоректність роботи.

Для початку перевіримо коректність додавання тренування у список тренувань користувача. Якщо користувач не надав назву тренуванню, воно не має зберегтись у базі даних. Повинно з'явитись повідомлення про помилку. Нижче наведено приклад (рис. 3.10)

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

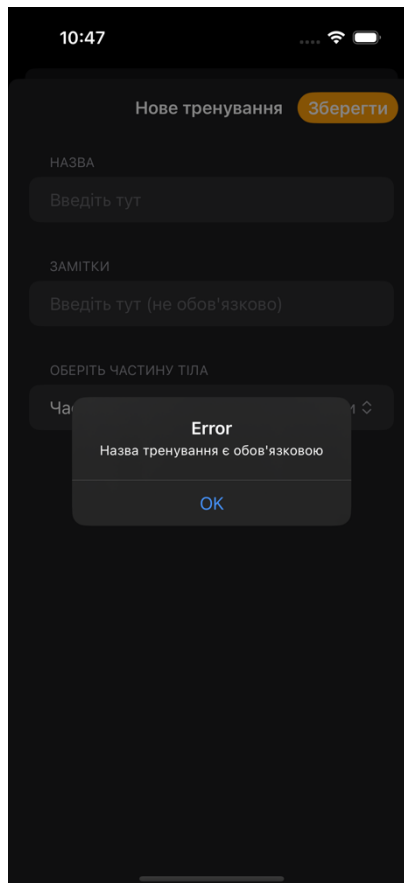


Рисунок 3.10 – Відображення помилки при створенні тренування

Наступне, що потрібно перевірити це відображення сповіщення про видалення вправи. Це допоможе уникнути непередбачуваних дій зі сторони користувача та запобігти випадковому видаленню тренування. Нижче наведено приклад (рис. 3.11)

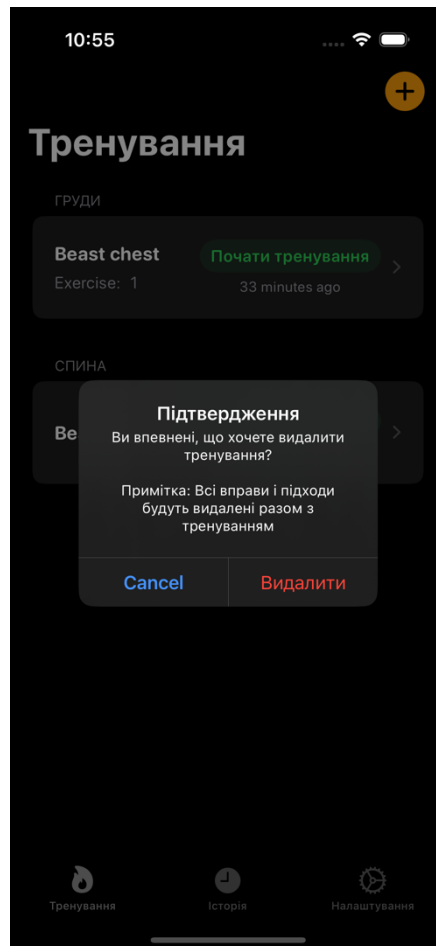


Рисунок 3.11 – Відображення вікна підтвердження при видаленні тренування

Висновки до третього розділу

У даному розділі було розглянуто інтерфейс та функціонал мобільного фітнес додатку. Також було розглянуто процес роботи з додатком, що користувач може робити у додатку та у якій послідовності. Додаток включає в себе додавання тренувань, вправ, підходів та кількість повторень. Також користувач має можливість переглядати історію своїх тренувань, статистику та фільтрувати тренування за групою м'язів та статусом тренування.

Було протестовано додаток на зручність та коректну роботу. Тестування допомогло підтвердити ефективність та надійність мобільного додатку.

