МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

15.03 — КМР. 1636–“С” 2024.10.29. 010 ПЗ

БАДРАКА МИКИТИ РОМАНОВИЧА

2024 р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету інформаційних технологій	Завідувач кафедри комп'ютерних наук
Глазунова О.Г., д.п.н., професор	Голуб Б.Л., к.т.н., доцент
_____	_____
____ _____ 2024 р.	**____ _____ 2024 р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему __Система моніторингу та аналізу показників фітнес-помічника_

Спеціальність ___122 - Комп'ютерні науки_____
(код і назва)

Освітня програма ____Інформаційно управляючі системи та технології_
(назва)

Орієнтація освітньої програми ____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

_____ _____ ___Густер О.М.__
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

_____к.т.н., доцент_____ _____ _____Даков С.Ю._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _____Бадрак М.Р._____
(підпис) (ПІБ студента)

КИЇВ - 2024

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) Інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"____" _____ 2024 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Бадрак Микита Романович
(прізвище, ім'я, по батькові)

Спеціальність 122 - Комп'ютерні науки
(код і назва)

Освітня програма Інформаційно управляючі системи та технології
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи Система моніторингу та аналізу показників фітнес-помічника

затверджена наказом ректора НУБіП України від "____"_____20___р. №_____

Термін подання завершеної роботи на кафедру _____
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи отримання звітів на основі аналізу власних та загальних даних щодо власних тренувань та показників харчування

Перелік питань, що підлягають дослідженню:

1. Системний аналіз предметної області
2. Моделювання системи
3. Розробка системи
4. Результати дослідження

Перелік графічного матеріалу (за потреби) _____

Дата видачі завдання "_____"_____20___ р.

Керівник магістерської кваліфікаційної роботи _____ Даков С.Ю.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Бадрак М.Р.
(підпис) (прізвище та ініціали студента)

ЗМІСТ

ВСТУП

У сучасному світі ведення здорового способу життя стало головною проблемою багатьох людей. Фітнес-помічники — як програмні додатки, так і переносні пристрої — відіграють вирішальну роль, допомагаючи користувачам відстежувати, аналізувати та покращувати своє фізичне здоров’я. Ці системи надають цінні дані про фізичну активність, частоту серцевих скорочень, режим сну та інші показники, необхідні для управління фітнесом. Незважаючи на зростання таких технологій, все ще існує потреба в більш досконалих, надійних системах, які дозволяють здійснювати точний моніторинг і комплексний аналіз індивідуальних показників фізичної форми. Це дослідження має на меті усунути цю прогалину, розробивши систему, яка не тільки відстежує, але й забезпечує глибоке розуміння цих показників, що робить її надзвичайно **актуальною** як для любителів фітнесу, так і для професіоналів у галузі охорони здоров’я.

Предметом дослідження є процес моніторингу та аналізу показників тренованості. **Об’єктом** є програмна система, призначена для моніторингу, збору та аналізу даних від фітнес-асистентів.

Мета цього дослідження полягає в тому, щоб покращити індустрію фітнесу та здоров'я принісши туди новий додаток, який допомагатиме стежити за своїми спортивними показниками та харчуванням.

Для досягнення мети дослідження будуть вирішені такі **завдання**:

1. виконати системний аналіз існуючих рішень для моніторингу фітнесу, щоб виявити прогалини та можливості;
2. змоделювати систему, включаючи архітектуру та функціональність для збору, зберігання та аналізу даних у реальному часі;
3. розробити програмну систему, здатну збирати, зберігати та аналізувати дані фітнес-асистента;
4. оцінити ефективність і точність розробленої системи за допомогою тестування та аналізу в реальному світі.

У цьому дослідженні використовуватиметься комбінація **методів дослідження**:

Технологія OLAP використовується для аналізу даних, перевірки гіпотез і розрахунку ключових показників продуктивності. Алгоритми інтелектуального аналізу даних використовуються для формування нових гіпотез на основі шаблонів індикаторів придатності, що дозволяє проводити прогнозний аналіз продуктивності користувача. Методи обробки великих даних застосовуються для обробки великої кількості даних, створених фітнес-помічниками, а також для забезпечення ефективного зберігання та пошуку даних. Моделі машинного навчання використовуються для аналізу історичних даних, надання персоналізованих рекомендацій і прогнозів для користувачів. UML-моделювання використовується для проектування архітектури системи та візуалізації потоків даних[14].

Наукова новизна: була розроблена система, яка поєднує розширений моніторинг індикаторів фітнес-асистента з аналізом даних у реальному часі, пропонуючи нову платформу для комплексного управління фітнесом. Було запропоновано покращити архітектуру системи, щоб забезпечити бездоганну інтеграцію збору, обробки та аналізу даних у реальному часі. Крім того, було впроваджено вдосконалення алгоритмів обробки даних, що оптимізує здатність системи аналізувати дані про фізичну форму та генерувати персоналізовану інформацію про фізичну форму. Було запропоновано набір інформаційних технологій, що дозволяє ефективно використовувати як методи OLAP, так і методи інтелектуального аналізу даних в рамках основних функцій системи.

