МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Інформаційна підсистема організації роботи з відвідувачами залу функціонального тренінгу "Парашут"»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студентка групи ІТ.м-01 Рєзнікова Аліса Сергіївна

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» грудня 2021 р.

Науковий керівник _____ к.т.н., доц., Ващенко С.М.

Голова комісії _____ Шифрін Д.М.

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2021

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. кафедри ІТ

_____ В. В. Шендрик
«___» _____ 2021 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Рєзнікова Аліса Сергіївна
(прізвище, ім'я, по батькові)

1 Тема проекту *Інформаційна підсистема організації роботи з відвідувачами залу функціонального тренінгу "Парашут"*

затверджена наказом по університету від «__»_____2021 р. № _____

2 Термін здачі студентом закінченого проекту «_17_» ___грудня___ 2021 р.

3 Вхідні дані до проекту Вимоги до програмного забезпечення від замовника, перелік файлів для розробки дизайну, дані для заповнення бази даних

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) Аналіз предметної області, постановка задачі та методи дослідження, проектування інформаційної підсистеми, розробка інформаційної підсистеми.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Актуальність та мета розробки, мета та цілі проекту, аналіз існуючих аналогів, планування та проектування, інструменти та етапи розробки

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____

(підпис)

Завдання прийняв до виконання _____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Формування завдання та проблеми проекту	18.10.2021	
2	Формування та деталізація мети проекту	19.10.2021 - 20.10.2021	
3	Аналіз програмних продуктів-аналогів	21.10.2021 - 22.10.2021	
4	Розробка WBS	25.10.2021 - 26.10.2021	
5	Розробка OBS	28.10.2021	
6	Створення діаграми Ганта	28.10.2021 - 29.10.2021	
7	Проведення аналізу ризиків	01.11.2021 - 03.11.2021	
8	Створення IDEF-діаграми	04.11.2021 - 07.11.2021	
9	Створення Telegram-bot для розкладу	08.11.2021 - 11.11.2021	
10	Створення календпрю тренувань	12.11.2021 - 22.11.2021	
11	Створення Telegram-bot для опитувань	23.11.2021 - 30.11.2021	
12	Створення сторінки юзерів	01.12.2021 - 06.12.2021	
13	Створення сторінки статистики здоров'я	07.12.2021 - 12.12.2021	
14	Розробка тест-кейсів	13.12.2021 - 16.12.2021	
15	Тестування за тест-кейсами	17.12.2021 - 21.12.2021	
16	Презентація проекту	22.12.2021	

Магістрант _____ Рєзнікова А.С.

Керівник роботи _____ к.т.н., доц. Ващенко С.М.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Інформаційна підсистема організації роботи з відвідувачами залу функціонального тренінгу "Парашут"».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 32 найменувань, 2 додатків. Загальний обсяг роботи − 90 сторінок, у тому числі 57 сторінок основного тексту, 2 сторінки списку використаних джерел, 33 сторінок додатків.

Кваліфікаційну роботу магістра присвячено розробці інформаційної підсистеми організації роботи з відвідувачами окремого тренувального залу, що включає у себе Telegram-бота, кабінет користувача та сайт-візитку.

В роботі проведено аналіз сучасних досліджень в області впливу використання фітнес-додатків на здоров’я людини та доведена актуальність їх розробки, досліджено область існуючих систем-аналогів, виконана постановка завдання.

У роботі виконано проектування підсистеми, розробку IDEF0 та Use Case діаграм, проведено планування робіт.

Результатом проведеної роботи є розроблена інформаційна підсистема, що включає у себе Telegram-бота, кабінет користувача та сайт-візитку, які використовуватимуться відвідувачами тренажерного залу “Парашут”.

Практичне значення роботи полягає у забезпеченні користувачів зручним інструментом відстеження прогресу власних спортивних досягнень та здоров’я.

Ключові слова: трекінг здоров’я, .NET, TypeScript, статистика, JSON, WEB-ДОДАТОК,TELEGRAM API, MUI.

ЗМІСТ

ВСТУП

Повсякдення людини у сучасному світі включає проведення великої частини часу у сидячому стані, що негативно впливає на стан внутрішніх органів, опорно-рухової системи та нервової системи. У реаліях діджиталізації існує великий спектр можливостей для самостійних тренувань та підтримки здорового стану тіла, наприклад, різноманітні трекери, мобільні додатки, розумні годинники. Але, не дивлячись на таке розмаїття, тренування під наглядом кваліфікованого тренера залишаються найбільш безпечним та ефективним способом підтримки задовільного стану здоров’я.

Для підвищення ефективності відстеження результатів тренувань, а також зручного керування власними відвідуваннями виникла потреба розробити ресурс, який би дозволив відвідувачам тренувального залу “Парашут” отримати доступ до консолідованої інформації та керуванням нею.

Об’єкт дослідження – інформаційні технології підтримки діяльності спортивних залів.

Предмет дослідження – засоби реалізації при організації роботи інформаційних систем.

Наукова новизна роботи полягає в тому, що запропоновано технологію реалізації інформаційної підсистеми організації роботи з відвідувачами тренувального залу з використанням.

Метою дипломної роботи визначено створення інформаційної підсистеми організації роботи з відвідувачами залу функціонального тренінгу "Парашут", що забезпечить зручне керування власним розкладом тренувань, а також дослідження впливу занять на загальний стан здоров’я.

Для досягнення поставленої мети було поставлено ряд завдань:

- провести аналіз предметної області, ознайомитися з вимогами до системи від замовника;

- провести аналіз сучасного стану питання використання відповідних програмних аналогів;
- визначити вимоги до інформаційної підсистеми та складсти технічне завдання;
- провести моделювання роботи підсистеми;
- виконати програмну реалізацію.

Практичне значення виконаної роботи полягає в тому, що відвідувачі тренувального залу отримають зручний інструмент для ведення статистики власного стану здоров’я у залежності від відвідуванних занять, а також встановити схему тренувань на певний проміжок часу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Важливою частиною повсякдення будь-якої людини є фізична активність - це не тільки необхідність звичайного пересування та заняття побутовими справами, а і виконання комплексів вправ, націлених на підтримку здоров’я організму. Згідно з останніми дослідженнями[1], у час глобального локдауну дорослій здоровій людині необхідно проводити від 150 до 300 хвилин на тиждень займаючись руховою активністю щоб знизити ризик виникнення респіраторних захворювань, підвищити стійкість імунної системи і укріпити м’язи серця та судини[2].

Серед великого розмаїття комплексів вправ і фізичних навантажень слід відзначити комплекс вправ для повного опору тіла (Total body resistance exercise TRX)[3]. Згідно з раніше згадуваними дослідженнями, фізичні вправи покращують функціональні можливості, баланс, силу, рухливість та якість життя людей. Даний комплекс використовує вагу тіла як опір під час вправи подібний до щоденних рухів. У поєднанні із регулярним використанням спеціалізованних фітнес-додатків та відстеженням свого прогресу користувачі досягають значного прогресу фізичної витривалості[7].

В умовах глобальної діджиталізаці більшість людей має можливість відстежувати показники свого здоров’я за допомогою різноманітних трекерів та мобільних додатків, які мають за мету не тільки ведення статистики, а і мотивування користувачів до регулярних занять спортом[5,13].

Доктор Ларанхо та дослідницька група проаналізували результати досліджень, проведених між 2014 і 2019 роками для жінок і чоловіків у віці від 18 до 65 років, які використовували різноманітні додатки для смартфонів (включаючи Moves і Accupedo-Pro) або трекери для носіння (включаючи

Fitbit, Fitbug, Withings Activité Steel and Jawbone). Вибірка також включає контрольну групу, яка не використовувала пристрій [12].

Дослідження продуктивності пристрою проводилися за допомогою самозвітів учасників і даних, зібраних від трекерів додатків і акселерометра, які також відстежували рівні активності учасників. Деякі види діяльності включали: щоденний підрахунок кроків, підрахунок хвилини на тиждень від помірної до інтенсивної фізичної активності, щотижневі дні тренувань, хвилини загальної фізичної активності на тиждень та вимірювання споживання кисню організмом під час тренування.

Цей аналіз є одним із перших, які використовувалися новітніми трекерами та додатками відстеження стану здоров'я. Завдяки дослідженням були розроблені девайси з використанням новітніх технологій, які відстежують активність і автоматично дають зворотний зв'язок, на відміну від, наприклад, зі старих пристроїв, які потрібно під'єднати до комп'ютера для завантаження даних про вправи. Велика частина роботи також виконана для людей із хронічними захворюваннями[8].

Попередні дослідження визначили, що старі пристрої призвели до високого або помірного збільшення активності[11]; метою цього огляду було перевірити, чи мають нові технології значний вплив на мотивацію до занять фізичними активностями[14].

Пристрої та програми були основними інструментами для мотивації та вимірювання фізичної активності в дослідженні, але учасники опитування також отримували підтримку, заохочення та допомогу у встановленні цілей та вирішенні проблем від інших учасників або керівників дослідження через зустрічі, телефонні дзвінки, електронні листи або тексти повідомлень[10].

Після середнього періоду спостереження в 13 тижнів (тривалість тесту коливалася від 2 до 40 тижнів) користувачі додатків і трекерів були більш активними, ніж контрольні групи, виходячи з щоденної кількості кроків[3].

Проаналізувавши вищезазначене, можна зробити висновок, що ефективність використання трекерів стану здоров'я є обгрунтованною. При

систематичних заняттях спортом та відстежуванні своїх результатів за допомогою додатків користувач не тільки знижує ризик захворювань та/або проводить їх профілактику, а і зберігає мотивацію, що є фактором позитивниго впливу на психічне здоров'я. Таким чином, у даному дослідженні проводитиметься аналіз даних, отриманих при розробці і використанні інформаційної підсистеми з метою виявлення ефективності відвідування тренувань.

1.2 Аналіз аналогів та визначення наявних проблем

Перед початком роботи над проектом необхідно визначити, чи є його створення актуальним з огляду на існування вже існуючих рішень. Попередньо визначено, що незаперечним фактом є твердження про ефективність відстеження прогресу спортивних тренувань за допомогою інформаційної підсистеми з огляду на ментальний стан здоров'я відвідувача, а також питання керування фінансами учасника групи[12]. У даному випадку дані для проекту отримуються від залу спортивнихтренувань "Парашут", що робить проект індивідуальною розробкою для втілення авторського підходу окремого тренера. У даному розділі розглянемо і порівняємо фітнес-орієнтовані додатки для індивідуальних занять, а також поточний підхід, що використовується тренером.

Спочатку розглянемо поточну організацію процесу тренувань з точки зору відвідувача групи. На даний момент у спортивному залі діє гнучка система організації: користувач може відвідати заняття одноразово, а також придбати місячний абонемент на вісім, десять та дванадцять занять. Абонемент являє собою картку формату візитівки із полями, де визначено кількість занять, ім'я відвідувача та поля для відміток про відвідування.

Кожна картка виконана із цупкого картону з авторським дизайном, розробленим під власне замовлення тренера. Така система має певні недоліки: користувачі часто забувають або втрачають картки, забувають встановити відмітки про відвідування, а друк і заповнення абонементу віднімають багато часу у обох сторін.

Рисунок 1.1 – Абонемент тренувального залу “Парашут”

Основною перевагою даного варіанту вирішення проблеми трекінгу занять є безпосередньо можливість відстеження фінансової сторони питання відвідувань як для тренера, так і для користувача, а також промоуція залу завдяки дизайну на оборотній стороні картки. Також, вона містить важливу інформацію для користувача - телефон тренера і адреса залу.

Рисунок 1.2 – Абонемент тренувального залу “Парашут”

Згідно предмету дослідження проведемо аналіз програмних проектів, що дозволяють спортсменам займатися функціональними тренуваннями самостійно[17]. Першим є мобільний додаток «Упражнения с петлями». Даний додаток містить достатньо широкий спектр функціоналу: можна обрати секцію вправ для певної групи м’язів, зібрати власну систему тренувань із відеозаписами, встановити розклад занять і нагадувань про їх початок. Інтерфейс мобільного додатку мінімалістичний та інтуїтивно зрозумілий. До недоліків можна віднести те, що додаток містить рекламу, а також платний фунціонал. Слід зазначити, що такі тренування можуть бути потенційно небезпечними, адже проходять без нагляду тренера.

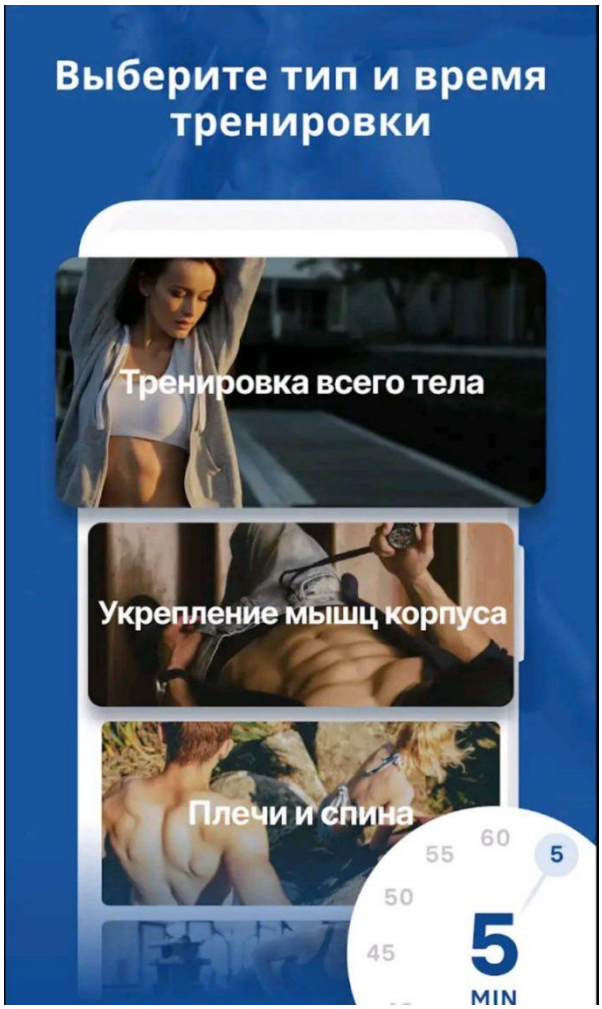

Рисунок 1.3 – Інтерфейс мобільного додатку Упражнения с петлями

Наступний додаток – TRX. Він також являє собою збірку тренувань із більш ніж 200 вправ, але, на відміну від свого попередника, не має можливості створення власних вибірок і розкладів. Підбірки записані професійними тренерами, розподілені на різні рівні фізичної підготовки[6]. Недоліками даного продукту можно визначити платний функціонал, відсутність нагляду спеціаліста, а також недопрацювання, що проявляється у тому що екран загасає під час відтворення відеозапису. Останній пункт можна назвати достатньо великою проблемою, адже користувачеві доводиться відволікатись під час занять.

Рисунок 1.4 – Додаток TRX

Наступним додатком для порівняння є Stark Suspension. Додаток, як і його попередники, представляє собою збірку вправ. Також завдяки даному

додатку можна відстежувати прогрес і спалені калорії під час занять, обирати рівні підготовки та групи м’язів, над якими будуть працювати. Інтерфейс данногододатку також є мінімалістичним. Даний додаток є платним, а також його використання зумовлюєвідсутність тренера під час занять.

Рисунок 1.5 – Мобільний додаток Stark Suspension

Провівши аналіз переваг та недоліків вищеперелічених підходів було проведено порівняння за визначиними критеріями і сформовано табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Назва критерію	Назва ресурсу				
	Абонемент -картка	Упражнения с петлями	TRX	Stark Suspension	Розроблювана інформаційна підсистема
Заняття із тренером	+	-	-	-	+
Трекінг ментального стану	-	-	-	-	+
Можливість безкоштовної роботи	-	+/-	-	-	-
Трекінг власних досягнень	-	+	-	+	+
Керування відвідуваннями	+	+	-	-	+

Виходячи із аналізу та ознайомившись з підходом тренера і відвідувачів до тренувань, вивчивши дану методологію викладання, а також дослідивши функціонал додатків для функціональних тренувань, можна сказати, що дана розробка є актуальною та доцільною[15]. Першочергово, інформаційна підсистема розроблюється як індивідуальне замовлення, тому відповідно отриманий продукт буде безкоштовним для всіх відвідувачів тренувального залу “Парашут”.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Головною метою даної дослідницької роботи є вплив використання інформаційної підсистеми на регулярність тренувань відвідувачів спортивного залу “Парашут”, а також аналіз рівня вмотивованності та ментального стану здоров’я користувачів. Для втілення даних задач потрібно реалізувати методи збору інформації від користувачів за допомогою заповнення відповідних полів профілю та його регулярного оновлення. Динаміка відстежуватиметься за наступними критеріями:

- маса тіла;
- лінійні заміри, стандартні при визначенні розміру одягу;
- ментальний стан;
- якість сну;
- регулярність відвідувань.

Дані показники визначені як основні виміри виходячи із досліджень, розглянутих у попередньому розділі. Метою створення інформаційної підсистеми організації роботи з відвідувачами залу функціонального тренінгу "Парашут" полягає у тому, що продукти із програмною реалізацію, що наявні на ринку мобільних додатків не задовольняють вимогам користувачів і тренера у повній мірі, не є безкоштовними, а також не відповідають тренерській методі замовника.

Таким чином, було встановлено наступний ряд задач на виконання розробки:

- вибір технологій для розробки додатку;
- проектування бази даних для збереження даних користувачів;
- реалізація частини фронтенду для користувацького інтерфейсу;

- реалізація фронтенду для виведення графіків динаміки прогресу;
- реалізація інтеграції із Telegram, що дозволить користувачам відмічатись на заняттях.

Вихідна сторінка повинна служити сайтом-візиткою, яка буде містити фото-матеріали облаштування залу, а також відео-матеріали з прикладами проведення занять. Вона також буде слугувати вхідною точкою до модулю управління за допомогою Telegram Web API.

Дана інформаційна підсистема має представляти собою особистий кабінет відвідувача тренажерного залу “Парашут” і бот для того, щоб відмічати свою майбутню присутність на занятті. Метою розроблюваного проекту є забезпечення користувачів зручною системою керування відвідуваннями тренувань, відстеженням прогресу фізичного здоров’я, зовнішнього вигляду та настрою [4].