ВИСНОВКИ

В процесі створення мобільного фітнес додатку було проведено глибокий аналіз інструментів управління проектами, вивчення вимог до апаратного та програмного забезпечення, необхідного для його реалізації. Цей аналіз відкрив важливі аспекти, які визначають успіх такого проекту. Наприклад, визначення потреб користувачів і уточнення функціональних вимог стали важливим етапом, що покладає основу для подальшої розробки.

Крім того, під час реалізації додатку велика увага була приділена його безпеці та зручності використання. Це включало в себе інтенсивне тестування, яке дозволило виявити і усунути потенційні проблеми ще до випуску продукту на ринок.

Отримані результати і технічний аналіз засвідчили ефективність і надійність додатку в реальних умовах використання. В результаті цього комплексного підходу було створено високоефективний і функціональний мобільний додаток для фітнесу. Він не лише задовольняє потреби користувачів у плануванні тренувань та підтримці здорового способу життя, але й надає надійність і зручність у використанні, що є критичними аспектами для успіху на конкурентному ринку мобільних додатків.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Nike Club Running [Електронний ресурс] – Режим доступу до ресурсу:
<https://apps.apple.com/ua/app/nike-run-club-running-coach/id387771637>
2. FitBit [Електронний ресурс] – Режим доступу до ресурсу:
<https://apps.apple.com/ua/app/fitbit-health-fitness/id462638897>
3. MyFitnessPal [Електронний ресурс] – Режим доступу до ресурсу:
<https://apps.apple.com/ua/app/myfitnesspal-calorie-counter/id341232718>
4. SwiftUI [Електронний ресурс] – Режим доступу до ресурсу:
<https://developer.apple.com/documentation/swiftui/>
5. Combine [Електронний ресурс] – Режим доступу до ресурсу:
<https://developer.apple.com/documentation/combine>
6. CoreData [Електронний ресурс] - Режим доступу до ресурсу:
<https://developer.apple.com/documentation/coredata/>
7. Xcode [Електронний ресурс] - Режим доступу до ресурсу:
<https://developer.apple.com/xcode/>
8. Combine - Combine documentation [Електронний ресурс] - Режим доступу до ресурсу:
<https://combine.readthedocs.io/en/master/>
9. HealthKit [Електронний ресурс] - Режим доступу до ресурсу:
<https://developer.apple.com/documentation/healthkit>
10. Swift [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.swift.org/documentation/>
11. Udemy [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.udemy.com>
12. SwiftUI Docs: Start Here [Електронний ресурс] - Режим доступу до ресурсу:
<https://swiftuidocs.com>
13. HealthKit Basics in Swift [Електронний ресурс] - Режим доступу до ресурсу:
<https://medium.com/ibtech/healthkit-basics-in-swift-30ea6ce27a78>
14. Stack Overflow [Електронний ресурс] - Режим доступу до ресурсу:
<https://stackoverflow.com>

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

15. iOS App Development for Beginners [Електронний ресурс] - Режим доступу до ресурсу: <https://www.coursera.org/learn/swift-programming>
16. Human Interface Guidelines [Електронний ресурс] - Режим доступу до ресурсу: <https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/>
17. 100 Days of Swift [Електронний ресурс] - Режим доступу до ресурсу: <https://www.hackingwithswift.com/100>
18. CodeSandBox [Електронний ресурс] - Режим доступу до ресурсу: <https://codesandbox.io/>
19. CodeAcademy [Електронний ресурс] - Режим доступу до ресурсу: <https://www.codecademy.com/>
20. UIKit [Електронний ресурс] - Режим доступу до ресурсу: <https://developer.apple.com/documentation/uikit>
21. Animations in Swift [Електронний ресурс] - Режим доступу до ресурсу: <https://medium.com/doyeona/animations-in-swift-f1ee069e21a7>
22. API Design Guidelines [Електронний ресурс] - Режим доступу до ресурсу: <https://www.swift.org/documentation/api-design-guidelines/>
23. Top 10 XCode ShortCuts For Beginners By Mohit Chug [Електронний ресурс] - Режим доступу до ресурсу: <https://medium.com/@hustlewithmohit/top-10-xcode-shortcuts-for-beginners-by-mohit-chug-73bb2414e5a6>
24. Top seven iOS app architecture patterns [Електронний ресурс] - Режим доступу до ресурсу: <https://medium.com/swift-concepts/ios-app-architecture-patterns-405931333ebe>
25. Protocol Oriented Programming in Swift [Електронний ресурс] - Режим доступу до ресурсу: <https://medium.com/swift-concepts/ios-app-architecture-patterns-405931333ebe>