Апробація результатів дослідження: результати дослідження були представлені на конференціях та опубліковані у відповідних наукових виданнях:

- презентація на Міжнародній конференції з інформатики охорони здоров’я, 2024 р;
- публікація в журналі Computational Fitness Technologies, 2024[16];
- тези, представлені на щорічній конференції з питань великих даних у сфері охорони здоров’я, 2024 р.

Структура даної магістерської роботи включає чотири основні розділи:

1. системний аналіз предметної області – у цьому розділі буде розглянуто існуючі рішення для моніторингу фітнесу, визначено ключові технології та висвітлено поточні обмеження в галузі;
2. моделювання системи – у цьому розділі розглядатимуться концептуальний дизайн і моделювання системи моніторингу та аналізу придатності;
3. розробка системи – у цьому розділі буде описано процес розробки запропонованої системи, включаючи ключові деталі впровадження;
4. результати дослідження – у заключному розділі будуть представлені результати впровадження та тестування системи, оцінка її продуктивності та надання рекомендацій щодо подальшого вдосконалення.

Магістерська робота містить 73 сторінки, в тому числі 30 рисунків, 1 таблицю, 2 додатки. Вона посилається на 30 джерел, включаючи наукові статті, книги та електронні ресурси.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Опис предметної області

Область цього дослідження лежить на перетині технологій відстеження фізичної активності, аналізу даних і управління особистим здоров'ям. Завдяки зростанню популярності смарт-годинників, фітнес-браслетів і мобільних додатків люди тепер можуть відстежувати широкий спектр даних, пов'язаних зі здоров'ям, таких як зроблені кроки, спалені калорії, пульс і режим сну. Незважаючи на те, що обсяг доступних даних продовжує зростати, залишається проблема зробити цю інформацію значущою та корисною для користувачів. Фітнес-помічники зараз чудово збирають дані, але їм часто не вистачає можливості надавати персоналізовану інформацію, довгостроковий аналіз тенденцій або детальні прогнози ефективності.

Фітнес-асистенти є основними інструментами для моніторингу фізичної активності. Ці пристрої, які зазвичай носять на зап'ясті або вбудовані в мобільні програми, відстежують різні аспекти фізичної продуктивності в режимі реального часу, включаючи кількість кроків, частоту серцевих скорочень і витрати калорій. Вони пропонують користувачам основні відгуки про їх щоденну діяльність, часто у формі підсумків або журналів активності. Наприклад, фітнес-помічник може повідомити користувача, що сьогодні він зробив 10 000 кроків або що його середній пульс під час тренування становив 120 ударів на хвилину. Незважаючи на доступність цих даних, користувачам часто важко інтерпретувати їх таким чином, щоб оптимізувати свої фітнес-процедури. Поточні системи представляють необроблені показники без особливого контексту, залишаючи прогалину в перетворенні цих даних у практичну інформацію[3].

Хоча фітнес-помічники ефективно збирають дані, вони обмежені в своїй здатності аналізувати та інтерпретувати довгострокові тенденції або створювати персоналізовані рекомендації. У більшості систем відсутні розширені

можливості, такі як історичний аналіз, прогнозне моделювання або персоналізація фітнес-цілей на основі індивідуального прогресу. Крім того, деякі платформи дозволяють користувачам порівнювати свою продуктивність з іншими в подібній демографічній групі або надавати значущі порівняння з їхньою попередньою продуктивністю з часом. У результаті користувачі залишаються з даними, які пропонують мінімальні вказівки щодо того, як покращити своє здоров'я чи ефективніше досягти цілей у формі.

Ключовим аспектом систем моніторингу фізичної активності є здатність перетворювати необроблені дані в цінну інформацію. Такі дані, як кількість кроків, частота серцевих скорочень і режим сну, мають обмежену цінність, якщо розглядати їх ізольовано. Однак, коли ці показники аналізуються комплексно, вони можуть запропонувати глибше уявлення про загальний прогрес у здоров'ї та фізичній формі людини. Наприклад, варіабельність серцевого ритму (ВСР) може служити корисним індикатором рівня стресу або фізичного відновлення, тоді як баланс між спожитими та спаленими калоріями може надати цінну інформацію для контролю ваги. Якість сну в поєднанні з рівнем щоденної активності може дати зрозуміти, чи певна поведінка впливає на загальний стан здоров'я. Сучасні фітнес-помічники збирають ці дані, але вони зазвичай не виконують глибокого аналізу, необхідного для перетворення цих показників на корисні, практичні поради.

Основною проблемою, яку намагається вирішити це дослідження, є відсутність розширеної системи, здатної як відстежувати фітнес-індикатори, так і проводити ретельний аналіз даних. Хоча існуючі фітнес-асистенти зосереджені в основному на зборі даних, їм бракує можливостей, необхідних для персоналізації інформації на основі індивідуальних потреб, пропонування прогнозного аналізу або ефективного відстеження довгострокової продуктивності. Мета полягає в тому, щоб створити систему, яка не тільки об'єднує дані з різних джерел, але й уможливлює аналіз у реальному часі та історичний аналіз, персоналізовані рекомендації та застосування прогнозної

аналітики. Це дозволить користувачам приймати обґрунтовані рішення на основі даних щодо своїх розпорядків фітнесу та управління здоров’ям.

Показники фітнесу, які будуть у центрі уваги цього дослідження, включають частоту серцевих скорочень, спалені калорії, кількість кроків, пройдену відстань, якість сну, інтенсивність активності та періоди відпочинку/відновлення. Ці показники надають повний огляд фізичної активності та загального стану здоров’я користувача.