Першочергово, дана підсистема спрямована на використання відвідувачами окремого тренувального залу. Система має бути реалізована у вигляді набору сторінок, що будуть завантажені у мережу Інтернет як частина інформаційної системи. Сторінки мають бути логічно пов’язані між собою, дозволяючи користувачеві зручно і швидко орієнтуватися. Для роботи із інформаційною підсистемою користувачеві вистачить загальних навичок роботи з персональним комп'ютером та/або смартфоном і браузером. Підсистема має складатися із наступних розділів:

- Головна сторінка – перелік розділів: точки, лінії, полігони; блок із обранням мови проходження тесту.

- Telegram-бот, щодо зволяє користувачам швидко відмічати зручний час та день відвідання тренування.

- Сторінка із власними параметрами, де можна вносити свою особисту інформацію та побачити графіки змін стану здоров’я.

- Сторінка із тестами, де користувач може оцінити стан самопочуття за день.

Інтерфейс додатку має бути інтуїтивно зрозумілим, забезпечувати швидкий перехід до інших сторінок. Система повинна забезпечувати навігацію за всіма доступними користувачеві ресурсами і відображати відповідну інформацію.

Стиль та кольори інформаційної підсистеми мають бути обрані на основі вже існуючого брендингу тренувального залу “Парашут”.

2.2 Методика оцінки стану відвідувача

Джерелом інформації для проведення аналізу стану відвідувача клубу є сторінка особистого кабінету користувачів інформаційної підсистеми, які регулярно відвідують заняття [20]. Вплив ведення регулярного відстеження стану користувача за зазначеними критеріями має визначатись за допомогою графіків, що наочно відображають зміни [19].

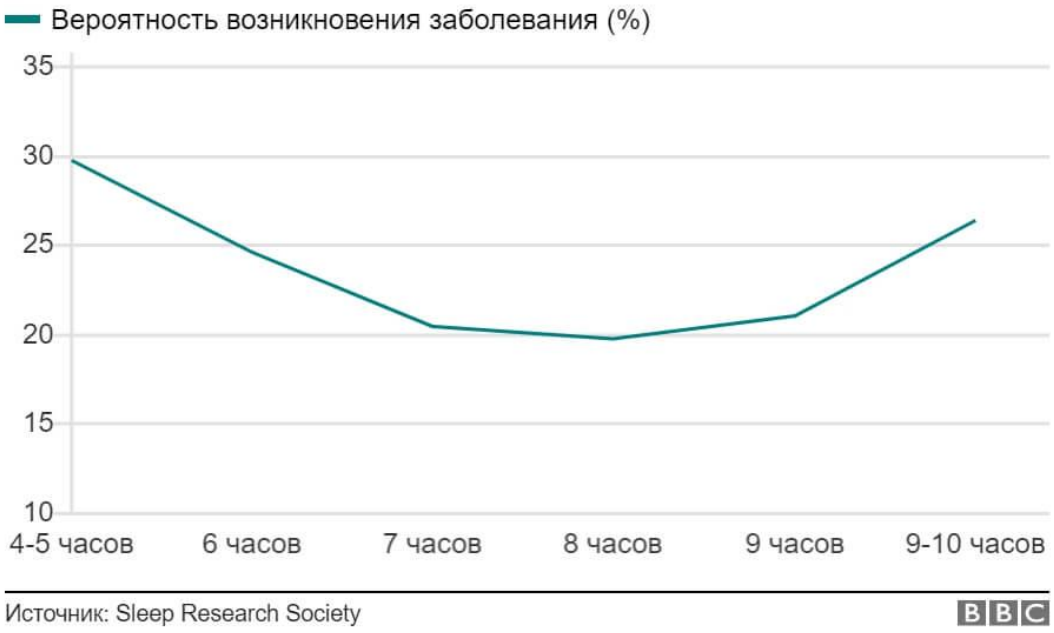

Рисунок 2.1 – Приклад графіку функції залежності сили супротиву імунної системи від кількості годин сну.

Зміна маси тіла індивідуума є важливим показником при діагностиці хвороб, а також загальним показником стану здоров’я. Часто надто швидка динаміка даного показника вказує на наявність хвороб - як стрімке збільшення ваги може вказувати на гормональні розлади, так і зниження може бути показником аутоімунних хвороб. Даний показник передається у систему безпосередньо користувачем у спеціально відведене поле, вага вказується у кілограмах.

Лінійні заміри тіла у динаміці мотивують користувача, а також, з огляду на регулярність фізичних вправ, дозволяють визначити ріст м’язової тканини і зменшення жирової тканини у окремо взятих частинах тіла. Заміри також вносяться у систему користувачем самостійно у сантиметрах: зріст, охват грудей, талії, стегон.

Відстеження ментального стану спортсменів є великою частиною дослідження, оскільки на нього впливають одночасно велика кількість факторів. У даному дослідженні вплив відстежується за допомогою заповнення двох простих тестів, що складаються зі стандартних питань. Тест проходять двічі, щоб встановити вплив фізичної активності на настрій піддослідного - до та після тренування. Користувачеві пропонується двічі оцінити настрій за десятибальною шкалою, де 10 - відмінний настрій, а 1 - дуже поганий.

Не можна недооцінювати вплив сну на рівень продуктивності людини, тому дана характеристика також відстежуватиметься. Важливо відслідкувати кореляцію між якістю сну та кількістю тренувань на тиждень: таким чином можна визначити, чи не було заняття занадто інтенсивним, або ж, навпаки, покращило самопочуття. Оцінка якості сну проводиться за тим же принципом, що і оцінка самопочуття.

Відстеження регулярності відвідувань є показником з яким проводитиметься кореляція відносно змін з вищеназваними критеріями - тобто, наприклад, залежність загального ментального стану від регулярності відвідувань. Також

це допомагає користувачеві наочно побачити, що вкладені у тренування фінанси окупаються у аспектах здоров’я і мотивують займатися регулярно.

Таким чином, будуть проводитись наступні розрахунки і виводитись доступні користувачеві графіки для наочного відображення прогресу:

1) ІМТ (індекс маси тіла) - розраховуєтьсяза формулою:

$$\text{ІМТ} = \frac{m}{h^2} \tag{2.1}$$

де m - маса тіла користувача у кілограмах, h - зріст у метрах.

Після цього, розраховується зміна ІМТ за часом, у даному випадку - за кількістю відвіданих занять. Фінальна формула залежності:

$$y=F(x) \tag{2.2}$$

де x - значення ІМТ, F - лінійна залежність, y - значення функції залежності індексу від кількості тренувань.

2) Аналогічно розраховується залежність зміни охватів грудної клітини, стегон та талії в залежності від тренувань, а також якість сну.

3) Статистика ментального здоров’я розраховується із двох окремих наборів даних: знаходимо різницю у цифровому еквіваленті оцінок до та після тренувань (наприклад, 8-5, 7-3, 6-2) і проводимо побудову графіку лінійної функції. Такі розрахунки дозволять визначити загальну тенденцію зміни настрою у наслідок регулярного відвідання залу.

Зібравши та провівши аналіз перелічених даних, а також продемонструвавши їх користувачеві наочно, користувач матиме можливість зробити власні висновки про ефективність тренувань у зв’язці із веденням щоденника.

3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ

3.1 Моделювання в IDEF0

Проектування веб-додатку починається із розробки контекстної діаграми А-0. Дана діаграма представляє собою спрощений вигляд роботи системи, дає наочно зрозумілий опис компонентів та чинників впливу, а також демонструє взаємодію системи із зовнішніми об’єктами.

Провівши аналіз компонентів системи, а також вивчивши вимоги до неї, визначено наступний перелік:

1) Входи – дані про виміри користувача, дані про оцінки самопочуття, голосування в Telegram, дані про активного користувача Telegram;
2) Дані управління – вимоги до відображення та збереження інформації на веб-сторінках, інструкція користувача;
3) Виходи – інформація про відвідування тренувань, статистика динаміки стану здоров’я;
4) Механізми – відвідувач, Telegram Bot, Web-додаток.

Контекстна діаграма А-0 представлена на рисунку 3.1.

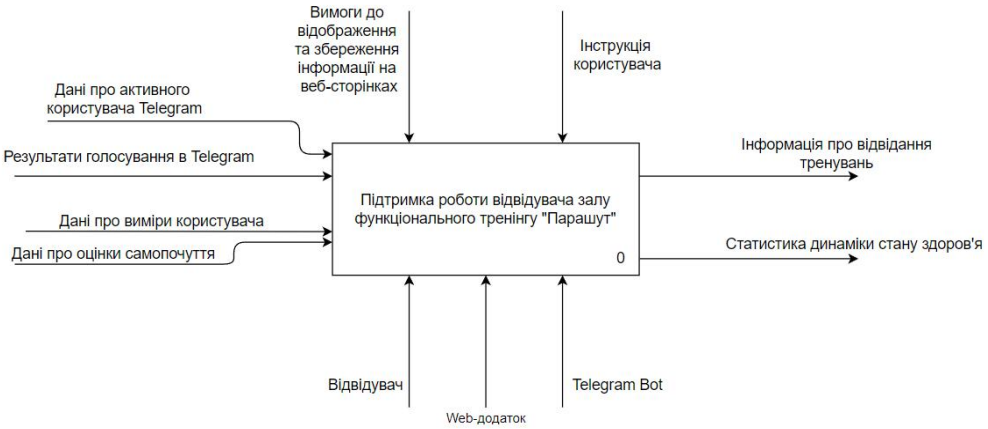

Рисунок 3.1 – Контекстна діаграма А-0

Діаграма A-0 надає представлення про систему в узагальненому вигляді, тому наступним кроком проводиться поетапна декомпозиція, яка дозволить наочно і більш детально представити послідовність роботи процесів розроблюваної системи.

Розбиваємо блок A-0 на набір окремих функцій, що відповідають більш конкретним процесам системи. Результатом декомпозиції стає діаграма IDEF1, яка зображена на рисунку 3.2.

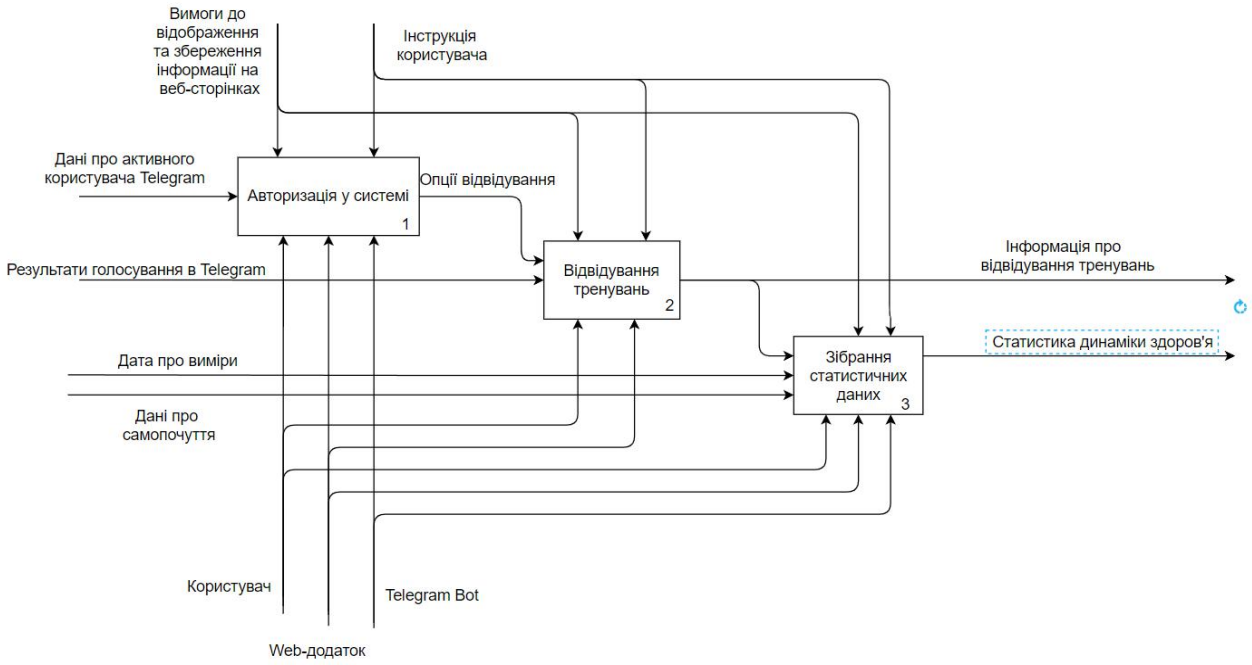

Рисунок 3.2 – Модель декомпозиції першого рівня

Для подальшого представлення роботи кожної підфункції проводимо декомпозицію другого рівня. Таким чином, принципи функціонування розроблюваної інформаційної підсистеми групуються у логічні блоки. Діаграми наведено на рисунках 3.3 та 3.4.

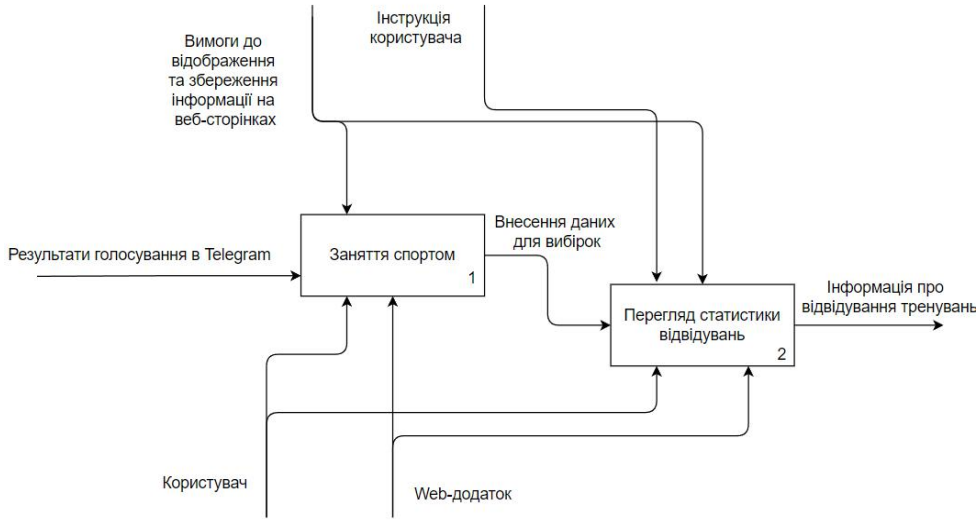

Рисунок 3.3 – Модель декомпозиції другого рівня підфункції «Відвідування тренувань»

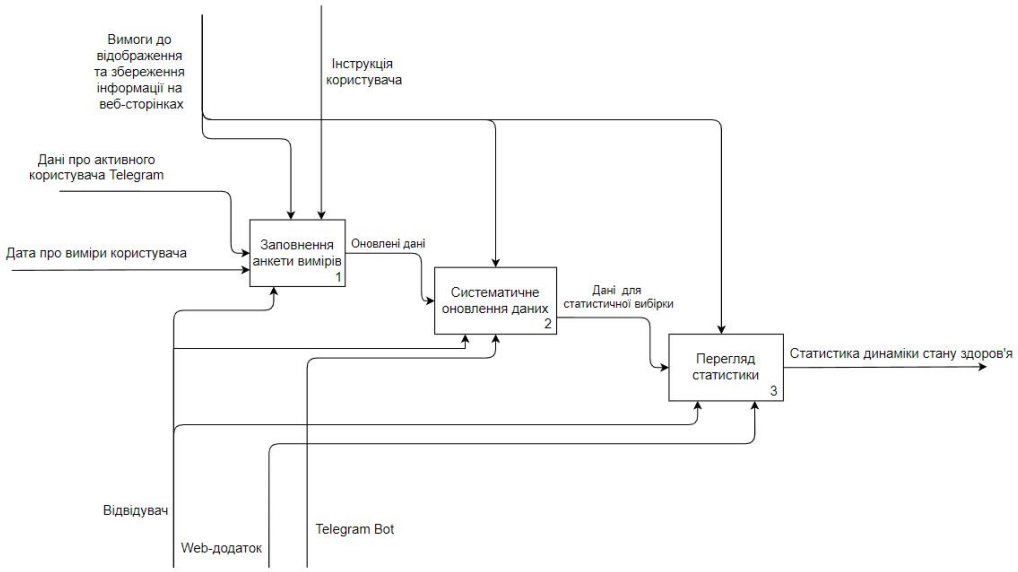

Рисунок 3.4 – Модель декомпозиції другого рівня підфункції «Зібрання статистичних даних»

3.2 Моделювання варіантів використання

Use Case діаграми представляють собою важливий етап розробки проекту, адже вони дають представлення про типові варіанти взаємодії користувача і системи, які мають приводити до конкретного результату. Вони відображують зовнішній інтерфейс системи із доступними опціями, не описуючи внутрішню структуру роботи окремих компонентів.

Для розроблюваної інформаційої підсистеми актором виступає відвідувач тренувального залу.

Визначено наступні артефакти підсистеми: база даних, що зберігає інформацію, яка необхідна для функціонування системи, а також Telegram Bot, за допомогою якого база даних поповнюється інформацією про відвідування тренувань, а також стан здоров’я та виміри користувачів.

Перелік встановлених варіантів використання:

1) ВВ 1 Авторизація – користувач реєструється у системі за допомогою даних із Telegram;
2) ВВ 2 Додавання відмітки про відвідування – користувач має можливість встановити відмітку про відвідання певного заняття за допомогою опитування у Telegram;
3) ВВ 3 Внесення даним про здоров’я – користувач має можливість пройти невелике опитування у Telegram і внести необхідні для збору та подальшого аналізу дані про свої параметри і стан здоров’я;
4) ВВ 4 Перегляд календаря тренувань – користувач має можливість переглянути тренування у вигляді відміток на календарі та спланувати свої відвідування;
5) ВВ 5 Перегляд статистики – можливість переглянути статистику відвідувань, а також динаміку здоров’я та параметрів у залежності від кількості відвідувань;

6) ВВ 6 Відміна відмітки про тренування – можливість скасувати відвідування тренування і опціонально повідомити причину;

– ВВ 7 Перегляд сторінки-візитки – користувач може переглянути початкову сторінку сайту, що представляє дані про зал, тренера і виконана згідно зі стилем замовника.