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг Helper.swift

```

import UIKit
import AVFoundation
import CoreData

class Helper: NSObject {

    class func hapticFeedback(style: UIImpactFeedbackGenerator.FeedbackStyle =
.rigid) {
        if AppSettings.isHapticEnabled() {
            let generator = UIImpactFeedbackGenerator(style: style)
            generator.impactOccurred()
        }
    }

    class func informDuration(duration: Int16) {
        let minutes = Int16(duration / 60)
        let seconds = Int16(duration % 60)

        print("\(minutes) \ (seconds)")

        if seconds == 0 && minutes > 0 && minutes % 10 == 0 {
            let speechText = duration.speechDuration()
            speak(word: speechText)
        }
    }

    /**Speaks given string as audio. Not using this currently, need to use it
soon*/
    class func speak(word: String) {
        let utterance = AVSpeechUtterance(string: word)
        let synthesizer = AVSpeechSynthesizer()
        synthesizer.speak(utterance)
    }

    class func createDefaultWorkouts() {
        if let count = checkWorkoutsCount(), count > 0 {
            return
        }
        let appSettings = AppSettings()

        if !appSettings.addedDefaultWorkouts {
            let exampleWorkouts = ["Sample workout"]
            let exampleExercises = [["Sample exercise"]]

            let moc = kAppDelegate.persistentContainer.viewContext

```

```

        for (index, eWorkout) in exampleWorkouts.enumerated() {
            let workout = Workout(context: moc)
            workout.name = eWorkout
            workout.notes = "\"This is the sample workout to get the taste of
this tracker app. Delete this workout and configure your workouts and exercises to
get started.\""
            workout.bodyPart = BodyParts.arms.rawValue
            workout.id = UUID()
            workout.createdAt = Date()
            if index == 0 {
                workout.isFavourite = true
            }
            workout.updatedAt = Date()

            for (_, name) in exampleExercises[index].enumerated() {
                let exercise = Exercise(context: moc)
                exercise.name = name
                exercise.bodyPart = workout.bodyPart
                exercise.id = UUID()
                exercise.createdAt = Date()
                exercise.updatedAt = Date()

                for j in 1...3 {
                    let newExerciseSet = ExerciseSet(context: moc)
                    newExerciseSet.name = "Set \(j)"
                    newExerciseSet.id = UUID()
                    newExerciseSet.createdAt = Date()
                    newExerciseSet.updatedAt = Date()
                    newExerciseSet.weight = Double(5 * j)
                    newExerciseSet.reputation = 12

                    exercise.addToExerciseSets(newExerciseSet)
                }

                workout.addToExercises(exercise)
            }
            if moc.hasChanges {
                do {
                    try moc.save()
                    appSettings.addedDefaultWorkouts = true
                } catch {
                    print(error.localizedDescription)
                }
            }
        }
    }

    /**Check and sends count of workouts count*/
    class func checkWorkoutsCount() -> Int? {

```

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        let fetchRequest = NSFetchRequest<NSFetchRequestResult>(entityName:
Workout.entity().name ?? "Workout")
        do {
            let count = try kAppDelegate.persistentContainer.viewContext.count(for:
fetchRequest)
            return count
        } catch {
            return nil
        }
    }

    class func startDate(from daysAgo: Int) -> (startDate: Date?, endDate: Date?) {
        let now = Date()

        var startDate: Date?