У досліджуваній предметній області можна виділити кілька ключових бізнес-процесів, що визначають функціонал системи моніторингу та аналізу показників фітнес-асистента.

Одним із основних процесів є можливість розпочати тренування, яке може бути перервано або завершено за бажанням користувача. Під час тренувального процесу користувач має змогу вводити власні показники для точнішого відстеження прогресу. Крім того, важливою функцією є встановлення бажаного результату, на основі якого система може адаптувати рекомендації та інтенсивність навантажень. Серед інших важливих можливостей — отримання повідомлень про зміну інтенсивності тренування, що допомагає регулювати фізичну активність в режимі реального часу. Також користувач може налаштовувати параметри системи відповідно до своїх потреб, зберігати або завантажувати раніше збережені налаштування для подальшого використання. Окрім цього, система веде детальну статистику тренувань та фізичних показників користувача, таких як кількість води, спожиті калорії та кроки. Це дозволяє користувачеві відстежувати свої досягнення та контролювати загальний стан здоров'я[2].

Детальніший опис класів та атрибутів предметної області подано в таблиці 1.1.

Таблиця 1.1

Опис атрибутів класів предметної області

Клас предметної області	**Атрибут**	**Опис**
Тренування	Список часу інтервалів	Інформація, отримана з налаштувань, використовується для відображення тривалості тренування та надсилання повідомлень користувачеві у точно зазначений час.
Статистика	Вага	Інформація щодо вказаної та бажаної ваги
	Калорії	Дані про спожиті калорії за день збираються, і на їх основі розраховується середнє значення з усієї кількості калорій.
	Кроки	Дані про кількість пройдених кроків за день фіксуються, і на їх основі обчислюється середнє значення з загальної кількості кроків.
	Вода	Дані про обсяг спожитої води за день відстежуються, що дозволяє розрахувати середнє значення з загальної кількості випитої води.
Налаштування	Кількість інтервалів	Це числове значення відображає кількість змін інтенсивності тренування від високої до низької.
	Бажаний результат	Показники, такі як бажана вага, кількість пройдених кроків, спожиті калорії та обсяг випитої води за день
Сповіщення	Нагадування	Система надсилає повідомлення, якщо користувач тривалий час відхиляється від встановлених норм

Сучасне програмне забезпечення для фітнес-помічників часто не забезпечує достатньої персоналізації, не пропонуючи індивідуальних рекомендацій, що відповідають потребам, уподобанням і цілям користувачів. Замість цього, воно обмежується загальними планами тренувань або рекомендаціями щодо харчування, які не враховують специфічні вимоги чи обмеження. Додатково, відсутність інтеграції з носимими пристроями та іншими інструментами для моніторингу фізичної активності знижує точність та ефективність цих рішень[5].

Багато людей також відчувають труднощі в підтриманні мотивації та дотриманні своїх фітнес-програм без належного керівництва та підтримки. Без отримання зворотного зв’язку в режимі реального часу, відстеження прогресу та заходів підзвітності людям важко залишатися на правильному шляху та вносити сталості в свій спосіб життя.

Іншою суттєвою проблемою є обмежена соціальна взаємодія та відсутність спільноти в існуючому програмному забезпеченні для фітнес-помічників. Багато людей отримують переваги від спілкування з однодумцями, обміну досягненнями та отримання підтримки від однолітків. Відсутність надійних функцій для взаємодії заважає створенню сприятливого та надихаючого середовища.

Крім того, недостатній аналіз даних та інформації в поточному програмному забезпеченні обмежує можливості користувачів у розумінні свого прогресу, визначенні закономірностей та прийнятті обґрунтованих рішень для оптимізації своїх фітнес-результатів. Без детальних звітів і аналізу користувачі можуть стикатися з труднощами у прийнятті обґрунтованих рішень і, як наслідок, не досягати бажаних результатів у своїй фізичній формі.

1.2 Огляд існуючих рішень

Розглянемо існуючі рішення предметної області.

MyFitnessPal — це популярна мобільна програма та веб-платформа, призначена для допомоги користувачам у відстеженні їх дієти та фізичних вправ. Запущений у 2005 році та пізніше придбаний Under Armour у 2015 році, він отримав широке визнання завдяки своєму зручному інтерфейсу та великій базі даних продуктів харчування[1].

Ось ключові функції та функції MyFitnessPal:

Додаток пропонує інтуїтивно зрозумілий дизайн, який дозволяє користувачам легко переміщатися між різними функціями, що робить його доступним для людей будь-якого віку та рівня технічної підкованості.

Інтерфейс додатку «MyFitnessPal» зображено на рис.1.