Діаграма варіантів використання представлена на рисунку 3.5.

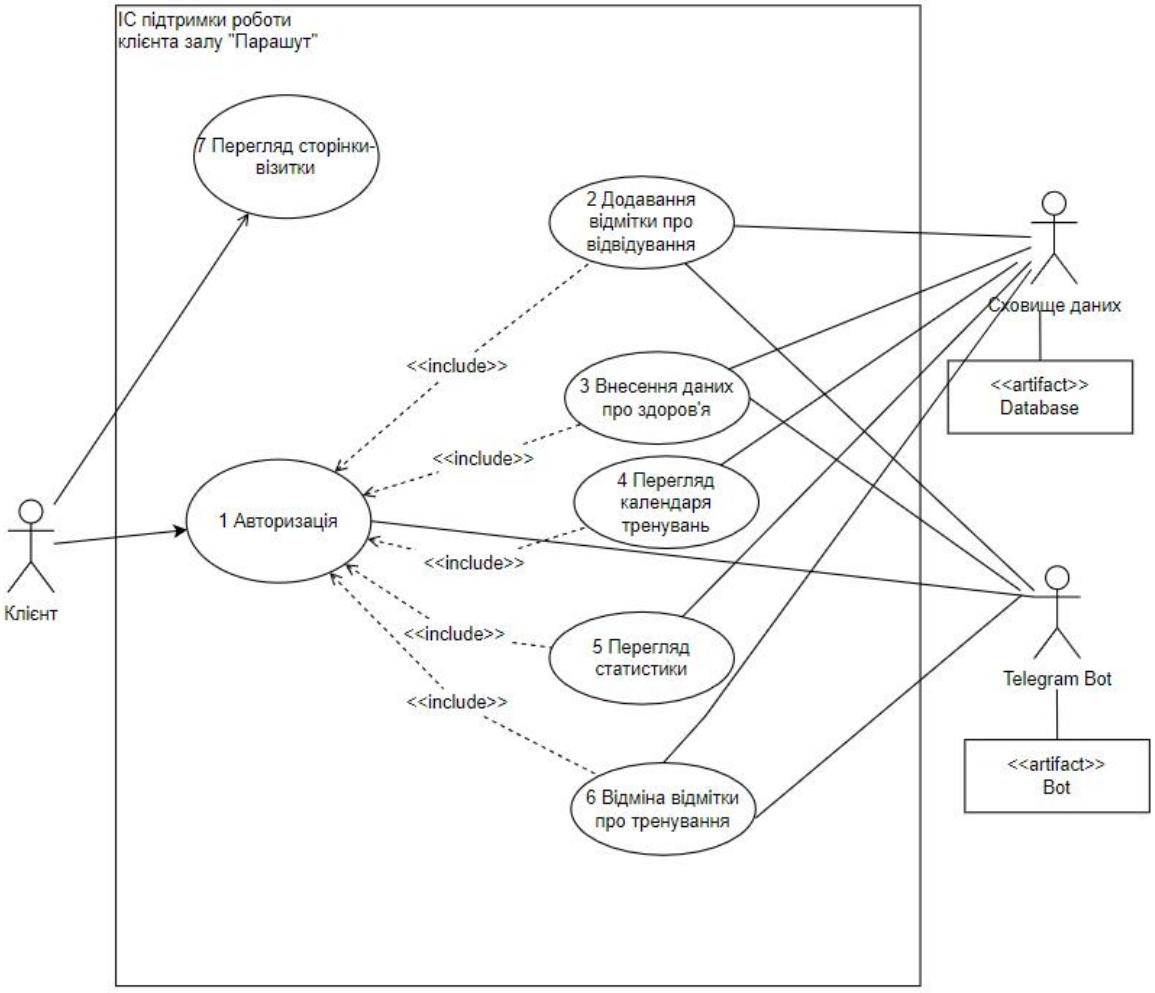

Рисунок 2.5 – Діаграма варіантів використання

3.3 Проектування бази даних

Наступним кроком роботи над проектом є проектування бази даних(БД): було розроблено модель, яка включає необхідні сутності системи та взаємозв’язки між ними.

Для виконання поставленої задачі була обрана реляційна система, що дозволяє встановити внутрішні зв’язки системи і представити сутності увигляді таблиць із атрибутами.

Розробка реляційної моделі у першу чергу встановлює сутності, необхідні для функціонування системи:

1) User – користувач-клієнт залу або тренер;
2) Group – група тренуючихся, до якої належить користувач;
3) MembershipType – тип абонементу який має користувач;
4) Membership – придбаний користувачем абонемент;
5) TrainingSession – тренування, яке користувач може відвідати згідно із календарем;
6) TrainingSessionAttendee – відношення користувача до тренування;
7) UserStats – дані про здоров’я та параметри користувача.

В таблиці 3.1 наведено список атрибутів кожної таблиці та їх опис.

На рисунку 3.6 наведено повну реляційну модель бази даних.

Таблиця 3.1 – Опис атрибутів таблиць

Таблиця	Атрибут	Зміст	Тип	Ключі	Обмеження
User	Id	Унікальний ідентифікатор об’єкту	Varchar(30)	PK	Unique, not null
	GroupId	Ідентифікатор групи	Varchar (30)	FK	Not null
	TelegramUserId	Ідентифікатор користувача в Telegram	Varchar (30)		Not null
	FirstName	Ім’я користувача в Telegram	Varchar (100)		
	LastName	Фамілія користувача в Telegram	Varchar (100)		
	UserName	Нік користувача в Telegram	Varchar (100)		Not null

Продовження таблиці 3.1

	Role	Роль користувача в системі (тренер або тренуючийся)	Varchar (10)		Not null
	CreatedAt	Дата створення об’єкту	Date		Not Null
	UpdatedAt	Дата останнього редагування об’єкту	Date		Not Null
	IsDeleted	Чи користувач покинув групу в Telegram	Boolean		Not null
Group	Id	Унікальний ідентифікатор об’єкту	Varchar(30)	PK	Unique, not null
	Name	Назва групи	Varchar(60)		Not null

Продовження таблиці 3.1

	ChatId	Унікальний ідентифікатор чату групи в Telegram	Varchar(30)		Not null
	TrainerId	Тренер, який тренує групу	Varchar(30)	FK	Not Null
	CreatedAt	Дата створення об’єкту	Date		Not Null
	UpdatedAt	Дата останнього редагування об’єкту	Date		Not Null
MembershipType	Id	Унікальний ідентифікатор об’єкту	Varchar(30)	PK	Unique, not null
	Duration	Тривалість активності абонементу	Timespan		Not Null

Продовження таблиці 3.1

	NumberOfSessions	Кількість занять, що входять до абонементу	int		Not Null
	Cost	Вартість абонементу	int		Not Null
	CreatedAt	Дата створення об’єкту	Date		Not Null
	UpdatedAt	Дата останнього редагування об’єкту	Date		Not Null
Membership	Id	Унікальний ідентифікатор об’єкту	Varchar(30)	PK	Unique, not null
	UserId	Унікальний ідентифікатор користувача, який придбав абонемент	Varchar(30)	FK	Not Null

Продовження таблиці 3.1

	MembershipTypeId	Тип абонементу	Varchar(30)	FK	Not Null
	StartDate	Початок активності абонементу	Date		Not Null
	EndDate	Закінчення роботи абонементу	Date		Not Null
	WasPaidFor	Чи клієнт вже заплатив за абонемент	Boolean		Not Null
	WasExtended	Чи абонемент було продовжено	Boolean		Not Null
	CreatedAt	Дата створення об’єкту	Date		Not Null
	UpdatedAt	Дата останнього редагування об’єкту	Date		Not Null

Продовження таблиці 3.1

TrainingSession	Id	Унікальний ідентифікатор об’єкту	Varchar(30)	PK	Unique, not null
	GroupId	Ідентифікатор групи	Varchar (30)	FK	Not null
	PollId	Унікальний ідентифікатор голосування в Telegram	Varchar (30)		Not null
	Date	Дата проведення заняття	Date		Not null
	Status	Статус заняття (проведено, перенесено тощо)	Varchar (15)		Not null
	CreatedAt	Дата створення об’єкту	Date		Not Null

Продовження таблиці 3.1

	UpdatedAt	Дата останнього редагування об’єкту	Date		Not Null
TrainingSessionAttendee	Id	Унікальний ідентифікатор об’єкту	Varchar(30)	PK	Unique, not null
	SessionId	Унікальний ідентифікатор тренування	Varchar(30)	FK	Not Null
	MembershipId	Ідентифікатор абонементу того, хто відвідав тренування	Varchar(30)	FK	Not Null
	DidCancel	Чи користувач сам відмінив своє відвідування	Boolean		Not Null

Продовження таблиці 3.1

	DidAttend	Чи користувач з’явився на тренуванні	Boolean		
	CreatedAt	Дата створення об’єкту	Date		Not Null
	UpdatedAt	Дата останнього редагування об’єкту	Date		Not Null
UserStat	Id	Унікальний ідентифікатор об’єкту	Varchar(30)	PK	Unique, not null
	UserId	Ідентифікатор користувача, який вносить дані	Varchar(30)	FK	Not Null
	MoodBefore	Оцінка настрою до тренування (1-10)	Int		Not Null
	Bust	Об’єм грудей	Float		Not Null

Продовження таблиці 3.1

	Waist	Об’єм талії	Float		Not Null
	Hips	Об’єм стегон	Float		
	Weight	Вага	Float		Not Null
	Height	Зріст	Float		Not Null
	Sleep	Якість сну (1-10)	Int		
	MoodAfter	Оцінка настрою після тренування (1-10)	Int		Not Null

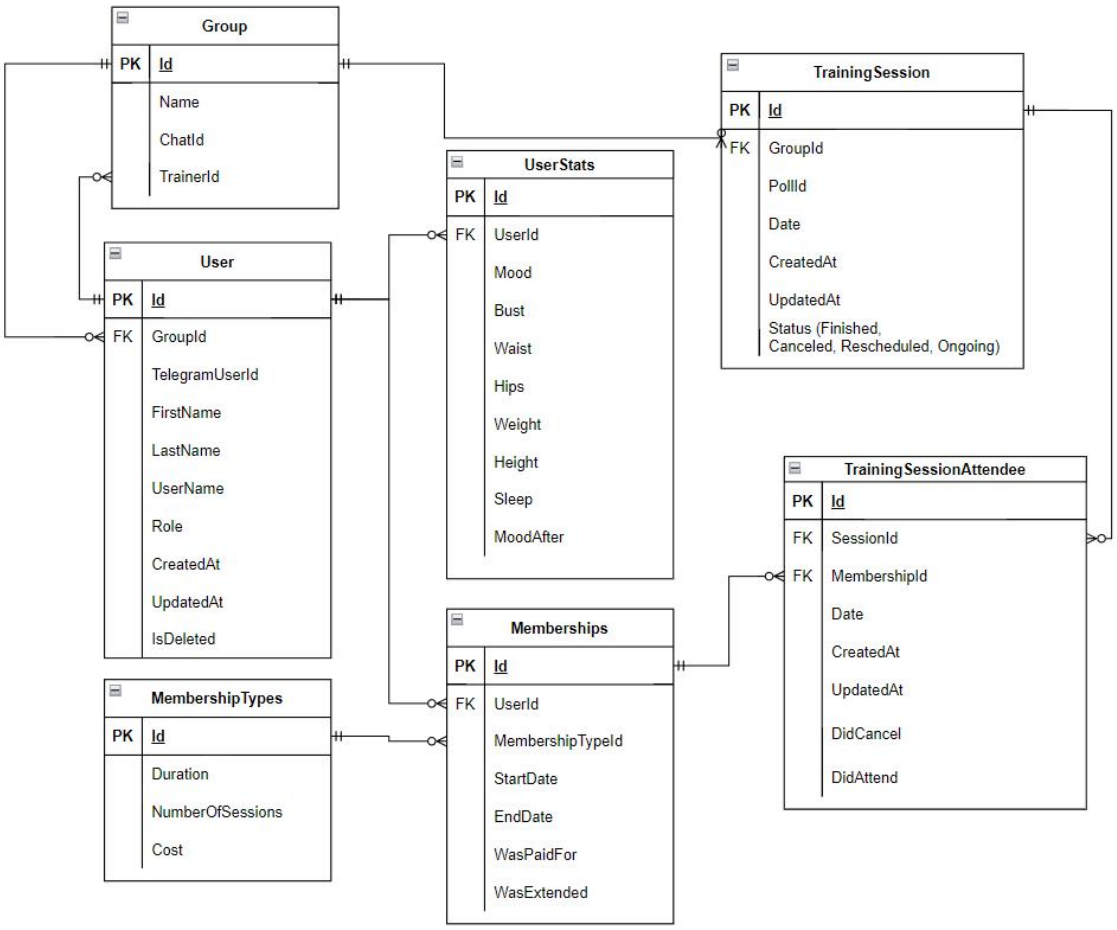

Рисунок 3.7 – Модель бази даних

4 РОЗРОБКА ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ

4.1 Реалізація сайту-візитки

Початок роботи над проектом є створенням відповідного проекту ASP .NET Core у середовищі розробки. Першим кроком стала модифікація файлу Startup.cs,: тут проводиться конфігурація програми, налаштовуються сервіси, які програма буде використовувати, встановлюються компоненти для обробки запитів тощо. Клас Startup визначає метод Configure(), а також метод ConfigureServices().

Перейдемо до деталізації налаштування даного файлу. Встановлюємо шлях зв’язку із базою даних, як зазначено на рис. 4.1:

```
string mySqlConnectionStr = Configuration.GetConnectionString("DefaultConnection");
services.AddDbContextPool<ParashutDbContext>(options =>
    options.UseMySql(mySqlConnectionStr, ServerVersion.AutoDetect(mySqlConnectionStr)
    //TODO: remove for prod
    .LogTo(Console.WriteLine, LogLevel.Information)
    .EnableSensitiveDataLogging()
    .EnableDetailedErrors()
);
```

Рисунок 4.1 – Опис зв’язку із базою даних

Для перевірки стабільності роботи додатку використовується виклик API-методу services.AddHealthChecks();, завдяки якому отримується відповідний статус-відповідь.

Для коректної серіалізації даних змінимо стандартний серіалайзер на newtonsoft.json та введемо потрібні зміни, що відображені на рис. 4.2:

```
services.AddControllers().AddNewtonsoftJson(opt =>
{
    opt.SerializerSettings.MissingMemberHandling = Newtonsoft.Json.MissingMemberHandling.Ignore;
    opt.SerializerSettings.ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.Ignore;
    opt.SerializerSettings.Converters.Add(new StringEnumConverter());
    opt.SerializerSettings.Converters.Add(new UnixDateTimeConverter());

});
```

Рисунок 4.2 – Налаштування серіалізації

Для зв’язку із необхідними модулями програми додамо наступні сервіси як на рис. 4.3:

```
services.AddScoped<ITelegramBotService, TelegramBotService>();
services.AddScoped<IGroupService, GroupService>();
services.AddScoped<IUserService, UserService>();
services.AddScoped<ITrainingSessionService, TrainingSessionService>();
services.AddScoped<ITrainingSessionAttendeeService, TrainingSessionAttendeeService>();
services.AddScoped<IMembershipTypeService, MembershipTypeService>();
services.AddScoped<IMembershipService, MembershipService>();
services.AddScoped<IHubEventsService, HubEventsService>();
```

Рисунок 4.3 – Налаштування серіалізації

Для інтеграції із Telegram Bot API 5.4 додаємо .NET 5 wrapper згідно із офіційною документацією. Даний крок відображено на рис. 4.4:

```
services.AddBotClient("ap1-telegram-token");
services.AddPolling<UpdateHandler>();
```

Рисунок 4.4 – Налаштування інтеграції із Telegram Bot

Для реалізації оновлення даних у реальному часі створюємо наступний метод дозволяє коду сервера надсилати асинхронні сповіщення до веб-програм на стороні клієнта:

```
 services.AddSignalR().AddJsonProtocol(opt =>
                opt.PayloadSerializerOptions.ReferenceHandler = ReferenceHa
ndler.Preserve
            );
```

Зручне тестування API-методів що потребують авторизації за допомогою Bearer-токена досягається за допомогою імплементації та використання сервісу AddSwaggerGen на рис. 4.5:

```
    services.AddSwaggerGen(c =>
    {
        c.SwaggerDoc("v1", new OpenApiInfo { Title = "Parashut.Api", Version = "v1" });
        c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
        {
            Description = @"JWT Authorization header using the Bearer scheme.
              <br><br>Example: 'Bearer 12345abcdef'",
            Name = "Authorization",
            In = ParameterLocation.Header,
            Type = SecuritySchemeType.ApiKey,
            Scheme = "Bearer"
        });

        c.AddSecurityRequirement(new OpenApiSecurityRequirement()
        {
            {
                new OpenApiSecurityScheme
                {
                    Reference = new OpenApiReference
                    {
                        Type = ReferenceType.SecurityScheme,
                        Id = "Bearer"
                    },
                },
                new List<string>()
            }
        });
        var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
        var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);
        c.IncludeXmlComments(xmlPath);
    });
}
```

Рисунок 4.5 – Налаштування методу тестування працездатності API-методів

Для реалізації головної сторінки-візитки було першочергово створено файл, вміст якого зображено на рис. 4.6. Він вміщує в собі основний каркас та посилання на інші компоненти:

```
import React from 'react'
import { LandingPageProps } from './LandingPage.types'
import { CssBaseline } from '@material-ui/core'
import { withRouter, Redirect } from 'react-router-dom'
import { compose } from 'redux'
import { connectToState } from './LandingPage.connect'
import { RouteConstants } from '../../RouteConstants'
import AppBar from './components/AppBar'
import AppFooter from './components/AppFooter'
import Pictures from './components/Pictures'
import Description from './components/ProductValues'
import BannerBase from './components/Banner'

export class LandingPageBase extends React.Component<LandingPageProps> {
  render() {
    if (this.props.currentUser.id) {
      return <Redirect to={RouteConstants.home} />
    }

    return (
      <>
        <CssBaseline />
        <AppBar />
        <BannerBase />
        <Description />
        <Pictures />
        <AppFooter />
      </>
    )
  }
}

export default compose(
  withRouter,
  connectToState,
)(LandingPageBase) as React.ComponentType<LandingPageProps>
```

Рисунок 4.6 – файл LandingPage.tsx.