        // Get end of the end date
        var endDate = Calendar.current.date(byAdding: .day, value: -1, to: now)
        endDate = Calendar.current.startOfDay(for: endDate!)

        var components = DateComponents()
        components.day = 1
        components.second = -1
        endDate = Calendar.current.date(byAdding: components, to: endDate!)

        if daysAgo > 0 && daysAgo != kTimePeriodAllOptionValue {
            startDate = Calendar.current.date(byAdding: .day, value: -daysAgo, to:
now)
            startDate = Calendar.current.startOfDay(for: startDate!) // Get start
of the start date.
        }

        return (startDate, endDate)
    }
}

```

Лістинг Extensions.swift

```

import SwiftUI

extension Int16 {

    func displayDuration() -> String {
        let duration = self
        return "\(String(format: "%02d", duration / kOneHour)):\(String(format:
"%02d", (duration % kOneHour) / kOneMinute)):\(String(format: "%02d", (duration %
kOneHour) % kOneMinute))"
    }
}

```

					ІПЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

func speechDuration() -> String {
    let duration = self
    let hours = duration / kOneHour
    let minutes = (duration % kOneHour) / kOneMinute
    var hourString = ""
    var minuteString = ""

    if hours > 0 {
        if hours == 1 {
            hourString.append("\(hours) hour")
        } else {
            hourString.append("\(hours) hours")
        }
    }
    if minutes > 0 {
        if minutes == 1 {
            minuteString.append("\(minutes) minute")
        } else {
            minuteString.append("\(minutes) minutes")
        }
    }
    let sample: [String] = [hourString, minuteString]
    return sample.join(separator: " ").trimmingCharacters(in:
.whitespacesAndNewlines)
}

func detailedDisplayDuration(shouldIncludeSeconds: Bool = true) -> String {
    let duration = self
    let hours = duration / kOneHour
    let minutes = (duration % kOneHour) / kOneMinute
    let seconds = (duration % kOneHour) % kOneMinute
    var hourString = ""
    var minuteString = ""
    var secondsString = ""

    if hours > 0 {
        hourString.append("\(hours)h")
    }
    if minutes > 0 {
        minuteString.append("\(minutes)m")
    }
    if seconds > 0 {
        secondsString.append("\(seconds)s")
    }
    let sample: [String] = [hourString, minuteString, secondsString]
    return sample.join(separator: " ").trimmingCharacters(in:
.whitespacesAndNewlines)
}
}

```

```

extension DateFormatter {

    var appDormatter: DateFormatter {
        let df = DateFormatter()
        df.dateStyle = .medium
        return df
    }

}

extension Double {

    func detailedDisplayDuration(shouldIncludeSeconds: Bool = true) -> String {
        let duration = self
        let kOneHour: Double = 3600.0
        let kOneMinute: Double = 60.0

        let hours = Int(duration / kOneHour)
        let minutes = Int((duration.truncatingRemainder(dividingBy: kOneHour)) /
kOneMinute)
        let seconds = Int((duration.truncatingRemainder(dividingBy:
kOneHour)).truncatingRemainder(dividingBy: kOneMinute))
        var hourString = ""
        var minuteString = ""
        var secondsString = ""

        if hours > 0 {
            hourString.append("\(hours)h")
        }
        if minutes > 0 {
            minuteString.append("\(minutes)m")
        }
        if seconds > 0 {
            secondsString.append("\(seconds)s")
        }
        let sample: [String] = [hourString, minuteString, secondsString]
        return sample.joined(separator: " ").trimmingCharacters(in:
.whitespacesAndNewlines)
    }

}

extension Notification.Name {

    static let didSaveTodayWorkoutSet = Notification.Name("didSaveTodayWorkoutSet")

}