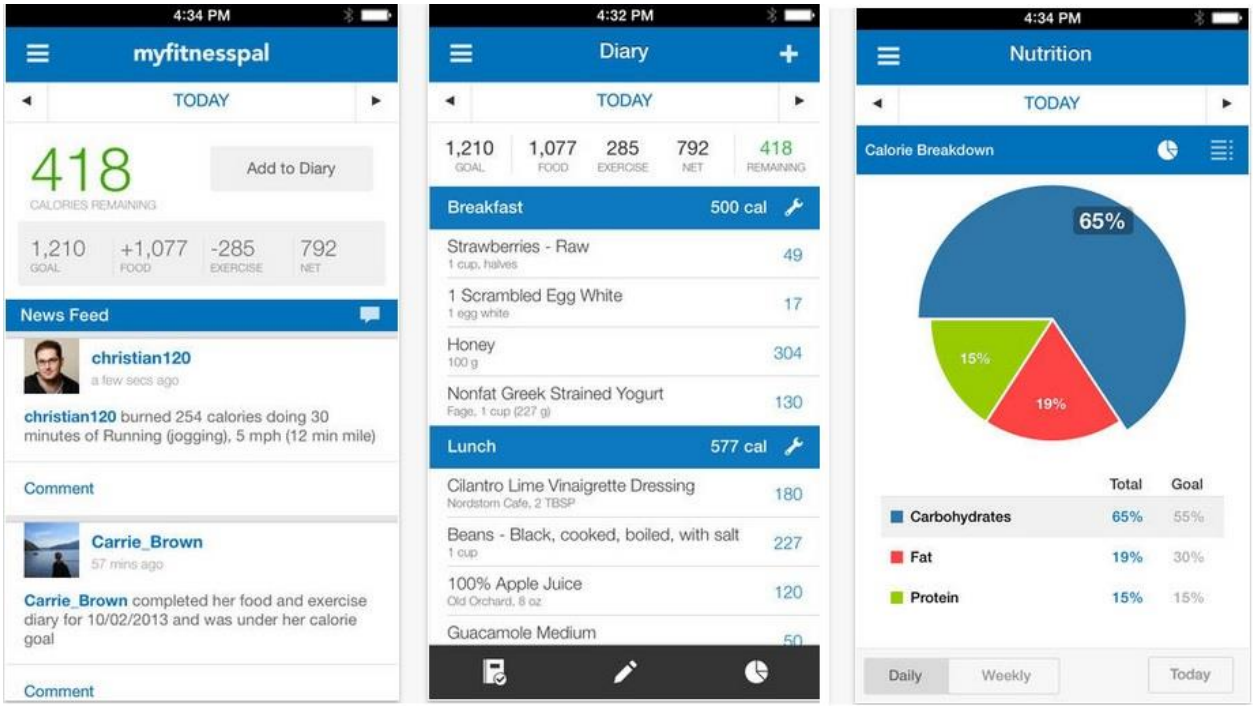


Рис. 1 Мобільний додаток «MyFitnessPal»

Однією з основних функцій MyFitnessPal є його щоденник харчування, який дозволяє користувачам реєструвати щоденне споживання їжі. Завдяки базі даних, що містить понад 11 мільйонів продуктів харчування, користувачі можуть швидко знаходити та додавати продукти, включаючи фірмові продукти та ресторанні страви. Додаток розраховує та відображає калорійність страв, допомагаючи користувачам залишатися в межах щоденної калорійності.

Окрім підрахунку калорій, MyFitnessPal надає детальний розподіл поживних речовин, включаючи макроелементи (вуглеводи, білки та жири) та

мікроелементи (вітаміни та мінерали). Ця функція допомагає користувачам робити обґрунтований вибір щодо своїх харчових звичок і розуміти харчову цінність своїх страв.

Користувачі можуть відстежувати свою фізичну активність, реєструючи тренування, які програма перетворює на спалені калорії. MyFitnessPal дозволяє користувачам підключатися до різних фітнес-пристроїв і додатків, таких як Fitbit і Apple Health, для безперебійного відстеження тренувань і щоденних рівнів активності.

Додаток дозволяє користувачам встановлювати персоналізовані цілі щодо здоров'я та фітнесу, як-от схуднення, збільшення ваги або збереження. MyFitnessPal пропонує інструменти відстеження прогресу, включаючи журнали ваги та вимірювань, щоб візуалізувати прогрес з часом за допомогою графіків і статистики.

MyFitnessPal містить соціальний компонент, що дозволяє користувачам спілкуватися з друзями, приєднуватися до груп і брати участь у форумах. Ця функція створює відчуття спільності та підтримки, дозволяючи користувачам ділитися своїми подорожами, проблемами та успіхами.

Користувачі можуть легко імпортувати рецепти та створювати плани харчування в програмі. MyFitnessPal розраховує харчову інформацію для рецептів і пропонує пропозиції щодо здоровіших альтернатив, що робить планування їжі простішим і ефективнішим[1].

Хоча базова версія MyFitnessPal безкоштовна, користувачі можуть вибрати преміум-підписку, яка відкриває додаткові функції, такі як розширена інформація про харчування, використання без реклами та можливість встановлювати власні цілі щодо макронутрієнтів.

MyFitnessPal зарекомендував себе як комплексний інструмент для людей, які прагнуть покращити своє здоров'я та фізичну форму за допомогою відстеження дієти та фізичних вправ. Його обширна база даних, зручний дизайн і підтримка спільноти роблять його цінним ресурсом для тих, хто починає займатися фітнесом.

Мобільний додаток «FatSecret» є рішенням для автоматизації процесу запису власних показників та ведення щоденника [1].