За допомогою конфігурації файлу LandingPage.connect.ts компонент отримує дані зі сховища. Компонент AppBar.tsx включає в себе віджет для авторизації у системі через Телеграм-API. Для цього було використано npm пакет 'react-telegram-login'.

У компоненті Banner.tsx реалізовано шапку сайту зі слоганом тренувального залу. Компонент Pictures.tsx включає в себе калейдоскоп зображень із “відгуком”. Блок інформації про власницю та тренера тренувального залу ”Парашу”, атакож про користь тренувань для здоров’я реалізовао у компоненті ProductValues.tsx. На футері, реалізованому в компоненті AppFooter.tsx, посилання на соціальні мережі тренера.

4.2 Реалізація Telegram-боту

Однією із важливих частин реалізації розроблюваного проекту стала робота над Telegram-ботом, який дозволятиме користувачам проводити оцінку власного прогресу за допомогою проходження опитування. Даний бот розроблювався для індивідуального користування для того, щоб відвідувач залу міг комфортно вносити дані про свої параметри тіла: користувач звертається до боту та ініціалізує проходження опитування, вносить бажані зміни і в подальшому повторює процес за бажанням.

Для того щоб створити власного бота, першочергово була вивчена відповідна документація із Bot API для коректної роботи із розрооблюваним функціоналом.

Першим кроком потрібно відкрити у Telegram та знайти бота BotFather. Він є офіційним функціоналом для створення власних ботів. Розпочинаємо роботу із ботом вводячи команду /start і отримуємо перелік можливостей. Оберемо створення нового боту із перелікукоманд і впишемо команду /newbot як зазначено на рис. 4.7:

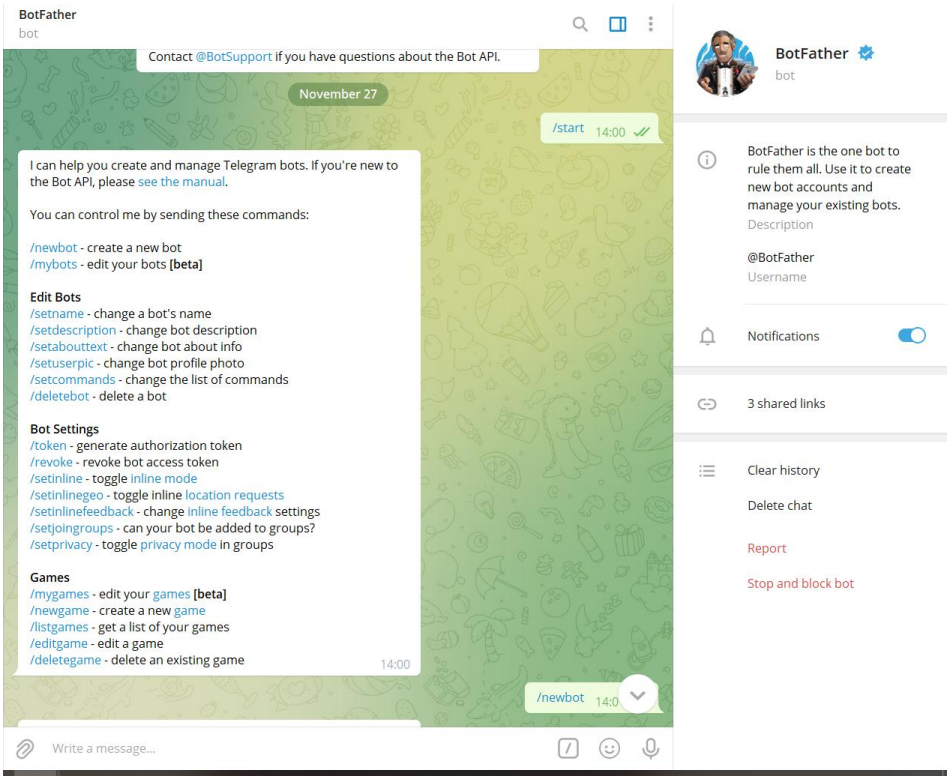

Рисунок 4.7 – Ініціалізація роботи із BotFather

Наступним кроком, даємо боту логічне ім’я із переліку доступних (таких, що ще не зареєстровані) і встановлюємо перелік команд із опціями, що надаватимуться користувачеві.

Оскільки бот є частиною підсистеми, створюємо новий клас TelegramBotService.cs, що відповідає за дії бота у чаті або групі. Наприклад, щоб закріпити надіслане ботом повідомлення у групі, потрібно реалізувати наступний метод (рис. 4.8):

```
public async Task<Result<bool>> PinChatMessage(long chatId, int messageId)
{
    PinChatMessage request = new PinChatMessage(chatId, messageId);
    Response<bool> response = await bot.HandleAsync(request);

    if (response.Ok)
    {
        return Result<bool>.Success(true);
    }
    else
    {
        Failure failure = response.Failure;
        return Result<bool>.Fail(failure.ErrorCode.Value, failure.Description);
    }
}
```

Рисунок 4.8 – Метод для закріплення ботом повідомлень

Для реалізації анкетування користувачів було створено список коректно сформульованих запитань, що надсилатимуться користувачеві, що відображено на рис. 4.9:

```
private readonly string[] questions = new[]
{
    "Зріст (см)",
    "Вага (кг)",
    "Об'єм талії (см)",
    "Об'єм стегон (см)",
    "Об'єм грудей (см)",
    "Настрій до тренування (оцінка від 1 до 10)",
    "Настрій після тренування (оцінка від 1 до 10)",
    "Як спалося (оцінка від 1 до 10)"
};
```

Рисунок 4.9 – Список питань користувачам.

Наприклад, було також реалізовано метод що відповідає за відправлення голосування щодо відвідування тренувань користувачами, що відображено на рис. 4.10. Даний визначено як не анонімний та дозволяє обрати лише одну опцію для відвідування задля запобігання помилок та непорозумінь серед користувачів та тренера. Також даний метод викликає попередньо зазначений метод для

закріплення голосувань, щоб підтримувати інформацію актуальною для наступного відвідування.

```
public async Task<Result<PollMessage>> SendPollToGroup(long chatId, string question, IE
{
    SendRegularPoll request = new SendRegularPoll(chatId, question, options, null)
    {
        IsAnonymous = false,
        AllowsMultipleAnswers = false
    };

    Response<PollMessage> response = await bot.HandleAsync(request);

    if (response.Ok)
    {
        PollMessage message = response.Result;

        await PinChatMessage(chatId, message.Id);

        return Result<PollMessage>.Success(message);
    }
    else
    {
        Failure failure = response.Failure;

        return Result<PollMessage>.Fail(failure.ErrorCode.Value, failure.Description);
    }
}
```

Рисунок 4.10 – Метод для надсилання опитувань у групу користувачів

Для того щоб передавати дані, отриманівід користувачів до бази даних, було також створено окремий клас UpdateHandler.cs, який продемонстровано на рис. 4.11. Наприклад, якщо до групи долучається новий користувач, за допомогою відповідного методу про це вноситься запис до бази даних. Таким же чином, якщо користувач видаляється із групи, метод вносить відповідні зміни:

```
}

private async Task HandleNewChatMembers(NewChatMembersMessage data)
{
    using (var scope = _serviceScopeFactory.CreateScope())
    {
        var userService = scope.ServiceProvider.GetService<IUserService>();
        await userService.AddMembersToGroup(data.Chat.Id, data.Members);
    }
}

private async Task HandleLeftChatMember(LeftChatMemberMessage data)
{
    using (var scope = _serviceScopeFactory.CreateScope())
    {
        var groupService = scope.ServiceProvider.GetService<IGroupService>();
        await groupService.SoftRemoveMember(data.Chat.Id, data.Member);
    }
}
```

Рисунок 4.11 – Методи для додавання та видалення запису про користувачів

4.3 Реалізація кабінету користувача

Наступним етапом є створення кабінету користувача, де він матиме можливість преглядати доступні тренування, а також відображення динаміки стану здоров’я та зміни вимірів тіла у вигляді графіків. Користувач має доступ лише про свою групу та свої дані.

У створеному класі UserController.cs імплементуємо контролер, за допомогою якого програма викликає коректий метод сервісу. У класі UserService.cs відповідно реалізовуються методи, що отримують дані про виміри користувача із Телеграм.

Доступ до сторінки із власною статистикою є лише у кожного окремого користувача персонально, що зроблено із ціллю збереження почуття конфіденційності. Вигляд статистики користувача складається із вкладок, де

окремо виведені графіки за параметрами. Повний лістинг коду наведено у додатку Б.

4.4 Приклади роботи програми

Користувача зустрічає головна сторінка-візитка, яка надає інформацію про тренажерний зал та доступ до контактів тренера. Також відвідувач має можливість зайти на власну сторінку через авторизацію у Телеграм. Приклади відображення сайту-візитки відображено на рисунках 4.12-4.14.

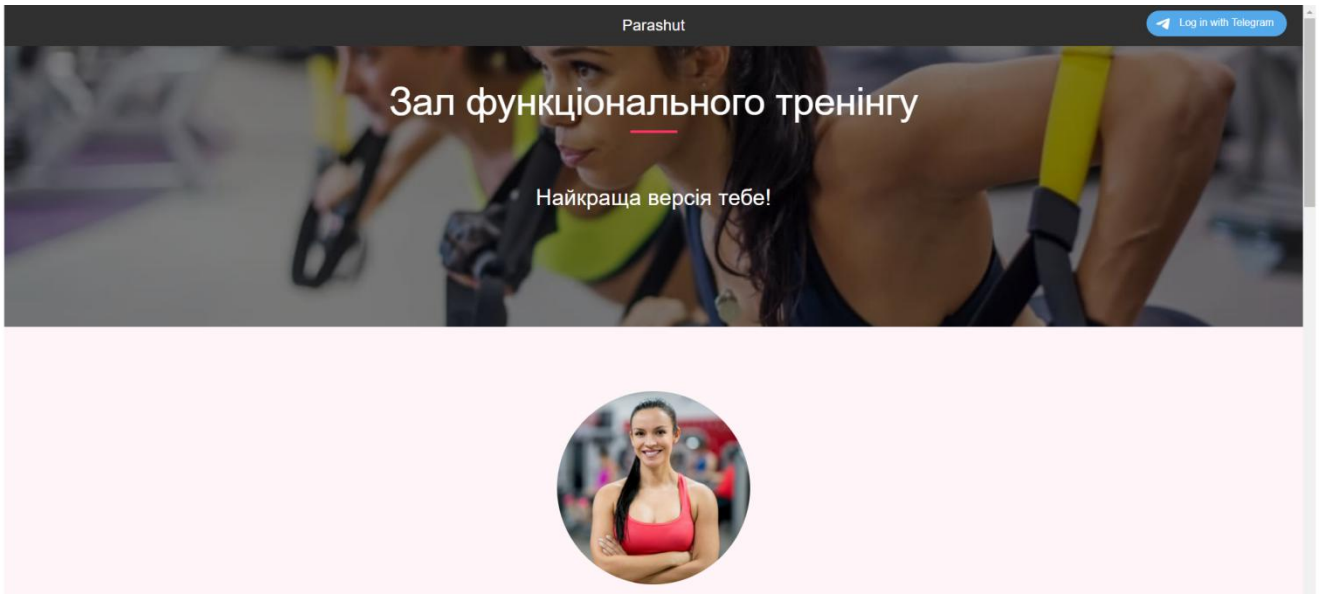

Рисунок 4.12 – Головна сторінка-візитка

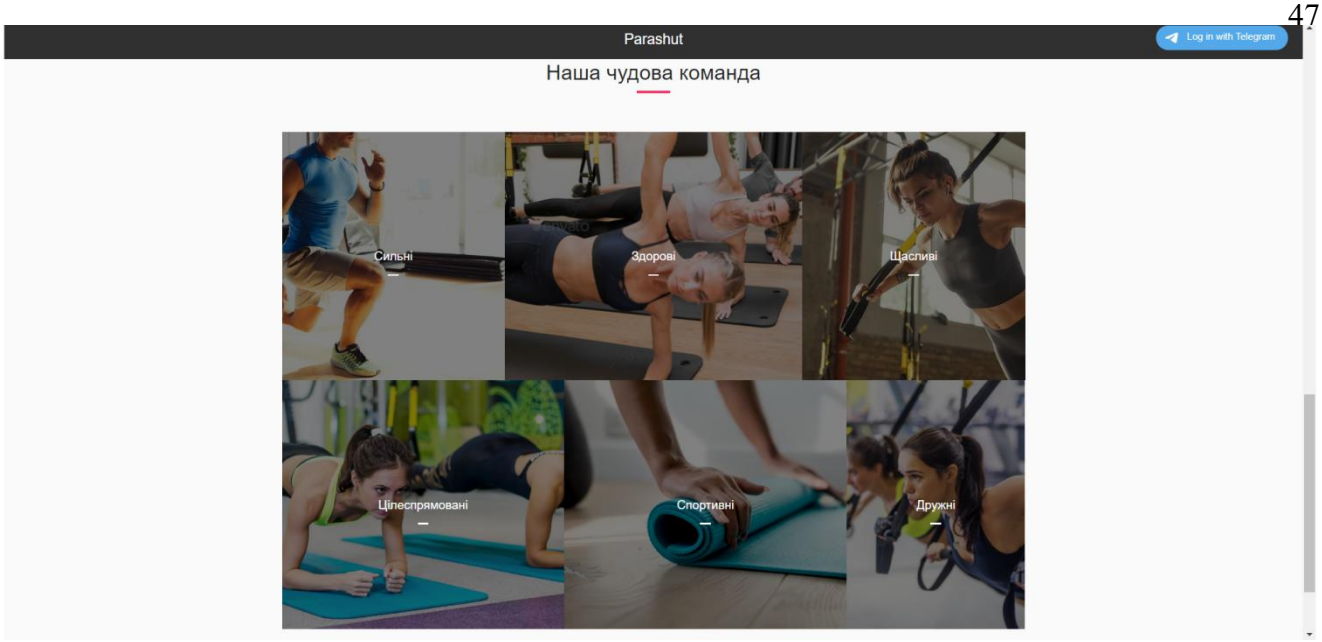

Рисунок 4.13 – Відображення вправ із мотиваційними написами.

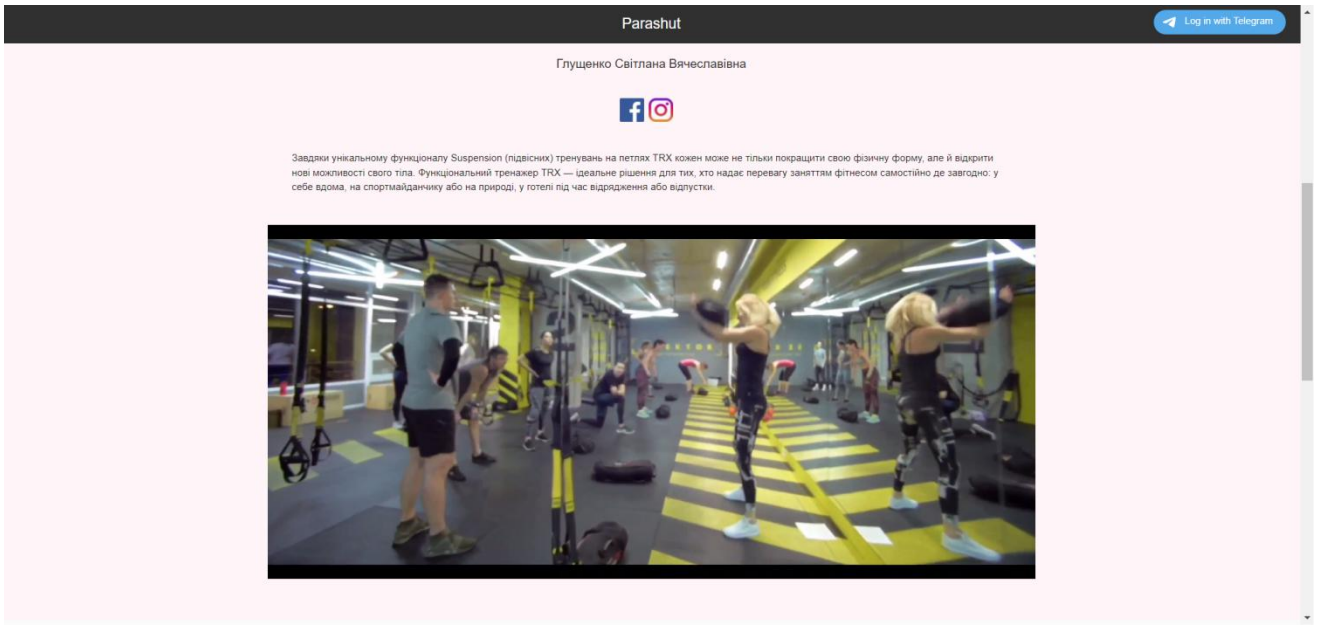

Рисунок 4.14 – Відеозапис демонстрації тренувань

Спілкування із Телеграм ботом для користувача відбувається у двох варіантах: голосування щодо відвідання тренувань у групі та індивідуальні проходження опитувань.