```

Лістинг NotificationHelper.swift

```
import UIKit

private let kLocalNotificationIdentifier = "kLocalNotificationIdentifier"
private let kLocalTimerNotificationIdentifier = "kLocalTimerNotificationIdentifier"

enum LocalNotificationType {
    case date(Date)
    case interval(TimeInterval)
}

class NotificationHelper: NSObject {

    static private func configureLocalNotification(at hour: Int?, minute: Int?) {
        invalidateLocalNotification(with: [kLocalNotificationIdentifier])

        let centre = UNUserNotificationCenter.current()

        let notificationContent = UNMutableNotificationContent()
        notificationContent.title = "Привіт, \(AppSettings.userName().isEmpty ?
kCommonUserName : AppSettings.userName())!"
        notificationContent.body = "Прийшов час тренуватись 🏋️"

        notificationContent.sound = UNNotificationSound.default

        var dateComponent = DateComponents()
        dateComponent.hour = hour
        dateComponent.minute = minute

        let triggerAt = UNCalendarNotificationTrigger(dateMatching: dateComponent,
repeats: true)

        let request = UNNotificationRequest(identifier:
kLocalNotificationIdentifier, content: notificationContent, trigger: triggerAt)
        centre.add(request) { (error) in
            if error != nil {
                print(error ?? "")
            }
        }
    }

    /**Invalidate local notification which we configured in
`configureLocalNotification()`*/
    static private func invalidateLocalNotification(with identifiers: [String]) {

        UNUserNotificationCenter.current().removeDeliveredNotifications(withIdentifiers:
identifiers)
```

					ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

UNUserNotificationCenter.current().removePendingNotificationRequests(withIdentifier
s: identifiers)
}

static func resetNotifications() {
    invalidateLocalNotification(with: [kLocalNotificationIdentifier])
}

static func addLocalNoifcation(type: LocalNotificationType) {
    let userNotification = UNUserNotificationCenter.current()
    userNotification.getNotificationSettings { (setting) in
        if setting.authorizationStatus == .authorized {
            switch type {
            case .date(let date):
                let hour = Calendar.current.component(.hour, from: date)
                let minute = Calendar.current.component(.minute, from: date)
                configureLocalNotification(at: hour, minute: minute)
            case .interval(let interval):
                scheduleNotification(with: interval)
            }
        } else {
            registerForPushNotification(type: type)
        }
    }
}

static private func registerForPushNotification(type: LocalNotificationType) {
    let userNotification = UNUserNotificationCenter.current()
    userNotification.requestAuthorization(options: [.sound, .badge, .alert]) {
(status, error) in
        if status && error == nil {
            addLocalNoifcation(type: type)
        }
    }
}

static func scheduleNotification(with interval: TimeInterval) {
    print("🔴🌟 \(interval)")
    invalidateLocalNotification(with: [kLocalTimerNotificationIdentifier])

    let centre = UNUserNotificationCenter.current()

    let notificationContent = UNMutableNotificationContent()
    notificationContent.title = "Час закінчився!"

    notificationContent.sound = UNNotificationSound.default

```

```

        let triggerAt = UNTimeIntervalNotificationTrigger(timeInterval: interval,
repeats: false)
        let request = UNNotificationRequest(identifier:
kLocalTimerNotificationIdentifier, content: notificationContent, trigger:
triggerAt)
        centre.add(request) { (error) in
            print("🔴🌟 \(error)")
            if error != nil {
                print(error ?? "")
            }
        }
    }

    static func resetTimerNotification() {
        invalidateLocalNotification(with: [kLocalTimerNotificationIdentifier])
    }
}

```

Лістинг SettingsView.swift

```

import SwiftUI

struct SettingsView: View {

    @EnvironmentObject var appSettings: AppSettings

    var body: some View {
        NavigationView {
            Form {
                Section {
                    ZStack {
                        RoundedRectangle(cornerRadius: 0)
                            .fill(Color.clear)
                        Toggle(isOn: $appSettings.enabledReminder.animation())
                    }
                    Text("Нагадування про тренування")
                }
                if appSettings.enabledReminder {
                    DatePicker("Нагадування щодня о", selection:
$appSettings.notificationTime, displayedComponents: .hourAndMinute)
                        .foregroundColor(.secondary)
                        .accentColor(appSettings.themeColorView())
                }
            }