FatSecret — це комплексний мобільний додаток і онлайн-платформа, розроблена, щоб допомогти користувачам стежити за своїм харчуванням, фізичними вправами та загальним станом здоров'я. Запущений у 2007 році, FatSecret перетворився на популярний вибір для людей, які прагнуть контролювати свою вагу та вести здоровий спосіб життя. Нижче наведено основні характеристики та функції FatSecret.

Інтерфейс додатку «FatSecret» представлено на рис.2.

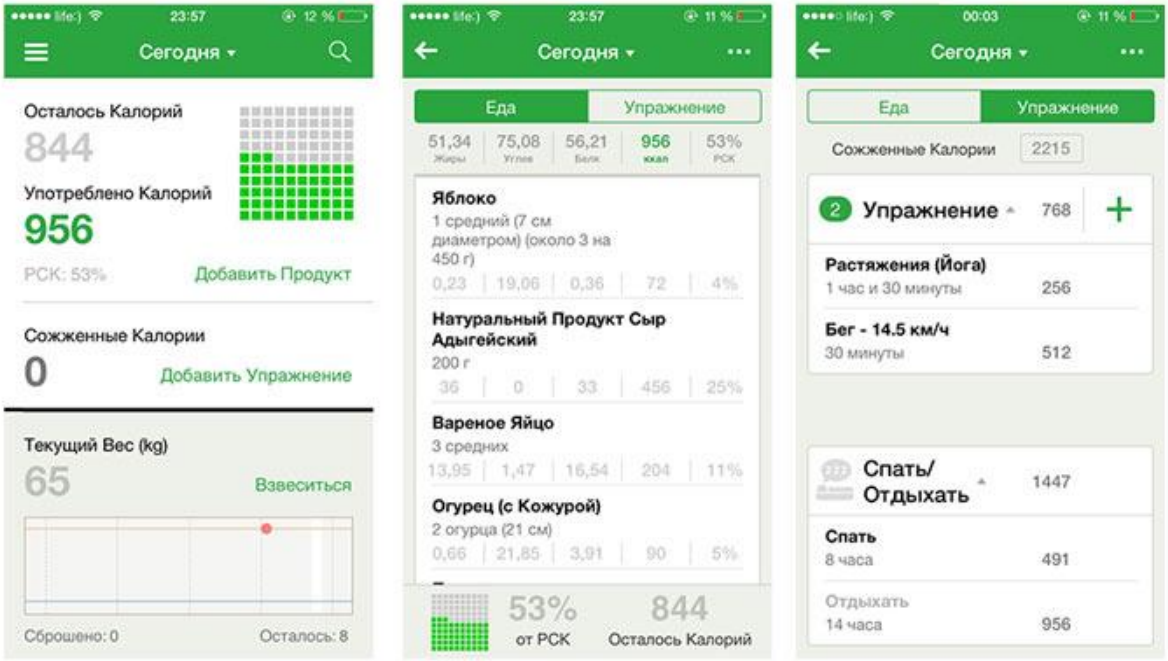


Рис. 2 Мобільний додаток "FatSecret"

FatSecret пропонує простий та інтуїтивно зрозумілий інтерфейс, який полегшує користувачам навігацію програмою та доступ до її функцій. Дизайн спрямований на те, щоб зробити процес відстеження максимально бездоганним, обслуговуючи користувачів із різними технічними можливостями[2].

Основною особливістю FatSecret є щоденник їжі, який дозволяє користувачам реєструвати щоденні прийоми їжі та закуски. Додаток може похвалитися базою даних з понад 1 мільйона продуктів харчування, включаючи загальні продукти харчування, ресторанні страви та фірмові продукти. Ця

обширна база даних спрощує процес реєстрації споживання їжі, допомагаючи користувачам ефективно контролювати споживання калорій.

Окрім підрахунку калорій, FatSecret надає детальну інформацію про харчування, включаючи розподіл макроелементів (вуглеводів, білків і жирів), а також інформацію про вітаміни та мінерали. Це дозволяє користувачам краще розуміти свої харчові звички та робити усвідомлений вибір щодо свого харчування.

Користувачі можуть реєструвати свої фізичні навантаження та тренування, а програма розраховує спалені калорії на основі введених вправ. FatSecret також підтримує інтеграцію з фітнес-трекерами та пристроями, що дозволяє користувачам легко синхронізувати дані про свою діяльність.

FatSecret дозволяє користувачам встановлювати персональні цілі щодо здоров'я, будь то втрата ваги, підтримка ваги або набір м'язової маси. Додаток містить інструменти відстеження прогресу, включаючи журнали ваги та вимірювання тіла, щоб допомогти користувачам візуалізувати свою подорож за допомогою детальних графіків і статистики.

FatSecret сприяє створенню почуття спільності через свої соціальні функції, де користувачі можуть спілкуватися з друзями, ділитися своїм досвідом і приєднуватися до груп, орієнтованих на конкретні цілі щодо здоров'я та фітнесу. Цей аспект спільноти забезпечує мотивацію та відповідальність, заохочуючи користувачів залишатися відданими своїм подорожам для здоров'я.

Додаток містить функцію пошуку рецептів, яка дозволяє користувачам знаходити здорові рецепти на основі їхніх дієтичних уподобань. Користувачі також можуть створювати плани харчування, за допомогою яких програма автоматично розраховує харчову інформацію для кожного прийому їжі, що полегшує дотримання дієтичних цілей[2].