Для того, щоб відмітитися на занятті, потрібно просто проголосувати у опитуванні, як продемонстровано на рис. 4.15:

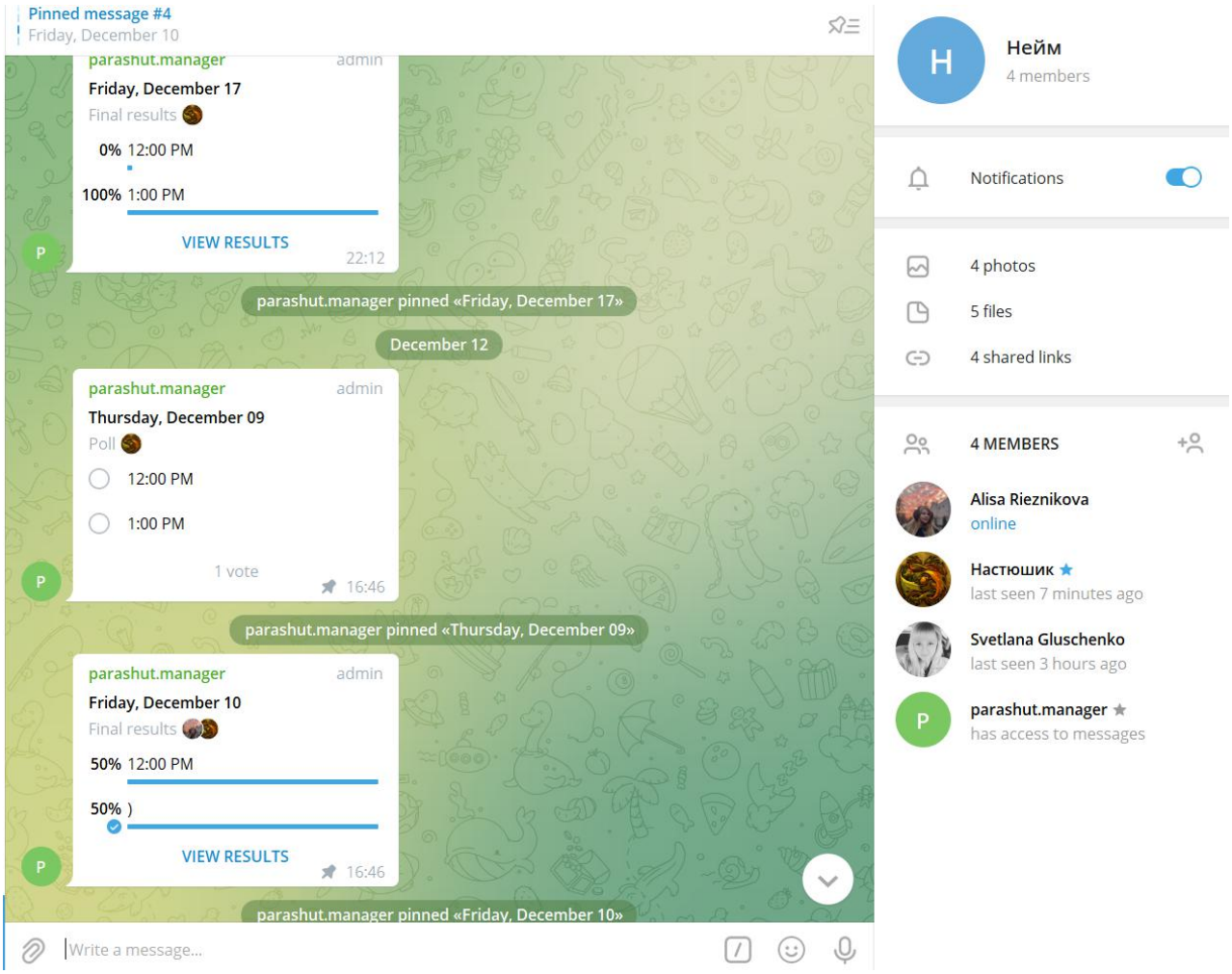

Рисунок 4.15 – Голосування для відмітки про відвідання

Для того, щоб пройти опитування для оновлення даних користувача, достатньо клацнути на бота та розпочати із ним чат. Після цього, користувач має відповісти на ряд запитань щоб оновити статистику, зазначеник на рис. 4.16.

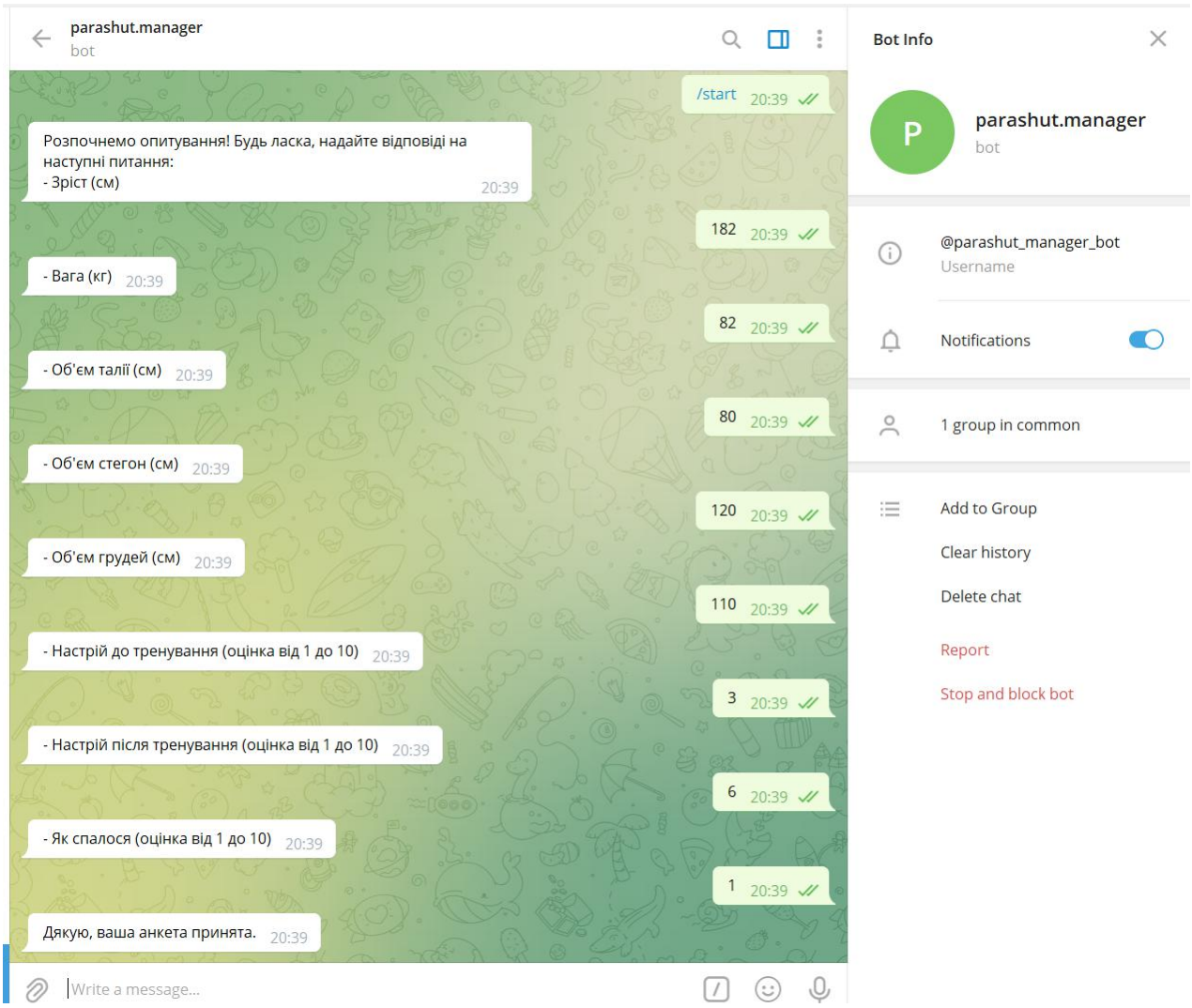

Рисунок 4.16 – Проходження опитування

Після авторизації за допомогою віджета Телеграм на сайті користувач потрапляє до особистого кабінету, інтерфейс якого продемонстровано на рис. 4.17.

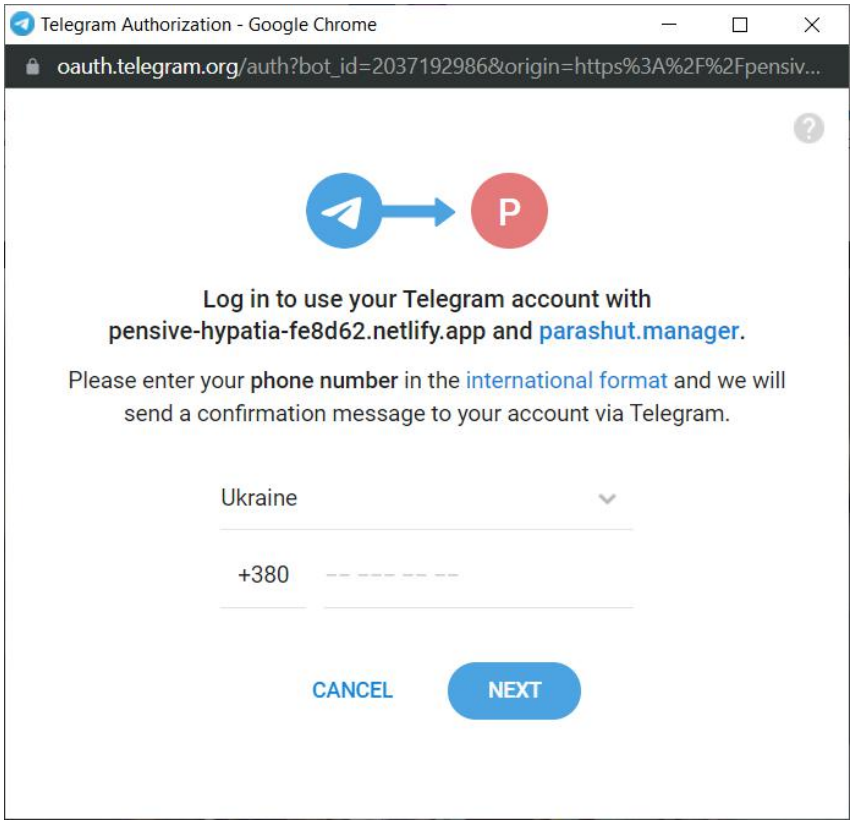

Рисунок 4.17 – Результат натискання на кнопку авторизації

Особистий кабінет користувача на рис. 4.18 дозволяє переглянути календар тренувань для своєї групи, а також чи відвідано було заняття.

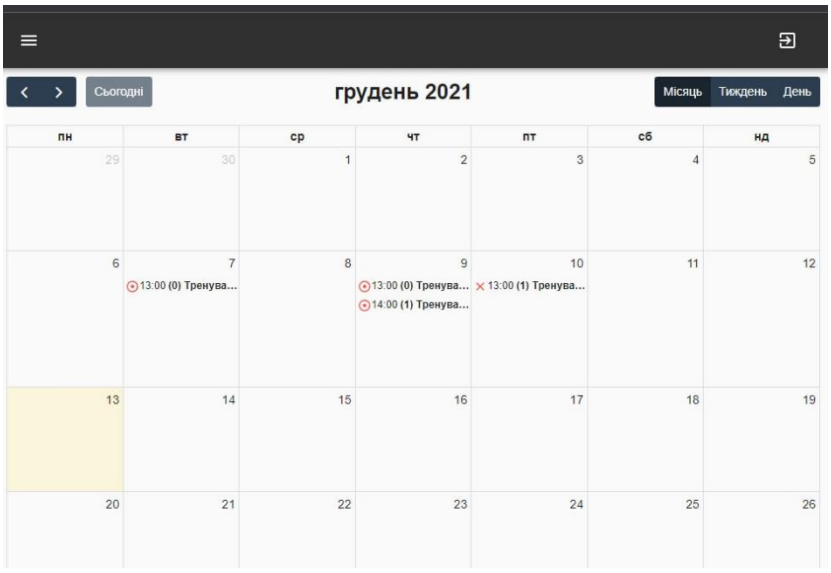

Рисунок 4.18 – Календар тренувань для користувача

Для перегляду статистики користувачеві виводяться таблиці із даними певного типу (вага, настрій, тощо) і відповідне відображення у вигляді графіків, що відображені на рисунках 4.19 та 4.20.

Рисунок 4.19 – Дані та графік динаміки ваги

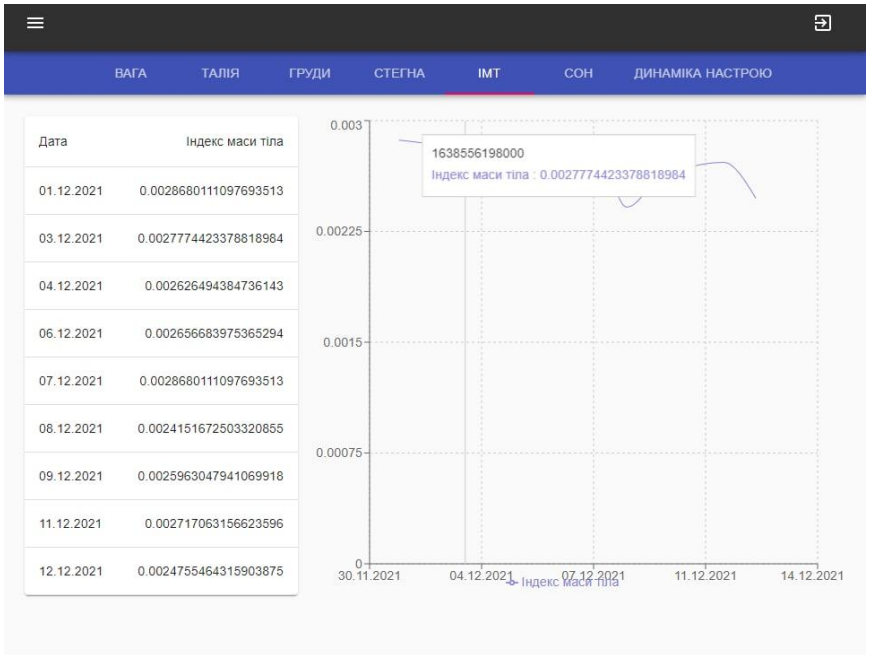

Рисунок 4.20 – Дані та графік динаміки індексу маси тіла

ВИСНОВКИ

Першим кроком у роботі над кваліфікаційною роботою магістра було досліджено доцільність розробки майбутнього додатку шляхом аналізу останніх досліджень у обраній сфері. Визначено, що статистика вказує на актуальність розроблюваної інформаційної підсистеми. Додатково було проведено огляд та оцінка існуючих альтернативних рішень, виходячи з яких було сформовано перелік вимог до продукту.

Після оцінки актуальності була сформульована мета дослідження та встановлені конкретні задачі. Було проведено моделювання майбутньої інформаційної підсистеми, що включає моделювання в IDEF0 та визначення варіантів використання. Додатково був сформований план робіт та проведено аналіз ризиків.

Реалізація інформаційної підсистеми включила в себе роботу над сайтом-візиткою, Телеграм-ботом та особистим кабінетом відвідувача тренувального залу “Парашут”. Сукупність розроблених елементів дозволяє частково автоматизувати роботу із клієнтами та значно спростити внутрішні процеси.

У результаті даний проект являє собою розробку, що дозволяє оцінити ефективність тренувань для користувача шляхом ведення регулярних оновлень даних, а також стає зручним інструментом для роботи із клієнтами. У майбутньому, даний проект буде отримувати оновлення для підтримки актуальності та внесення нового функціоналу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lack of exercise is a major cause of chronic diseases [Електронний ресурс] / Frank W. Booth, Christian K. Roberts, Matthew J. Laye. – 2014. – Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4241367/.
1. Mobile technology and the digitization of healthcare [Електронний ресурс] / Bhavnani SP, Narula J, Sengupta PP. – 2016. – Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4914890/.
2. The measure of a man [Електронний ресурс] / Savage N. – 2017. – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S0092867417306281.
3. Use of mobile applications to collect data in sport, health, and exercise science [Електронний ресурс] / Peart DJ, Shaw MP, Balsalobre-Fernández C. – 2018. – Режим доступу до ресурсу: https://journals.lww.com/nsca-jscr/Abstract/2019/04000/Use_of_Mobile_Applications_to_Collect_Data_in.29.aspx.fS_TRENDS_FOR_2019.6.aspx
4. A Systematic Review of Fitness Apps and Their Potential Clinical and Sports Utility for Objective and Remote Assessment of Cardiorespiratory Fitness [Електронний ресурс] / Adrià Muntaner-Mas, Carl J. Lavie, Robert Ross, Steven N. Blair, Ross Arena. – 2019. – Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6422959/#CR28.
5. Applied sport sciences. Sprint performance and mechanical outputs computed with an iPhone app: Comparison with existing reference methods [Електронний ресурс] / Natalia Romero-Franco, Fernando Capelo-Ramírez, Adrián Castaño-Zambudio. – 2017. – Режим доступу до ресурсу: http://repositorio.ucam.edu/bitstream/handle/10952/3156/romerofranco2016.pdf?sequence=1.

6. The validity and reliability of an iPhone app for measuring running mechanics [Електронний ресурс] / Balsalobre-Fernández C, Agopyan H, Morin J-B. – 2017. – Режим доступу до ресурсу: https://journals.humankinetics.com/view/journals/jab/33/3/article-p222.xml.
7. Psychological mechanisms underlying the relationship between commercial physical activity app use and physical activity engagement [Електронний ресурс] / Jasmine M. Petersen, Lucy K. Lewis, Eva Kemps. – 2020. – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S1469029219308568? via%3Dihub
8. Apps of steel: are exercise apps providing consumers with realistic expectations? A content analysis of exercise apps for presence of behavior change theory [Електронний ресурс] / Logan T Cowan, Brittany A Brown, P Cougar Hall, Sarah A Van Wagenen, Riley J Hedin / Health Education & Behavior 40. – 2013. – Режим доступу до ресурсу: https://www.researchgate.net/publication/230880117_Apps_of_Steel_Are_Exercise_Apps_Providing_Consumers_With_Realistic_Expectations_A_Content_Analysis_of_Exercise_Apps_for_Presence_of_Behavior_Change_Theory.
9. Efficacy of interventions that use apps to improve diet, physical activity and sedentary behaviour: a systematic review [Електронний ресурс] / Stephanie Schoeppe, Wendy Van Lippevelde, Stephanie Alley // w. International Journal of Behavioral Nutrition and Physical Activity 13. – 2016. – Режим доступу до ресурсу: https://www.academia.edu/35669048/Efficacy_of_interventions_that_use_apps_to_improve_diet_physical_activity_and_sedentary_behaviour_a_systematic_review?auto=download.
10. "Keep Going!": Understanding the Implications of Coaching through Fitness Apps to Support Physical Training [Електронний ресурс] / Raluca-Alexandra Stoica. – 2018. – Режим доступу до ресурсу: https://uclic.ucl.ac.uk/content/2-

study/4-current-taught-course/1-distinction-projects/12-18/stoica_ralucaalexandra_2018.pdf.