            Section(footer: Text("Таймер відпочинку запуститься автоматично,
щойно ви позначите підхід як завершений")) {
                Toggle(isOn: $appSettings.shouldAutoStartRestTimer.animation())
            }
            Text("Таймер автоматичного відпочинку")
            if appSettings.shouldAutoStartRestTimer {

```

Лістинг SummaryView.swift

					ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        var timePeriodString: String {
            let dates = Helper.startDate(from:
appSettings.historySelectedTimePeriod.rawValue)

            if let startDate = dates.startDate, let endDate = dates.endDate {
                let formatter = DateFormatter()
                formatter.dateStyle = .medium
                return "\(formatter.string(from: startDate)) to
\(formatter.string(from: endDate))"
            }
            return "Весь час"
        }

        var body: some View {
            ZStack {
                if progress.count == 0 {
                    EmptyStateInfoView(title: "Підсумок тренування не доступний.
Почніть тренування.")
                } else {
                    List {
                        VStack(spacing: 0) {
                            PieChart(progress: self.progress, segments: self.segments)
                            Text(timePeriodString)
                                .font(kPrimaryFootnoteFont)
                                .foregroundColor(.secondary)
                                .frame(alignment: .center)
                        }
                        ForEach(0..

```

```

        }
        .padding([.top, .bottom], 10)
    }
}
}
.listStyle(InsetGroupedListStyle())
}
.navigationBarItems(trailing:
    Menu(content: {
        Section {
            ForEach(0..
```

```

        WorkoutHistory.fetchSummary(startedAgo:
appSettings.historySelectedTimePeriod.rawValue, context: self.managedObjectContext)
{ (data) in
    self.progress = data
    self.chartData()
}
}
.navigationBarTitle(Text("Підсумок"))
}

func percentage(of duration: Double) -> Double {
    (duration / total) * 100.0
}

func chartData() {
    var lastEndAngle = 0.0
    segments = []

    for pro in self.progress {
        let percent = percentage(of: pro.0)
        let angle = (360 / 100) * (percent)
        let segmentData = SegmentData(percentage: percent, startAngle:
lastEndAngle, endAngle: lastEndAngle + angle)

        self.segments.append(segmentData)
        lastEndAngle += angle
    }
}

}

struct SummaryView_Previews: PreviewProvider {

    static let moc = NSManagedObjectContext(concurrencyType:
.mainQueueConcurrencyType)

    static var previews: some View {
        return SummaryView(totalWorkoutTime: 100, progress: [(70, BodyParts.arms,
10), (30, BodyParts.chest, 10)], segments: [SegmentData(percentage: 70, startAngle:
0, endAngle: 252), SegmentData(percentage: 30, startAngle: 252, endAngle:
360)]).environment(\.managedObjectContext, moc).environmentObject(AppSettings())
    }
}

struct PieChart: View {

    var progress: [(Double, BodyParts, Double)] = []
    var total : Double {
        let durations = progress.map { $0.0 }
        return durations.reduce(0.0, +)
    }
}

```

```

    }
    var totalSessions: Double {
        let durations = progress.map { $0.2 }
        return durations.reduce(0.0, +)
    }

    var segments: [SegmentData] = []
    @State private var shouldShowChart = false

    var body: some View {
        GeometryReader { geoProxy in
            if self.shouldShowChart {
                ZStack {
                    ForEach(0..

```

```

        }
    }

}

struct SegmentData {
    var percentage: Double
    var startAngle: Double
    var endAngle: Double
}

struct Segment: Shape {

    var radius: CGFloat
    var startAngle: Double
    var endAngle: Double

    func path(in rect: CGRect) -> Path {
        var segmentPath = Path()
        segmentPath.move(to: CGPoint(x: rect.midX, y: rect.midY))
        segmentPath.addArc(center: CGPoint(x: rect.midX, y: rect.midY), radius:
radius, startAngle: .degrees(startAngle), endAngle: .degrees(endAngle), clockwise:
false)
        return segmentPath
    }

}

```

Лістинг RestTimerView.swift

```

import SwiftUI

import Combine

enum RestTimerStatus {

    case notStarted, playing, paused

}

struct RestTimerView: View {

    @EnvironmentObject var appSettings: AppSettings

    @State private var offset: CGFloat = 70