FatSecret доступний як на мобільних пристроях, так і на веб-платформах, забезпечуючи користувачам доступ до своїх облікових записів і відстеження прогресу в будь-який час і в будь-якому місці. Ця гнучкість дозволяє користувачам залишатися на висоті своїх цілей щодо здоров'я.

Незважаючи на те, що FatSecret пропонує безкоштовну версію, вона також надає преміум-підписку з додатковими функціями, такими як покращена інформація про харчування, перегляд без реклами та інструменти персоналізованого планування їжі.

FatSecret виділяється як надійний інструмент для людей, які прагнуть покращити своє здоров’я та фізичну форму шляхом ретельного відстеження свого харчування та фізичної активності. Завдяки великій базі даних, зручному дизайну та спільноті підтримки FatSecret надає цінні ресурси, щоб допомогти користувачам на шляху до здорового способу життя.

1.3 Постановка завдання

Основною умовою даного проекту є ведення користувачами записів про свої показники, такі як спожиті калорії, кількість випитої води, пройдені кроки та вага. Для цього розроблена програма використовує базу даних, в яку користувачі можуть вносити інформацію, а згодом отримувати персоналізовані звіти про свої досягнення.

Основна мета програмного забезпечення полягає в тому, щоб надати користувачам можливість швидко отримувати важливу інформацію, що стосується їхнього фізичного стану.

Основне завдання програмного забезпечення, що розробляється - надати можливість користувачам швидко отримувати інформацію про:

- ефективність власних тренувань;
- правильність вживання води та їжі;
- кількість пройдених кроків;
- залишок, який потрібно виконати для необхідної норми.

Програма функціонує в діалоговому режимі з використанням меню. Користувач обирає запит, і програма виконує відповідні дії відповідно до його вибору.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Загальні відомості

UML, або уніфікована мова моделювання, служить стандартизованою мовою моделювання, яка широко використовується в розробці програмного забезпечення для візуалізації, специфікації, побудови та документування програмних систем. Його основна мета — надати набір графічних нотацій, які полегшують моделювання програмних додатків, покращуючи спілкування між розробниками, дизайнерами та зацікавленими сторонами.

Мета UML багатогранна. Він пропонує візуальне представлення, яке спрощує розуміння структур і поведінки програмної системи. Встановлюючи спільну мову, UML усуває прогалини в спілкуванні між різними зацікавленими сторонами, гарантуючи, що кожен має чітке розуміння проекту. Крім того, діаграми UML служать офіційною документацією, допомагаючи як у проектуванні, так і в поточному обслуговуванні програмних систем. Крім того, UML підтримує різні етапи розробки програмного забезпечення, від початкового аналізу та проектування до реалізації та тестування.

UML складається з кількох типів діаграм, які можна розділити на дві основні групи: структурні діаграми та діаграми поведінки. Структурні діаграми ілюструють статичні аспекти системи, зосереджуючись на організації та зв’язках між компонентами. Ключові приклади включають діаграми класів, які детально описують класи, атрибути, методи та їхні взаємозв’язки; діаграми компонентів, які показують організацію та залежності програмних компонентів; діаграми розгортання, які ілюструють фізичне розташування артефактів на апаратних вузлах; і діаграми об'єктів, які представляють екземпляри класів і їхні зв'язки в даний момент[10].

З іншого боку, поведінкові діаграми зображують динамічні аспекти системи, фіксуючи взаємодії та зміни з часом. До важливих типів належать діаграми варіантів використання, які окреслюють функціональні вимоги шляхом

зображення акторів та їх взаємодії з системою; діаграми послідовності, які описують взаємодію об’єктів і послідовність обміну повідомленнями; діаграми діяльності, які моделюють робочі процеси системи, демонструючи потік керування та даних; і діаграми стану, які представляють різні стани об’єкта та переходи між цими станами на основі подій.

UML знаходить численні застосування в розробці програмного забезпечення. Він відіграє життєво важливу роль на різних етапах життєвого циклу розробки, включаючи збір вимог, проектування системи та впровадження. Хоча традиційно асоціюється з більш традиційними методологіями розробки програмного забезпечення, UML також можна адаптувати для гнучких середовищ, пропонуючи легкий підхід до моделювання системи. Крім того, UML виявляється безцінним для аналізу та документування застарілих систем, сприяючи більш чіткому розумінню існуючих програм і інформуючи про плани оновлення або міграції. Використання діаграм UML сприяє кращому спілкуванню та співпраці між членами команди та зацікавленими сторонами, забезпечуючи спільне розуміння системи[12].

По суті, UML є потужним інструментом, який забезпечує стандартизований підхід до візуалізації та документування систем програмного забезпечення. Його різноманітні діаграми покращують розуміння та передачу складних ідей, що в кінцевому підсумку призводить до створення краще розроблених та зручніших програмних рішень.

У цій роботі також будуть використані методи інтелектуального аналізу даних.

Розуміння інтелектуального аналізу даних вимагає глибокого вивчення різних процесів, пов’язаних з керуванням даними, включаючи очищення, розпізнавання образів, проектування моделі та тестування. Цей комплексний підхід об’єднує елементи машинного навчання, статистики та керування базами даних, відрізняючи інтелектуальний аналіз даних від суміжних галузей, таких як аналітика даних і наука про дані.