11. Evaluation of a personalized coaching system for physical activity: User appreciation and adherence [Електронний ресурс] / Julia S Mollee, Saskia J te Velde, Anouk Middelweerd. – 2017. – Режим доступу до ресурсу: https://www.researchgate.net/publication/322533523_Evaluation_of_a_personalized_coaching_system_for_physical_activity_user_appreciation_and_adherence.
12. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being [Електронний ресурс] / Richard M Ryan, Edward L Deci // American psychologist 55. – 2000. – Режим доступу до ресурсу: https://selfdeterminationtheory.org/SDT/documents/2000_RyanDeci_SDT.pdf.
13. E-learning SSU [Електронний ресурс] – Режим доступу до ресурсу: https://elearning.sumdu.edu.ua/
14. Developing it security metrics with goal question metric approach and smart criteria [Електронний ресурс] / Augustono Basuki. – 2017. – Режим доступу до ресурсу: https://www.researchgate.net/publication/321655537_DEVELOPING_IT_SECURITY_METRICS_WITH_GOAL_QUESTION_METRIC_APPROACH_AND_SMART_CRITERIA.
15. Work breakdown structure (WBS) [Електронний ресурс] / Mohaymen Alutbi. – 2020. – Режим доступу до ресурсу: https://www.researchgate.net/publication/342163727_WORK_BREAKDOWN_STRUCTURE_WBS.
16. Worldwide trends in insufficient physical activity from 2001 to 2016: a pooled analysis of 358 population-based surveys with 1·9 million participants [Електронний ресурс] / R.Guthold, G. Stevens, L. Riley, F. Bull // Lancet Glob Health. – 2018. – Режим доступу до ресурсу: https://www.thelancet.com/journals/langlo/article/PIIS2214-109X(18)30357-7/fulltext#%20.

17. Why software design is important? [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://hackernoon.com/why-software-design-is-important-9ecbea883bbb
18. Advantages And Disadvantages Of ER Model In DBMS [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://pctechnicalpro.blogspot.com/2017/04/advantages-disadvantages-er-model-dbms.html
19. Key benefits of using entity relationship diagrams [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://pcdreams.com.sg/key-benefits-of-using-entity-relationship-diagrams
20. Introduction to unified modeling language [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.uml.org/what-is-uml.htm
21. REST API | camunda BPM docs [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.camunda.org/manual/7.3/api-references/rest/
22. Telegram APIs [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://core.telegram.org/api
23. Process Definition [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.camunda.org/manual/7.3/api-references/rest/#process-definition
24. Process Instance [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.camunda.org/manual/7.3/api-references/rest/#process-instance
25. User [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://core.telegram.org/bots/api#user
26. User [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.camunda.org/manual/7.3/api-references/rest/#user
27. Telegram Messenger [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://telegram.org/

28. Workflow and Decision Automation Platform | Camunda [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://camunda.com/
29. Who guidelines on physical activity and sedentary behaviour [Електронний ресурс] – 2020. – Режим доступу до ресурсу: https://apps.who.int/iris/bitstream/handle/10665/337001/9789240014886-eng.pdf
30. Global Recommendations on Physical Activity for Health. Geneva: World Health Organization [Електронний ресурс] – 2010. Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/books/NBK305049/
31. Developing it security metrics with goal question metric approach and smart criteria [Електронний ресурс] / Augustono Basuki. – 2017. – Режим доступу до ресурсу: https://www.researchgate.net/publication/321655537_DEVELOPING_IT_SECURITY_METRICS_WITH_GOAL_QUESTION_METRIC_APPROACH_AND_SMART_CRITERIA.
32. Work breakdown structure (WBS) [Електронний ресурс] / Mohaymen Alutbi. – 2020. – Режим доступу до ресурсу: https://www.researchgate.net/publication/342163727_WORK_BREAKDOWN_STRUCTURE_WBS.

ДОДАТОК А
ПЛАНУВАННЯ РОБІТ

1 Ідентифікація мети ІТ-проекту

Мета розробки даного проекту визначена за допомогою використання методу SMART. Даний спосіб зумовлює визначення подальшої роботи за допомогою наступних показників:

- Specific – визначена ціль, яка є однозначною,
- Measurable – ціль, що вимірюється в кількісних показниках,
- Achievable – ціль, яка є реальною для втілення,
- Relevant – ціль, яка є актуальною для досягнення відносно реалій сьогодення,
- Time-framed – ціль, досягнення якої є обмежено у часі.

Деталізація мети проекту методом SMART наведена у табл. А.1.

Таблиця А.1 − Деталізація мети методом SMART

Specific - конкретність	Розробити інформаційну підсистему підтримки роботи з відвідувачами спортивного залу.
Measurable - вимірюваність	Розробити як частину системи Телеграм-бота, та модулі для підтримки роботи користувачів, в тому числі для контролю стану здоров’я, відвідуваннями занять, керування тренуваннями.
Achievable - досяжність	Розробка проводиться на C# та Typescript на фреймворка React, .NET Core, із якими команда розробки має навички роботи, а база даних спроектована і втілена за допомогою MySQL.

Продовження таблиці А.1

Relevant - актуальність	Розробити інформаційну підсистему що відповідає індивідуальним вимогам замовника, а дизайн - стилю бренду.
Time-framed - обмеження у часі	Втілити проект згідно встановленого календарного плану.

А.2 Планування робіт проекту (WBS)

Виконання структурної декомпозиції робіт є важливою частиною проектування процесу розробки програмного продукту. У даному випадку на вершині ієрархії стоїть задача реалізації проекту, а саме розробка інформаційної підсистеми. Цей рівень декомпозується на чотири рівні: встановлення вимог, проектування та дизайн, реалізація, документуваня.

Описання рівнів декомпозиції:

1. Ініціалізація - зібрання вимог, аналіз актуальності проекту та аналогів, специфікація вимог, деталізація мети проекту.
2. Планування включає в себе розробку WBS- та OBS-структури проекту, визачення ризиків та плану робіт.
3. Реалізація продукту проекту – розробка Телеграм-боту, створення власного кабінету користувача, календаря відвідувань.
4. Проведення тестування продуктута попередня розробка тест-кейсів.
5. Етап завершення включає презентацію проекту.

WBS-структуру проекту наведенано рисунку А.1.

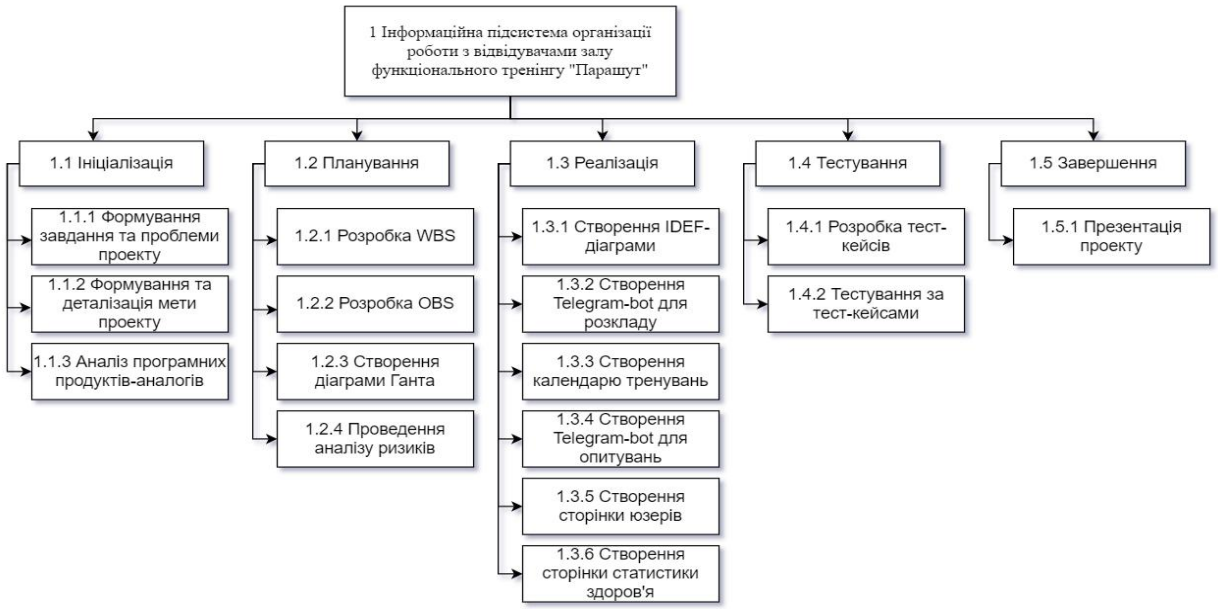

Рисунок А.1 – WBS-структура проекту

А.3 Планування структури організації (OBS)

Наступним етапом є створення організаційної структури проекту, яка відображає учасників проекту та осіб, які відповідальні за реалізацію проекту. На кожному із рівнів відтворено розбиття на команди, які стануть відповідальними за виконання окремих робіт.

OBS структура наведена на рисунку А.2.

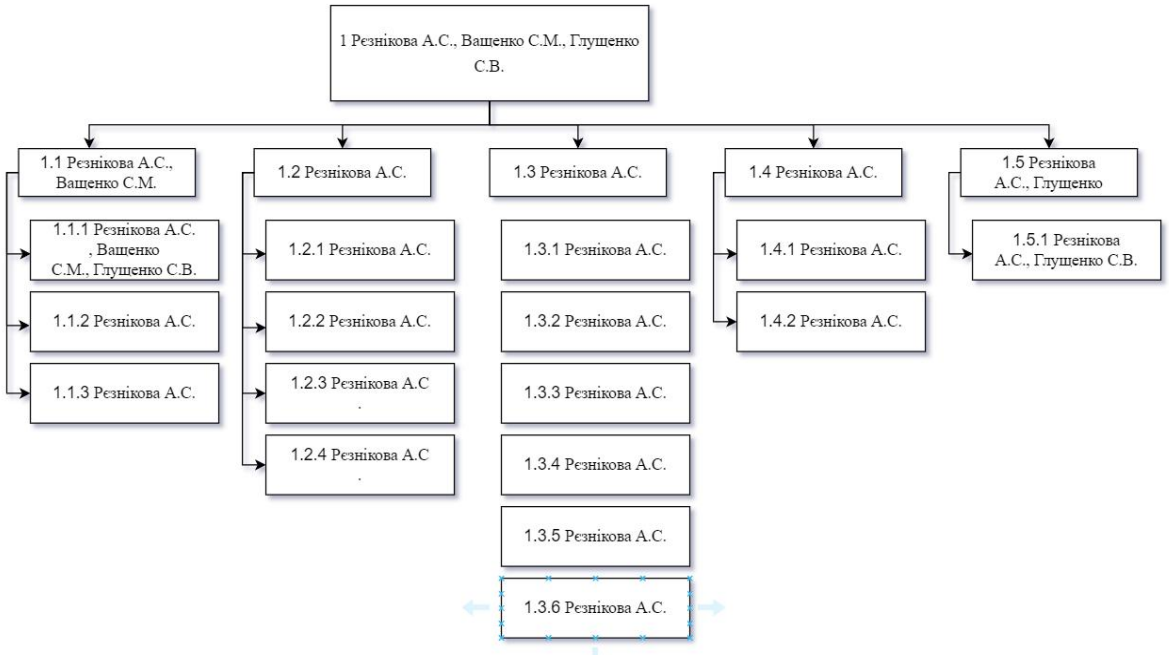

Рисунок А.2 – OBS-структура проекту

У таблиці А.2 представлено перелік стейкхолдерів - осіб, зацікавлених у втіленні проекту.

Таблиця А.2 - Представлення зацікавлених сторін

	Аліса Рєзнікова	Світлана Ващенко	Світлана Глущенко
Контактна інформація	@a.rznkva	s.vashchenko@cs.sumdu.edu.ua	@s.glooshchenko
Внесок *(1-5)*	5	3	3
Вплив *(1-5)*	5	3	4
Що важливо для сторони?	Залученість усіх стейкхолдерів та вчасні відгуки; Коректний тайм-менеджмент; Достатній рівень володіння технологіями розробки та аналізу;	Своєчасний відгук зі сторони команди розробки; висока продуктивність.	Доставка якісного продукту, що відповідає вимогам та дедлайну.

Продовження табл. А.2

Як зацікавлені сторони можуть внести свій вклад в проект?	Грамотна оцінка можливості втілення частин проекту, продуктивна робота та грамотний тайм-менеджмент.	Своєчасна координація роботи розробників; Надання консультацій команді за необхідністю.	Складення чіткого списку вимог; Своєчасне надання відгуку про кожний мінімальний продукт спринту.
Як зацікавлена особа може заблокувати проект?	Вимога додаткового часу для вивчення нових технологій; Недотримання термінів; Брак спілкування з іншими зацікавленими сторонами.	Несвоєчасний фідбек за проміжними результатами роботи команди.	Викладення незрозумілих або суперечливих вимог до продукту; Змінадедлайнів роботи; Закриття проекту.
Роль у проекті	Розробники	Скрам майстер	Власник

А.3 Побудова календарного графіку виконання ІТ-проекту.

Для встановлення термінів виконання робіт та чіткої візуалізації послідовності їх виконання було побудовано діаграму Ганта. Горизонтальна лінійка у графічному представленні являє собою визначений таймлайн у відповідних одиницях - дні, тижні, роки (у залежності від тривалості проекту). Зліва (як представлено на рисунку А.4) визначають перелік робіт, розбивають їх на підкатегорії та формують зв’язки, що зумовлюють послідовність виконання.

Діаграма Ганта представлена на рисунках А.3, А.4.

Рисунок А.3 – діаграма Ганта

	Проект "Парашут"	**66 days**	**Mon 29.11.21**	**Mon 28.02.22**	
	Ініціалізація	**5 days**	**Mon 18.10.21**	**Fri 22.10.21**	
	Формування завда	1 day	Mon 18.10.21	Mon 18.10.21	
	Формування та де	2 days	Tue 19.10.21	Wed 20.10.21	3
	Аналіз програмних	2 days	Thu 21.10.21	Fri 22.10.21	4
	Планування	**8 days**	**Mon 25.10.21**	**Wed 03.11.21**	**2**
	Розробка WBS	2 days	Mon 25.10.21	Tue 26.10.21	5
	Розробка OBS	1 day	Wed 27.10.21	Wed 27.10.21	7
	Створення діаграми Ганта	2 days	Thu 28.10.21	Fri 29.10.21	8
	Проведення аналізу ризиків	3 days	Mon 01.11.21	Wed 03.11.21	9
	Реалізація	**27 days**	**Thu 04.11.21**	**Sun 12.12.21**	**6**
	Створення IDEF-діаграми	3 days	Thu 04.11.21	Sun 07.11.21	10
	Створення Telegram-bot для	4 days	Mon 08.11.21	Thu 11.11.21	12
	Створення календ	7 days	Fri 12.11.21	Mon 22.11.21	13
	Створення Telegra	6 days	Tue 23.11.21	Tue 30.11.21	14
	Створення сторінки юзерів	4 days	Wed 01.12.21	Mon 06.12.21	15
	Створення сторінк	5 days	Tue 07.12.21	Sun 12.12.21	16
	Тестування	**7 days**	**Mon 13.12.21**	**Tue 21.12.21**	**11**
	Розробка тест-кейсів	4 days	Mon 13.12.21	Thu 16.12.21	17
	Тестування за тест-кейсами	1 day	Tue 21.12.21	Tue 21.12.21	19
	Завершення	**1 day**	**Wed 22.12.21**	**Wed 22.12.21**	**18**
	Презентація проек	1 day	Wed 22.12.21	Wed 22.12.21	20

Рисунок А.4 – Графік робіт за діаграмою

А.4 Управління ризиками.

Ризиками називаються події, що виникають або можуть виникнути із певною ймовірністю, які можуть вплинути на проект позитивно чи негативно.

Для зниження рівня виникнення ризиків важливо вивчити та застосувати одну із методологій управління. Відстеження ризиків забезпечує учасників проекту можливістю приймати дієві рішення до настання потенційно впливових подій.

Обраний шлях роботи із ризиками включає наступні етапи: ідентифікація, оцінювання ризиків, заходи реагування та моніторинг. До переліку вибраних способів управління ризиками було віднесено страхування і скасування, запобігання та контролювання та поглинання ризиків.

У роботі над даним проектом була використана стратегія пом’якшення та прийняття. Було визначено ризики, яким неможливо запобігти і ті, можливо визначити, передбачити та запобігти їм. У наступній таблиці А.3 наведено виявлені можливі ризики, оцінена ймовірність їх виникнення, а також наведено план по їх оминанню якщо можливо.

Таблиця А.3 – Ризики проекту

№	**Опис ризику**	**Вплив**	**Ймовірність**	**Рівень небезпеки**	**Стратегія**
1	Встановлення нечітких вимог	4	1		Налагодження ефективної комунікації із замовником.
2	Недостатній рівень обізнанності та практичних навичок.	4	2		Грамотне розподілення часу із виділенням періоду підготовки
3	Можливість виникнення конфліктів сторін.	2	2		Вчасне обговорення та урегулювання конфліктів із залученням скрам-майстра.
4	Неоптимальний розподіл часу роботи	4	4		Розробка чіткого плану та його дотримання

Продовження табл. А.3

5	Низька ретельність тестування.	3	3		Складання чітких та всеохоплюючих тест-кейсів
6	Непередбачені обставини (хвороба, політичне становище, т.д.)	3	1		Виділення додаткового часу на початку планування і можливість заморозити проект.