```

```

@State private var completedTime: TimeInterval = 0

@State private var shouldShowMenus = false

@State private var status: RestTimerStatus = .notStarted

@State private var timer = Timer.publish(every: 5000, on: .main, in:
.common).autoconnect() // Dummy initialiser with dummy TimeInterval

@State private var backgroundAt = Date()

@Binding var alignment: Alignment

var progress: CGFloat {

    CGFloat((appSettings.workoutTimerInterval - completedTime) /
appSettings.workoutTimerInterval)

}

var body: some View {

    ZStack {

        Color.clear

        .frame(width: 0, height: 0)

        .onReceive(NotificationCenter.default.publisher(for:
UIApplication.willResignActiveNotification)) { (_) in

            self.stopTimer()

            self.backgroundAt = Date()

        }

        .onReceive(NotificationCenter.default.publisher(for:
UIApplication.willEnterForegroundNotification)) { (_) in

            if self.status == .playing {

                let backgroundInterval =
TimeInterval(Int(Date().timeIntervalSince(self.backgroundAt) + 1))

                if (self.completedTime + backgroundInterval) >=
(self.appSettings.workoutTimerInterval - 1) {

```

```

        self.resetDetails()

    } else {

        self.completedTime += backgroundInterval

        self.startTimer()

    }

}

}

if shouldShowMenus {

    Color.white

        .opacity(0.7)

        .edgesIgnoringSafeArea(.all)

}

// Main timer view button

ZStack {

    if status == .playing || status == .paused {

        Circle()

            .trim(from: 0, to: progress)

            .stroke(appSettings.themeColorView(), style:
StrokeStyle(lineWidth: 7, lineCap: .round))

            .animation((status == .playing || status == .paused) ?
Animation.linear(duration: 1) : nil)

            .rotationEffect(.degrees(-90))

            .frame(width: shouldShowMenus ? 220 : 70, height:
shouldShowMenus ? 220 : 70)

    }

}

```

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Text("\((Int(appSettings.workoutTimerInterval - completedTime) ==
0) ? Int(appSettings.workoutTimerInterval) : Int(appSettings.workoutTimerInterval -
completedTime))s")

        .font(kPrimaryLargeTitleFont)

        .bold()

        .foregroundColor(.white)

        .frame(width: shouldShowMenus ? 200 : 50, height:
shouldShowMenus ? 200 : 50)

        .background(appSettings.themeColorView())

        .clipShape(Circle())

    Button(action: {

        mainTimerViewButtonAction()

    }) {

        Color.clear

    }

    .frame(width: shouldShowMenus ? 200 : 50, height: shouldShowMenus ?
200 : 50)

    .padding(10)

    .shadow(radius: shouldShowMenus ? 5 : 0)

}

.offset(y: shouldShowMenus ? -(offset) : 0)

.animation(.spring())

.onReceive(NotificationCenter.default.publisher(for:
Notification.Name.didSaveTodayWorkoutSet)) { (_) in

    mainTimerViewButtonAction(isFromNotification: true)

}

// Minus button

Button(action: {

```

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Helper.hapticFeedback(style: .soft)

        if self.appSettings.workoutTimerInterval > 5 {

            self.appSettings.workoutTimerInterval =
self.appSettings.workoutTimerInterval - 5

        }

    }) {

        Image(systemName: "minus")

        .timerControlStyle(backgroundColor: Color.blue)

    }

    .shadow(radius: shouldShowMenus ? 5 : 0)

    .offset(x: shouldShowMenus ? -offset : 0, y: shouldShowMenus ? (offset
+ 20) : 0)

    .animation(.spring())

// Plus button

Button(action: {

    Helper.hapticFeedback(style: .soft)

    self.appSettings.workoutTimerInterval =
self.appSettings.workoutTimerInterval + 5

}) {

    Image(systemName: "plus")

    .timerControlStyle(backgroundColor: Color.blue)

}

    .shadow(radius: shouldShowMenus ? 5 : 0)

    .offset(x: shouldShowMenus ? offset : 0, y: shouldShowMenus ? (offset +
20) : 0)

    .animation(.spring())

// Stop button