В основі ефективного аналізу даних лежить кілька ключових концепцій і методів, які полегшують навігацію складними наборами даних. Перший крок передбачає очищення та підготовку даних, коли необроблена інформація з багатьох джерел стандартизується для забезпечення послідовності та сумісності для подальшого аналізу. Цей етап включає виявлення помилок, усунення відсутніх значень і видалення дублікатів, серед інших дій попередньої обробки.

Штучний інтелект (ШІ) відіграє вирішальну роль у видобутку даних, імітуючи когнітивні процеси людини, дозволяючи системам виконувати такі завдання, як планування, міркування, вирішення проблем і навчання. Інтеграція штучного інтелекту значно розширює аналітичні можливості, спрощуючи отримання інформації з даних.

Ще один важливий метод — вивчення правил асоціації, який зазвичай називають аналізом ринкового кошика. Цей метод визначає зв’язки між змінними в наборі даних, допомагаючи виявити закономірності спільного виникнення та надаючи розуміння поведінки клієнтів, зокрема щодо асоціацій продуктів.

Кластеризація — це метод, який сегментує великі набори даних на значущі підмножини або кластери. Цей процес розкриває базові закономірності та зв’язки між точками даних, уможливлюючи цілеспрямований аналіз і групуючи схожі елементи для більш ефективної обробки.

Класифікація передбачає класифікацію елементів у наборі даних за попередньо визначеними класами, покращуючи точність прогнозування для майбутніх точок даних. Ця техніка допомагає визначити закономірності та тенденції в даних, підтримуючи обґрунтоване прийняття рішень.

Коли дані оброблені та підготовлені, застосовуються методи аналізу даних, щоб отримати інформацію, виявити тенденції та створити практичну інформацію. Цей етап є критично важливим для перетворення необроблених даних у цінну інформацію для тих, хто приймає рішення.

Сховища даних є невід’ємною частиною великомасштабних ініціатив інтелектуального аналізу даних, оскільки включають зберігання великих обсягів

бізнес-даних у структурах, оптимізованих для швидкого пошуку й аналізу. Ефективне сховище даних забезпечує своєчасне прийняття рішень, надаючи доступ до повних наборів даних.

Регресійний аналіз є ще одним цінним методом, який передбачає числові значення в заданому діапазоні на основі існуючих шаблонів даних. Він знаходить застосування для прогнозування різних кількісних змінних, таких як температура, ціни на акції та показники продажів.

Сформувавши ці базові концепції, тепер ми можемо заглибитися в типовий робочий процес проекту інтелектуального аналізу даних, щоб побачити, як ці методи застосовуються на практиці.

2.2 Об’єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Діаграма варіантів використання — це візуальне представлення, яке ілюструє взаємодію між користувачами (акторами) і системою для досягнення конкретних цілей. Він надає огляд функціональності системи та висвітлює зв’язки між різними учасниками та сценаріями використання[14].

Компоненти діаграми варіантів використання

Актори - це зовнішні сутності, які взаємодіють із системою. Актори можуть представляти користувачів, інші системи або апаратні компоненти. Кожен учасник має певні ролі та обов’язки в системі.

Випадки використання - це конкретні дії або завдання, які система виконує у відповідь на запит актора. Варіанти використання часто

зображуються овалами на діаграмі та описують функціональні можливості, які надає система.

Відносини - лінії, що з'єднують акторів із варіантами використання, ілюструють взаємодію між ними. Ці відносини можуть приймати різні форми:

Асоціація - проста лінія, що з'єднує актора з варіантом використання, вказуючи, що актор бере участь у цьому варіанті використання.

Спроєктована діаграма прецедентів представлена на рис.3.