Таблиця А.3 – Ризики проекту (зведення)

5					
4	1	2		4	
3	6		5		
2		3			
P/I	1	2	3	4	5

ДОДАТОК Б
ЛІСТИНГ КОДУ

AppBar.tsx

```
import React from 'react'
import { Theme, WithStyles, withStyles } from '@material-ui/core/styles'
import Link from '@material-ui/core/Link'
import Toolbar from '@material-ui/core/Toolbar'
import MuiAppBar from '@material-ui/core/AppBar'
import Button from '@material-ui/core/Button'
import TelegramLoginButton from 'react-telegram-login'
import autobind from 'autobind-decorator'
import { login } from '../../../store/currentUser/currentUserActions'
import { bindActionCreators, Dispatch } from 'redux'
import { connect } from 'react-redux'

const styles = (theme: Theme) => ({
  title: {
    fontSize: 24,
  },
  toolbar: {
    justifyContent: 'space-between',
    height: 64,
    backgroundColor: '#303030',
  },
  left: {
    flex: 1,
  },
  right: {
    flex: 1,
    display: 'flex',
    justifyContent: 'flex-end',
  },
})

export interface AppBarBaseProps {
  login: typeof login
}

type AppBarProps = WithStyles<typeof styles> & AppBarBaseProps

const mapDispatchToProps = (dispatch: Dispatch) => {
  return {
    login: bindActionCreators(login, dispatch),
  }
}

const connectToState = (component: React.ComponentType<any>) =>
  connect(() => {}, mapDispatchToProps)(component)

export class AppBarBase extends React.Component<AppBarProps> {
  // @autobind
  // handleSubmit(e: any): void {
  //   // USED ONLY FOR DEVELOPMENT
  //   this.props.login(344923873, '')
  // }
```

```
  @autobind
  handleTelegramResponse(response: any): void {
    this.props.login(response.id, response.hash)
  }
  render() {
    const { classes } = this.props

    return (
      <div>
        <MuiAppBar elevation={0} position="fixed">
          <Toolbar className={classes.toolbar}>
            <div className={classes.left} />
            <Link
              variant="h6"
              underline="none"
              color="inherit"
              className={classes.title}
              href="/"
            >
              {'Parashut'}
            </Link>
            <div className={classes.right}>
              {/* <Button
                type="submit"
                variant="contained"
                color="primary"
                onClick={this.handleSubmit}
              >
                Sign in
              </Button> */}
              <TelegramLoginButton
                dataOnauth={this.handleTelegramResponse}
                botName="parashut_manager_bot"
              />
            </div>
          </Toolbar>
        </MuiAppBar>
      </div>
    )
  }
}

export default withStyles(styles)(connectToState(AppBarBase))
```

AppFooter.tsx

```
import React from 'react'
import {
  createStyles,
  Theme,
  withStyles,
  WithStyles,
} from '@material-ui/core/styles'
import Grid from '@material-ui/core/Grid'
import Container from '@material-ui/core/Container'
import Typography from './Typography'

const styles = (theme: Theme) =>
  createStyles({
    container: {
```

```
      marginTop: theme.spacing(6),
      marginBottom: theme.spacing(2),
    },
    iconsWrapper: {
      height: 120,
    },
    icons: {
      display: 'flex',
    },
    icon: {
      width: 40,
      height: 40,
      display: 'flex',
      justifyContent: 'center',
      alignItems: 'center',
      marginRight: theme.spacing(1),
    },
    list: {
      margin: 0,
      listStyle: 'none',
      padding: 0,
    },
    listItem: {
      paddingTop: theme.spacing(0.5),
      paddingBottom: theme.spacing(0.5),
    },
  })

export class AppFooterBase extends React.Component<WithStyles<typeof styles>> {
  render() {
    const { classes } = this.props

    return (
      <Container className={classes.container}>
        <Grid container justifyContent="center">
          <Grid container item xs={12} sm={12} md={12}>
            <Typography variant="h6" marked="left">
              Зв'язок з нами
            </Typography>
            <Grid
              container
              direction="column"
              justify="center"
              justifyContent="center"
              className={classes.iconsWrapper}
              spacing={2}
            >
              <Grid item className={classes.icons}>
                <a href="https://www.facebook.com" className={classes.icon}>
                  <img
                    src="/static/facebook-icon.png"
                    alt="Facebook"
                    className={classes.icon}
                  />
                </a>
                <a href="https://www.instagram.com" className={classes.icon}>
                  <img
                    src="/static/instagram-icon.png"
                    alt="Instagram"
                    className={classes.icon}
                  />
                </a>
```

```
              </Grid>
              <Grid item>
                <>
                  {'© Parashut '}
                  {new Date().getFullYear()}
                </>
              </Grid>
            </Grid>
          </Grid>
        </Grid>
      </Container>
    )
  }
}

export default withStyles(styles)(AppFooterBase)
```

Banner.tsx

```
import React from 'react'
import {
  createStyles,
  Theme,
  WithStyles,
  withStyles,
} from '@material-ui/core/styles'
import Container from '@material-ui/core/Container'
import Typography from './Typography'

const styles = (theme: Theme) =>
  createStyles({
    root: {
      color: theme.palette.common.white,
      position: 'relative',
      display: 'flex',
      alignItems: 'center',
      [theme.breakpoints.up('sm')]: {
        height: '80vh',
        minHeight: 300,
        maxHeight: 500,
      },
    },
    container: {
      marginTop: theme.spacing(10),
      marginBottom: theme.spacing(14),
      display: 'flex',
      flexDirection: 'column',
      alignItems: 'center',
    },
    backdrop: {
      position: 'absolute',
      left: 0,
      right: 0,
      top: 0,
      bottom: 0,
      backgroundColor: theme.palette.common.black,
      opacity: 0.5,
      zIndex: -1,
    },
    background: {
```

```
      position: 'absolute',
      left: 0,
      right: 0,
      top: 0,
      bottom: 0,
      backgroundSize: 'cover',
      backgroundRepeat: 'no-repeat',
      zIndex: -2,
      backgroundImage: `url(static/banner.jpg)`,
      backgroundPosition: 'center',
    },
    h5: {
      marginBottom: theme.spacing(4),
      marginTop: theme.spacing(4),
      [theme.breakpoints.up('sm')]: {
        marginTop: theme.spacing(10),
      },
      color: 'inherit',
    },
    more: {
      marginTop: theme.spacing(2),
      color: 'inherit',
    },
  })

export class BannerBase extends React.Component<WithStyles<typeof styles>> {
  render() {
    const { classes } = this.props

    return (
      <section className={classes.root}>
        <Container className={classes.container}>
          <Typography variant="h2" marked="center">
            Зал функціонального тренінгу
          </Typography>
          <Typography variant="h4" className={classes.h5}>
            Найкраща версія тебе!
          </Typography>
          <div className={classes.backdrop} />
          <div className={classes.background} />
        </Container>
      </section>
    )
  }
}

export default withStyles(styles)(BannerBase)
```

Pictures.tsx

```
import React from 'react'
import {
  Theme,
  withStyles,
  createStyles,
  WithStyles,
} from '@material-ui/core/styles'
import Container from '@material-ui/core/Container'
import Typography from './Typography'
```

```
import { Grid } from '@material-ui/core'

const styles = (theme: Theme) =>
  createStyles({
    root: {
      marginTop: theme.spacing(8),
      marginBottom: theme.spacing(4),
    },
    images: {
      marginTop: theme.spacing(8),
      display: 'flex',
      flexWrap: 'wrap',
    },
    imageWrapper: {
      position: 'relative',
      display: 'block',
      padding: 0,
      borderRadius: 0,
      height: '40vh',
      [theme.breakpoints.down('sm')]: {
        width: '100% !important',
        height: 100,
      },
      '&:hover': {
        zIndex: 1,
      },
      '&:hover $imageBackdrop': {
        opacity: 0.15,
      },
      '&:hover $imageMarked': {
        opacity: 0,
      },
      '&:hover $imageTitle': {
        border: '4px solid currentColor',
      },
    },
    imageButton: {
      position: 'absolute',
      left: 0,
      right: 0,
      top: 0,
      bottom: 0,
      display: 'flex',
      alignItems: 'center',
      justifyContent: 'center',
      color: theme.palette.common.white,
    },
    imageSrc: {
      position: 'absolute',
      left: 0,
      right: 0,
      top: 0,
      bottom: 0,
      backgroundSize: 'cover',
      backgroundPosition: 'center 40%',
    },
    imageBackdrop: {
      position: 'absolute',
      left: 0,
      right: 0,
      top: 0,
      bottom: 0,
```

```
      background: theme.palette.common.black,
      opacity: 0.5,
      transition: theme.transitions.create('opacity'),
    },
    imageTitle: {
      position: 'relative',
      padding: `${theme.spacing(2)}px ${theme.spacing(4)}px 14px`,
      color: 'inherit',
    },
    imageMarked: {
      height: 3,
      width: 18,
      background: theme.palette.common.white,
      position: 'absolute',
      bottom: -2,
      left: 'calc(50% - 9px)',
      transition: theme.transitions.create('opacity'),
    },
    title: {
      textAlign: 'center',
    },
  })

export class PicturesBase extends React.Component<WithStyles<typeof styles>> {
  render() {
    const { classes } = this.props

    const images = [
      {
        url: '/static/image_2021-12-12_18-43-06.png',
        title: 'Сильні',
        width: '30%',
      },
      {
        url: 'static/image_2021-12-12_18-43-36.png',
        title: 'Здорові',
        width: '40%',
      },
      {
        url: 'static/image_2021-12-12_18-44-05.png',
        title: 'Щасливі',
        width: '30%',
      },
      {
        url: 'static/image_2021-12-12_18-44-39.png',
        title: 'Цілеспрямовані',
        width: '38%',
      },
      {
        url: 'https://images.unsplash.com/photo-1533727937480-da3a97967e95?auto=format&fit=crop&w=400&q=80',
        title: 'Спортивні',
        width: '38%',
      },
      {
        url: 'static/photo_2021-12-13_02-15-13.jpg',
        title: 'Дружні',
        width: '24%',
      },
    ]

    return (
      <Container className={classes.root} component="section">
```

```
        <Typography variant="h4" marked="center" className={classes.title}>
          Наша чудова команда
        </Typography>
        <div className={classes.images}>
          {images.map((image) => (
            <Grid
              key={image.title}
              className={classes.imageWrapper}
              style={{
                width: image.width,
              }}
            >
              <div
                className={classes.imageSrc}
                style={{
                  backgroundImage: `url(${image.url})`,
                }}
              />
              <div className={classes.imageBackdrop} />
              <div className={classes.imageButton}>
                <Typography variant="h6" className={classes.imageTitle}>
                  {image.title}
                  <div className={classes.imageMarked} />
                </Typography>
              </div>
            </Grid>
          ))}
        </div>
      </Container>
    )
  }
}

export default withStyles(styles)(PicturesBase)
```

ProductValues.tsx

```
import React from 'react'
import {
  createStyles,
  Theme,
  WithStyles,
  withStyles,
} from '@material-ui/core/styles'
import Grid from '@material-ui/core/Grid'
import Container from '@material-ui/core/Container'
import Typography from './Typography'
import { Avatar } from '@material-ui/core'

const styles = (theme: Theme) =>
  createStyles({
    root: {
      display: 'flex',
      overflow: 'hidden',
      backgroundColor: '#fff5f8',
    },
    container: {
      marginTop: theme.spacing(15),
      marginBottom: theme.spacing(10),
      display: 'flex',
```

```
      position: 'relative',
    },
    item: {
      display: 'flex',
      flexDirection: 'column',
      alignItems: 'center',
      padding: theme.spacing(0, 5),
    },
    image: {
      height: 55,
    },
    title: {
      marginTop: theme.spacing(5),
      marginBottom: theme.spacing(5),
    },
    avatar: {
      width: 300,
      height: 300,
    },
    video: {
      maxWidth: '100%',
      marginTop: '50px',
    },
    icons: {
      marginBottom: theme.spacing(6),
      display: 'flex',
    },
    icon: {
      width: 40,
      height: 40,
      display: 'flex',
      justifyContent: 'center',
      alignItems: 'center',
      marginRight: theme.spacing(1),
    },
  })

export class DescriptionBase extends React.Component<
  WithStyles<typeof styles>
> {
  render() {
    const { classes } = this.props

    return (
      <div className={classes.root}>
        <Container className={classes.container}>
          <Grid container spacing={5}>
            <Grid>
              <div className={classes.item}>
                <Avatar
                  alt="Trainer"
                  src="/static/coach.png"
                  className={classes.avatar}
                />
                <Typography variant="h6" className={classes.title}>
                  Глущенко Світлана Вячеславівна
                </Typography>
                <Grid item className={classes.icons}>
                  <a href="https://www.facebook.com" className={classes.icon}>
                    <img
                      src="/static/facebook-icon.png"
                      alt="Facebook"
```

```
                    className={classes.icon}
                  />
                </a>
                <a href="https://www.instagram.com" className={classes.icon}>
                  <img
                    src="/static/instagram-icon.png"
                    alt="Instagram"
                    className={classes.icon}
                  />
                </a>
              </Grid>
              <Typography variant="body1">
                {
                  'Завдяки унікальному функціоналу Suspension (підвісних) тренувань на петлях TRX кожен може н
е тільки покращити свою фізичну форму, але й відкрити нові можливості свого тіла. Функціональний тренажер TRX —
 ідеальне рішення для тих, хто надає перевагу заняттям фітнесом самостійно де завгодно: у себе вдома, на спортма
йданчику або на природі, у готелі під час відрядження або відпустки.'
                }
              </Typography>
            </div>
          </Grid>

          <Grid>
            <video className={classes.video} autoPlay loop muted>
              <source src="/static/video2.mp4" type="video/mp4" />
              Your browser does not support the video tag.
            </video>
          </Grid>
        </Grid>
      </Container>
    </div>
  )
 }
}

export default withStyles(styles)(DescriptionBase)
```

Typography.tsx

```
import React from 'react'
import {
  createStyles,
  Theme,
  WithStyles,
  withStyles,
} from '@material-ui/core/styles'
import { capitalize } from '@material-ui/core/utils'
import MuiTypography from '@material-ui/core/Typography'
import { Variant } from '@material-ui/core/styles/createTypography'

const styles = (theme: Theme) =>
  createStyles({
    markedH2Center: {
      height: 4,
      width: 73,
      display: 'block',
      margin: `${theme.spacing(1)}px auto 0`,
      backgroundColor: '#ff3366',
    },
    markedH3Center: {
      height: 4,
```

```
      width: 55,
      display: 'block',
      margin: `${theme.spacing(1)}px auto 0`,
      backgroundColor: '#ff3366',
    },
    markedH4Center: {
      height: 4,
      width: 55,
      display: 'block',
      margin: `${theme.spacing(1)}px auto 0`,
      backgroundColor: '#ff3366',
    },
    markedH6Left: {
      height: 2,
      width: 28,
      display: 'block',
      marginTop: theme.spacing(0.5),
      background: 'currentColor',
    },
  })

const variantMapping = {
  h1: 'h1',
  h2: 'h1',
  h3: 'h1',
  h4: 'h1',
  h5: 'h3',
  h6: 'h2',
  subtitle1: 'h3',
}

export interface TypographyBaseProps {
  children?: any
  classes?: any
  marked?: string
  variant: 'inherit' | Variant
  className?: string
}

export type TypographyProps = TypographyBaseProps & WithStyles<typeof styles>

export class Typography extends React.Component<TypographyProps> {
  render() {
    const { children, classes, marked = false, variant, ...other } = this.props

    return (
      <MuiTypography
        variantMapping={variantMapping}
        variant={variant}
        {...other}
      >
        {children}
        {marked ? (
          <span
            className={
              classes[`marked${capitalize(variant) + capitalize(marked)}`]
            }
          />
        ) : null}
      </MuiTypography>
    )
  }
```

```
}

export default withStyles(styles)(Typography)
```

LandingPage.connect.ts

```
import * as React from 'react'
import { connect } from 'react-redux'
import { Shim } from '../../store/shim'
import currentUserSelectors from '../../store/currentUser/currentUserSelectors'

const mapStateToProps = (state: Shim) => {
  return {
    currentUser: currentUserSelectors.currentUserState(state),
  }
}

export const connectToState = (component: React.ComponentType<any>) =>
  connect(mapStateToProps, () => {})(component)
```

LandingPage.tsx

```
import React from 'react'
import { LandingPageProps } from './LandingPage.types'
import { CssBaseline } from '@material-ui/core'
import { withRouter, Redirect } from 'react-router-dom'
import { compose } from 'redux'
import { connectToState } from './LandingPage.connect'
import { RouteConstants } from '../../RouteConstants'
import AppBar from './components/AppBar'
import AppFooter from './components/AppFooter'
import Pictures from './components/Pictures'
import Description from './components/ProductValues'
import BannerBase from './components/Banner'

export class LandingPageBase extends React.Component<LandingPageProps> {
  render() {
    if (this.props.currentUser.id) {
      return <Redirect to={RouteConstants.home} />
    }

    return (
      <>
        <CssBaseline />
        <AppBar />
        <BannerBase />
        <Description />
        <Pictures />
        <AppFooter />
      </>
    )
  }
}

export default compose(
  withRouter,
  connectToState,
)(LandingPageBase) as React.ComponentType<LandingPageProps>
```

LandingPage.types.ts

```
import { RouteComponentProps } from 'react-router-dom'
import { IUser } from '../../entities/IUser'

export interface LandingPageBaseProps {
  currentUser: IUser
}

export type LandingPageProps = LandingPageBaseProps & RouteComponentProps
```

UserStatsPage.connect.ts

```
import * as React from 'react'
import { connect } from 'react-redux'
import { bindActionCreators, Dispatch } from 'redux'
import { login } from '../../store/currentUser/currentUserActions'
import { Shim } from '../../store/shim'
import currentUserSelectors from '../../store/currentUser/currentUserSelectors'
import userStatsSelectors from '../../store/userStats/userStatsSelectors'

const mapStateToProps = (state: Shim) => {
  return {
    currentUser: currentUserSelectors.currentUserState(state),
    userStats: userStatsSelectors.getUserStatsList(state),
  }
}

const mapDispatchToProps = (dispatch: Dispatch) => {
  return {
    login: bindActionCreators(login, dispatch),
  }
}

export const connectToState = (component: React.ComponentType<any>) =>
  connect(mapStateToProps, mapDispatchToProps)(component)
```