```

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if status == .playing {

    Button(action: {

        Helper.hapticFeedback()

        self.resetDetails()

        NotificationHelper.resetTimerNotification()

    }) {

        Image(systemName: "stop")

        .timerControlStyle(backgroundColor: Color.red)

    }

    .shadow(radius: shouldShowMenus ? 5 : 0)

    .offset(y: shouldShowMenus ? (offset + 20) : 0)

    .zIndex((status == .playing || status == .paused) ? 10 : 0)

}

// Menu/Close button

if (status == .notStarted) {

    Button(action: {

        Helper.hapticFeedback()

        self.shouldShowMenus.toggle()

        self.alignment = shouldShowMenus ? .center : .bottomLeading

    }) {

        Image(systemName: shouldShowMenus ? "xmark" : "timer")

        .font(kPrimaryTitleFont)

        .frame(width: 50, height: 50)

        .background(shouldShowMenus ? Color.secondary.opacity(0.2)
: appSettings.themeColorView())

        .foregroundColor(shouldShowMenus ?
Color.secondary.opacity(0.5) : .white)

```

```

        .clipShape(Circle())

    }

    .shadow(radius: 5)

    .offset(y: shouldShowMenus ? (offset + 20) : 0)

    .zIndex((status == .playing || status == .paused) ? 0 : 10)

    }

}

.padding(.leading, shouldShowMenus ? 0 : 10)

.onReceive(timer, perform: { (_) in

    if self.status == .playing {

        self.completedTime += 1

        if self.completedTime >= self.appSettings.workoutTimerInterval - 1
{
            DispatchQueue.main.asyncAfter(deadline: .now() + 1) {

                self.resetDetails()

            }

        }

    }

    print("⚠️ \$(self.completedTime)")

})

}

fileprivate func mainTimerViewButtonAction(isFromNotification: Bool = false) {

    Helper.hapticFeedback()

    var delay: TimeInterval = 0

    if isFromNotification {

        self.shouldShowMenus.toggle()

    }

}

```

```

        self.alignment = shouldShowMenus ? .center : .bottomLeading

        delay = 1 // We need to wait for an second before auto starting rest
timer
    }

    DispatchQueue.main.asyncAfter(deadline: .now() + delay) {

        if self.status == .playing {

            self.status = .paused

        } else if self.status == .paused || self.status == .notStarted {

            self.status = .playing

        }

        if self.status == .playing {

            self.startTimer()

            NotificationHelper.addLocalNoification(type:
.interval(TimeInterval(self.appSettings.workoutTimerInterval -
self.completedTime)))

        } else if self.status == .paused {

            self.stopTimer()

            NotificationHelper.resetTimerNotification()

        }

    }

}

func resetDetails() {

    self.status = .notStarted

    self.stopTimer()

    self.completedTime = 0

    self.shouldShowMenus = false

    self.alignment = .bottomLeading

```

```

    }

    func stopTimer() {
        self.timer.upstream.connect().cancel()
    }

    func startTimer() {
        self.timer = Timer.publish(every: 1, on: .main, in: .common).autoconnect()
    }
}

struct TimerView_Previews: PreviewProvider {
    static var previews: some View {
        RestTimerView(alignment:
            .constant(.bottomLeading)).environmentObject(AppSettings())
    }
}

struct TimerControlStyle: ViewModifier {

    var backgroundColor: Color

    func body(content: Content) -> some View {
        content
        .font(kPrimaryTitleFont)
        .frame(width: 50, height: 50)
        .background(backgroundColor)
    }
}

```

```

        .foregroundColor(.white)

        .clipShape(Circle())
    }
}

extension View {

    func timerControlStyle(backgroundColor: Color) -> some View {
        self.modifier(TimerControlStyle(backgroundColor: backgroundColor))
    }
}

```

					ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		