UserStatsPage.styles.ts

```
import { createStyles, Theme } from '@material-ui/core'

export const styles = (theme: Theme) =>
  createStyles({
    root: {
      '& .MuiTableContainer-root': {
        maxHeight: 'calc(100vh - 146px)',
      },
    },
    statsTab: {
      minWidth: '100px',
    },
  })
```

UserStatsPage.tsx

```
import React from 'react'
import {
  UserStatsPageProps,
  UserStatsPageStateProps,
} from './UserStatsPage.types'
import { Grid, Tab, Tabs, withStyles } from '@material-ui/core'
import { styles } from './UserStatsPage.styles'
import { withRouter } from 'react-router-dom'
import { compose } from 'redux'
import { connectToState } from './UserStatsPage.connect'
import Spinner from '../../components/Spinner/Spinner'
import Wrapper from '../../components/Wrapper/Wrapper'
import autobind from 'autobind-decorator'
import DataGraph from '../../components/DataGraph/DataGraph'
import DataTable from '../../components/DataTable/DataTable'
import TabPanel, { a11yProps } from '../../components/TabPanel/TabPanel'
import AppBar from '@material-ui/core/AppBar'

export class UserStatsPageBase extends React.Component<
  UserStatsPageProps,
  UserStatsPageStateProps
> {
  constructor(props: UserStatsPageProps) {
    super(props)
    this.state = {
      activeTab: 0,
    }
  }
  render() {
    const { isFetching } = this.props
    const content = isFetching ? <Spinner /> : this.renderContent()

    return <Wrapper contentComponent={content} />
  }

  private renderContent() {
    const { classes } = this.props
    const { activeTab } = this.state
    return (
      <div className={classes.root}>
        <AppBar position="static">
          <Tabs
            value={activeTab}
            onChange={this.handleChange}
            aria-label="simple tabs example"
            centered
          >
            <Tab className={classes.statsTab} label="Вага" {...a11yProps(0)} />
            <Tab className={classes.statsTab} label="Талія" {...a11yProps(1)} />
            <Tab className={classes.statsTab} label="Груди" {...a11yProps(2)} />
            <Tab
              className={classes.statsTab}
              label="Стегна"
```

```
          {...a11yProps(3)}
        />
        <Tab className={classes.statsTab} label="IMT" {...a11yProps(4)} />
        <Tab className={classes.statsTab} label="Сон" {...a11yProps(5)} />
        <Tab
          className={classes.statsTab}
          label="Динаміка настрою"
          {...a11yProps(6)}
        />
      </Tabs>
    </AppBar>
    <TabPanel value={activeTab} index={0}>
      {this.renderUserStatTab(
        this.getSimpleData('weight'),
        'Вага',
        'Динаміка зміни ваги',
      )}
    </TabPanel>
    <TabPanel value={activeTab} index={1}>
      {this.renderUserStatTab(
        this.getSimpleData('waist'),
        'Обсяг талії',
        'Динаміка обсягу талії',
      )}
    </TabPanel>
    <TabPanel value={activeTab} index={2}>
      {this.renderUserStatTab(
        this.getSimpleData('bust'),
        'Обсяг грудей',
        'Динаміка обсягу грудей',
      )}
    </TabPanel>
    <TabPanel value={activeTab} index={3}>
      {this.renderUserStatTab(
        this.getSimpleData('hips'),
        'Обсяг стегна',
        'Динаміка обсягу стегна',
      )}
    </TabPanel>
    <TabPanel value={activeTab} index={4}>
      {this.renderUserStatTab(
        this.imtData,
        'Індекс маси тіла',
        'Індекс маси тіла',
      )}
    </TabPanel>
    <TabPanel value={activeTab} index={5}>
      {this.renderUserStatTab(
        this.getSimpleData('sleep'),
        'Якість сну',
        'Динаміка якості сну',
      )}
    </TabPanel>
    <TabPanel value={activeTab} index={6}>
      {this.renderUserStatTab(
        this.moodData,
        'Настрій',
        'Динаміка зміни настрою',
      )}
    </TabPanel>
  </div>
)
```

```
  }

  private renderUserStatTab(data: any, valueName: string, name: string) {
    return (
      <Grid container>
        <Grid xs={4} md={4} lg={2}>
          <DataTable data={data} valueName={valueName} />
        </Grid>
        <Grid xs={8} md={8} lg={10}>
          <DataGraph data={data} name={name} />
        </Grid>
      </Grid>
    )
  }

  @autobind
  private handleChange(e: React.ChangeEvent<{}>, newValue: number) {
    this.setState({ activeTab: newValue })
  }

  private getSimpleData(name: string) {
    const { userStats } = this.props

    return userStats.map((us) => ({
      date: us.createdAt * 1000,
      value: (us as any)[name] as number,
    }))
  }

  private get moodData() {
    const { userStats } = this.props

    return userStats.map((us) => ({
      date: us.createdAt * 1000,
      value: us.moodAfter! - us.moodBefore!,
    }))
  }

  private get imtData() {
    const { userStats } = this.props

    return userStats.map((us) => ({
      date: us.createdAt * 1000,
      value: us.weight! / (us.height! * us.height!),
    }))
  }
}

export default compose(
  withStyles(styles),
  withRouter,
  connectToState,
)(UserStatsPageBase) as React.ComponentType<any>
```

UserStatsPage.types.ts

```
import { WithStyles } from '@material-ui/core'
import { styles } from './UserStatsPage.styles'
import { RouteComponentProps } from 'react-router-dom'
```

```
import { login } from '../../store/currentUser/currentUserActions'
import { IUser } from '../../entities/IUser'
import { IUserStats } from '../../entities/IUserStats'

type UserStatsPageStyles = typeof styles

export interface UserStatsPageBaseProps {
  login: typeof login
  currentUser: IUser
  isFetching: boolean
  userStats: IUserStats[]
}

export interface UserStatsPageStateProps {
  activeTab: number
}

export type UserStatsPageProps = UserStatsPageBaseProps &
  RouteComponentProps &
  WithStyles<UserStatsPageStyles>
```

DataTable.tsx

```
import {
  TableContainer,
  Paper,
  Table,
  TableHead,
  TableRow,
  TableCell,
  TableBody,
} from '@material-ui/core'
import React from 'react'

interface IStatsData {
  date: number
  value: number
}

interface DataTableProps {
  data: IStatsData[]
  valueName: string
}

export default class DataTable extends React.Component<DataTableProps> {
  render() {
    const { data, valueName } = this.props
    return (
      <TableContainer component={Paper}>
        <Table aria-label="simple table">
          <TableHead>
            <TableRow>
              <TableCell>Дата</TableCell>
              <TableCell align="right">{valueName}</TableCell>
            </TableRow>
          </TableHead>
          <TableBody>
            {data.map((row, i) => (
              <TableRow key={i}>
                <TableCell component="th" scope="row">
                  {new Date(row.date).toLocaleDateString('uk-UA')}
```

```
                </TableCell>
                <TableCell align="right">{row.value}</TableCell>
              </TableRow>
            ))}
          </TableBody>
        </Table>
      </TableContainer>
    )
  }
}
```

TabPanel.tsx

```
import { Box } from '@material-ui/core'

export interface TabPanelProps {
  children?: React.ReactNode
  index: any
  value: any
}

export function a11yProps(index: any) {
  return {
    id: `simple-tab-${index}`,
    'aria-controls': `simple-tabpanel-${index}`,
  }
}

export default function CustomTabPanel(props: TabPanelProps) {
  const { children, value, index, ...other } = props

  return (
    <div
      role="tabpanel"
      hidden={value !== index}
      id={`simple-tabpanel-${index}`}
      aria-labelledby={`simple-tab-${index}`}
      {...other}
      style={{ height: 'calc(100vh - 112px)' }}
    >
      {value === index && (
        <Box p={3} style={{ height: 'calc(100vh - 112px)' }}>
          {children}
        </Box>
      )}
    </div>
  )
}
```

UserController.cs

```
using Microsoft.AspNetCore.Mvc;
using Parashut.Api.Entities;
using Parashut.Api.Services.Abstraction;
using Parashut.Api.Utils;
using Parashut.Api.ViewModels;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
```

```
namespace Parashut.Api.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class UserController : ControllerBase
    {
        private readonly IUserService userService;

        public UserController(IUserService userService)
        {
            this.userService = userService;
        }

        [HttpPost("Authenticate")]
        public async Task<IActionResult> Authenticate(AuthenticateRequest request)
        {
            Result<AuthenticateResponse> result = await userService.Authenticate(request.TelegramUserId);

            if (result.IsSuccess)
            {
                return Ok(result.Item);
            }

            return StatusCode(result.StatusCode, result.ErrorMessage);
        }

        [HttpGet("CurrentUser")]
        public IActionResult GetCurrentUser()
        {
            Result<User> result = userService.GetCurrentUser();

            if (result.IsSuccess)
            {
                return Ok(result.Item);
            }

            return StatusCode(result.StatusCode, result.ErrorMessage);
        }

        [HttpGet("CurrentUserUsers")]
        public async Task<IActionResult> GetAllUsersForUser()
        {
            Result<List<User>> result = await userService.GetAllUsersForTrainer();

            if (result.IsSuccess)
            {
                return Ok(result.Item);
            }

            return StatusCode(result.StatusCode, result.ErrorMessage);
        }

        [HttpGet("UserStats")]
        public async Task<IActionResult> GetAllUserStatsForTrainee()
        {
            Result<List<UserStats>> result = await userService.GetUserStatsForTrainee();

            if (result.IsSuccess)
            {
                return Ok(result.Item);
            }
```

```
            return StatusCode(result.StatusCode, result.ErrorMessage);
        }
    }
}
```

UserService.cs

```
using Microsoft.AspNetCore.Http;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Options;
using Microsoft.IdentityModel.Tokens;
using Parashut.Api.Db;
using Parashut.Api.Entities;
using Parashut.Api.Services.Abstraction;
using Parashut.Api.Utils;
using Parashut.Api.ViewModels;
using System;
using System.Collections.Generic;
using System.IdentityModel.Tokens.Jwt;
using System.Linq;
using System.Security.Claims;
using System.Text;
using System.Threading.Tasks;

namespace Parashut.Api.Services
{
    public class UserService : IUserService
    {
        private readonly AppSettings appSettings;
        private readonly ParashutDbContext dbContext;
        private readonly IHttpContextAccessor httpContextAccessor;
        private readonly ILogger<UserService> logger;
        private readonly IGroupService groupService;

        //MYTODO: generate new Secret for prod.
        public UserService(
            IOptions<AppSettings> appSettings,
            ParashutDbContext dbContext,
            IHttpContextAccessor httpContextAccessor,
            ILogger<UserService> logger,
            IGroupService groupService)
        {
            this.appSettings = appSettings.Value;
            this.dbContext = dbContext;
            this.httpContextAccessor = httpContextAccessor;
            this.logger = logger;
            this.groupService = groupService;
        }

        public async Task<Result> AddMembersToGroup(long chatId, IEnumerable<Telegram.Bots.Types.User> memberDto
s)
        {
            try
            {
                Group userGroup = await dbContext.Groups.FirstOrDefaultAsync(_ => _.TelegramChatId == chatId);

                IEnumerable<User> newMembers = memberDtos
                    .Where(_ => !_.IsBot)
```

```
            .Select(_ =>
            new User()
            {
                Id = new Guid(),
                FirstName = _.FirstName,
                LastName = _.LastName,
                Username = _.Username,
                Role = Role.Trainee,
                IsDeleted = false,
                TelegramUserId = _.Id,
                GroupId = userGroup.Id
            }
        );

        if (!newMembers.Any())
        {
            return Result.Success();
        }

        await dbContext.Users.AddRangeAsync(newMembers);
        await dbContext.SaveChangesAsync();
        return Result.Success();
    }
    catch (Exception ex)
    {
        string errorMessage = "Unexpected error on adding users to group.";
        logger.LogError(ex, errorMessage);
        return Result.Fail(500, errorMessage);
    }
}

public async Task<Result<User>> Create(Telegram.Bots.Types.User userDto, Role userRole)
{
    try
    {
        User user = new User()
        {
            Id = new Guid(),
            FirstName = userDto.FirstName,
            LastName = userDto.LastName,
            Username = userDto.Username,
            Role = userRole,
            IsDeleted = false,
            TelegramUserId = userDto.Id
        };

        await dbContext.Users.AddAsync(user);
        await dbContext.SaveChangesAsync();
        return Result<User>.Success(user);
    }
    catch (Exception ex)
    {
        string errorMessage = "Unexpected error on user creation.";
        logger.LogError(ex, errorMessage);
        return Result<User>.Fail(500, errorMessage);
    }
}

public async Task<Result<AuthenticateResponse>> Authenticate(long telegramUserId)
{
    try
    {
```

```
        User user = await dbContext.Users
            .FirstOrDefaultAsync(_ => _.TelegramUserId == telegramUserId);

        if (user == null)
        {
            string errorMessage = $"User does not exist in the database.";
            logger.LogError(errorMessage);
            return Result<AuthenticateResponse>.Fail(StatusCodes.Status400BadRequest, errorMessage);
        }

        string token = GenerateJwtToken(user);

        return Result<AuthenticateResponse>.Success(new AuthenticateResponse(user, token));
    }
    catch (Exception ex)
    {
        string errorMessage = "Unexpected error on user authentication.";
        logger.LogError(ex, errorMessage);
        return Result<AuthenticateResponse>.Fail(StatusCodes.Status500InternalServerError, errorMessage);
    }
}

public async Task<Result<List<User>>> GetAll()
{
    try
    {
        List<User> users = await dbContext.Users.ToListAsync();
        return Result<List<User>>.Success(users);
    }
    catch (Exception ex)
    {
        string errorMessage = "Unexpected error on getting all users.";
        logger.LogError(ex, errorMessage);
        return Result<List<User>>.Fail(StatusCodes.Status500InternalServerError, errorMessage);
    }
}

/// <summary>
/// Get all users that belong to groups of a specific trainer.
/// </summary>
/// <returns></returns>
public async Task<Result<List<User>>> GetAllUsersForTrainer()
{
    try
    {
        User user = GetCurrentUser().Item;
        if (user.Role == Role.Trainer)
        {
            List<Guid> groupIds = (await groupService.GetAllGroupsForUser())
                .Item.Select(_ => _.Id).ToList();
            List<User> users = await dbContext.Users
                .Where(_ => _.GroupId.HasValue && groupIds.Contains((Guid)_.GroupId))
                .ToListAsync();

            // TODO: remove.
            User trainer = (await GetById(user.Id)).Item;
            users.Add(trainer);
            return Result<List<User>>.Success(users);
        }
        else
        {
            List<User> users = await dbContext.Users
```

```
                .Where(_ => _.GroupId == user.GroupId)
                .ToListAsync();
            return Result<List<User>>.Success(users);
        }

    }
    catch (Exception ex)
    {
        string errorMessage = "Unexpected error on getting all users.";
        logger.LogError(ex, errorMessage);
        return Result<List<User>>.Fail(StatusCodes.Status500InternalServerError, errorMessage);
    }
}

public async Task<Result<User>> GetById(Guid id)
{
    try
    {
        User user = await dbContext.Users.FindAsync(id);

        if (user == null)
        {
            string errorMessage = $"User with id = {id} not found.";
            logger.LogError(errorMessage);
            return Result<User>.Fail(StatusCodes.Status404NotFound, errorMessage);
        }

        return Result<User>.Success(user);
    }
    catch (Exception ex)
    {
        string errorMessage = "Unexpected error on getting user by id.";
        logger.LogError(ex, errorMessage);
        return Result<User>.Fail(StatusCodes.Status500InternalServerError, errorMessage);
    }
}

public Result<User> GetCurrentUser()
{
    try
    {
        if (httpContextAccessor.HttpContext.Items["User"] is not User currentUser)
        {
            string errorMessage = "Unauthorized";
            logger.LogError(errorMessage);
            return Result<User>.Fail(StatusCodes.Status401Unauthorized, errorMessage);
        }

        return Result<User>.Success(currentUser);
    }
    catch (Exception ex)
    {
        string errorMessage = "Unexpected error on getting current user.";
        logger.LogError(ex, errorMessage);
        return Result<User>.Fail(StatusCodes.Status500InternalServerError, errorMessage);
    }
}

public async Task<Result<User>> GetByTelegramId(long id)
{
    try
    {
```

```
            User user = await dbContext.Users.FirstOrDefaultAsync(_ => _.TelegramUserId == id);

            if (user == null)
            {
                string errorMessage = $"User with id = {id} not found.";
                logger.LogError(errorMessage);
                return Result<User>.Fail(StatusCodes.Status404NotFound, errorMessage);
            }

            return Result<User>.Success(user);
        }
        catch (Exception ex)
        {
            string errorMessage = "Unexpected error on getting user by telegram id.";
            logger.LogError(ex, errorMessage);
            return Result<User>.Fail(StatusCodes.Status500InternalServerError, errorMessage);
        }
    }

    public async Task<Result<List<UserStats>>> GetUserStatsForTrainee()
    {
        try
        {
            User user = GetCurrentUser().Item;
            List<UserStats> userStats = await dbContext.UserStats
                .Where(us => us.UserId == user.Id)
                .ToListAsync();

            return Result<List<UserStats>>.Success(userStats);
        }
        catch (Exception ex)
        {
            string errorMessage = "Unexpected error on getting user by telegram id.";
            logger.LogError(ex, errorMessage);
            return Result<List<UserStats>>.Fail(StatusCodes.Status500InternalServerError, errorMessage);
        }
    }

    private string GenerateJwtToken(User user)
    {
        // generate token that is valid for 8 hours
        JwtSecurityTokenHandler tokenHandler = new JwtSecurityTokenHandler();
        byte[] key = Encoding.ASCII.GetBytes(appSettings.Secret);

        SecurityTokenDescriptor tokenDescriptor = new SecurityTokenDescriptor
        {
            Subject = new ClaimsIdentity(
                new[]
                {
                    new Claim("id", user.Id.ToString())
                }),
            Expires = DateTime.UtcNow.AddHours(1),
            SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.Hm
acSha256Signature)
        };

        SecurityToken token = tokenHandler.CreateToken(tokenDescriptor);
        return tokenHandler.WriteToken(token);
    }
}
}
```