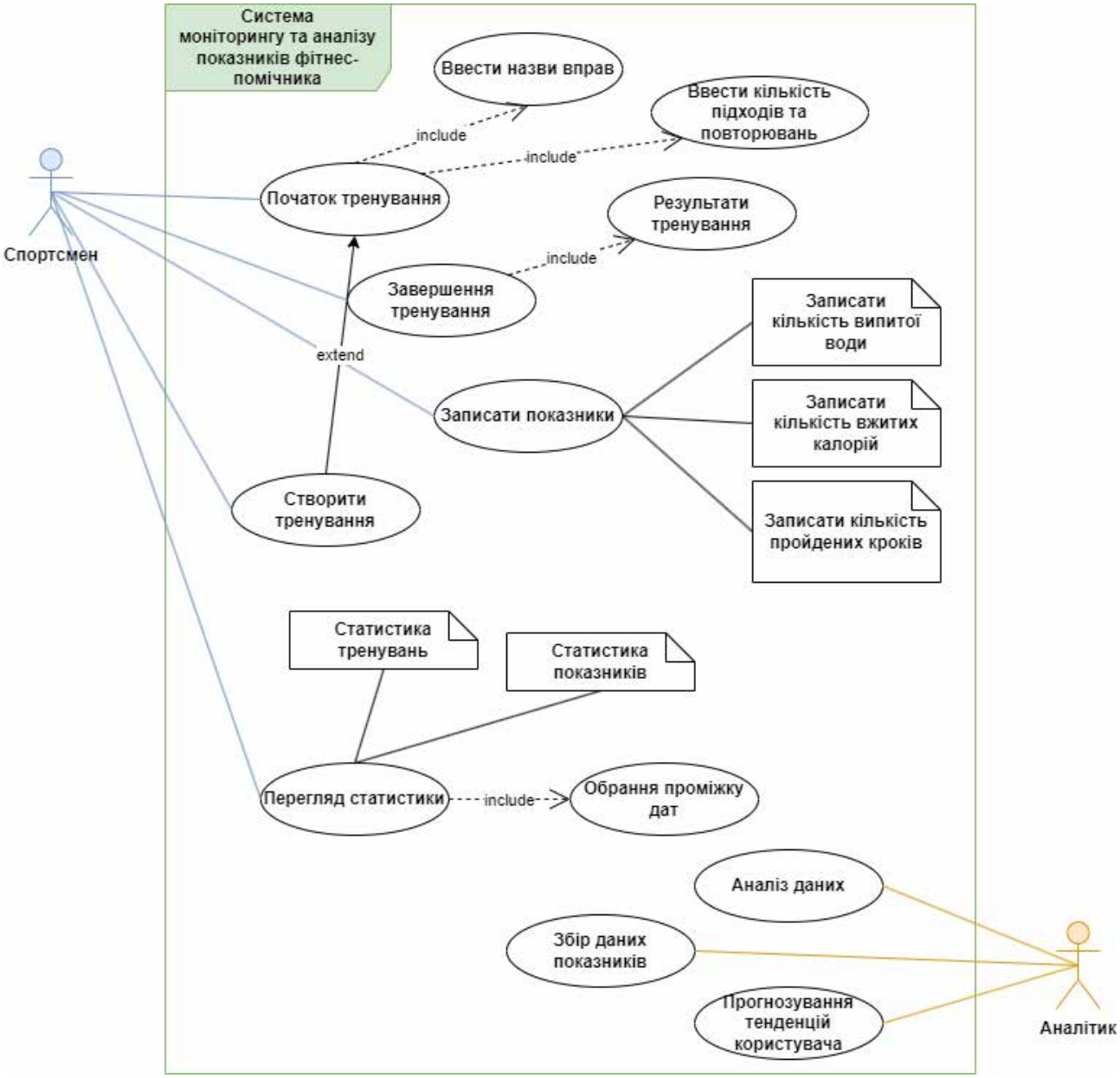


Рис. 3 Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Спортсмен”;
- “Аналітик”.

Актор «Спортсмен» включає такі прецеденти:

- початок тренування;
- завершення тренування;
- записати показники;
- перегляд статистики.

Актор «Аналітик» виконує такі прецеденти:

- аналіз даних;
- збір даних показників;
- прогнозування тенденцій користувача.

Прецеденти певним чином залежать одне від одного.

Варіант використання «Почати тренування» розширено додатковими варіантами використання, такими як «Введіть назви вправ» і «Вхідні набори та повторення». Подібним чином сценарій використання «Запис показників» охоплює анотації на зразок «Зареєструвати споживання води», «Записати споживання калорій» і «Відстежити зроблені кроки». Крім того, варіант використання «Переглянути статистику» містить анотації для «Статистики тренування» та «Статистика показників», а також включає варіант використання «Вибір діапазону дат».

Розглянемо детальніше вищеописані прецеденти.

Назва випадку використання: "Початок навчання"

Основною перевагою цього прецеденту є надання можливості спортсмену ініціювати тренувальну сесію, реєструючи вправи та супутні деталі для свого тренування на поточний день.

Актори:

Спортсмен: Користувач, який використовує додаток для відстеження своїх тренувальних сесій.

Аналітик: Особа, відповідальна за аналіз даних тренувань та їх

візуалізацію для спортсмена, що залежить від оптимізації тренувального процесу.

Передумови:

Спортсмен має бути увійшов до системи. Спортсмен повинен мати доступ до сторінок тренувань.

Оптимістичний сценарій:

1. спортсмен переходить на сторінку тренувальних сесій;
2. спортсмен натискає кнопку "Створити тренування";
3. відкривається форма, в яку спортсмен вводить: назву вправи, кількість підходів, кількість повторень, додатковий коментар (за бажанням);
4. система перевіряє введені дані;
5. виявляєте порожні поля;
6. перевіряє числові значення для підходів та повторень;
7. переконується, що назва вправи не містить заборонених символів;
8. після успішної перевірки система зберігає деталі третьої сесії в базі даних;
9. спортсмен отримує підтвердження про успішну реєстрацію тренування.

Прагматичний сценарій:

1. спортсмен вводить недопустимі символи у назві вправи або нечислові значення для підходів та повторень;
2. система перевірки наявності помилок у формі заповнення;
3. спортсмен отримує повідомлення, яке вказує на помилки у введенні;
4. неправильні символи у назві вправи;
5. ненумеричні дані для підходів чи повторень;
6. спортсмену пропонується виправити помилки та повторно надіслати форму;
7. після повторної подачі система перевіряє дані;
8. якщо спортсмен не відповідає на повідомлення про помилки;

9. система скасовує запис тренувань і може перенаправити спортсмена на основну сторінку тренувань.

Назва прецеденту використання: «Аналіз даних»

Основною перевагою цього прецеденту є надання можливості користувачеві (аналітику або спортсмену) за допомогою аналізу даних, що стосуються тренувань, харчування та інших показників. Це дозволяє користувачам отримати цінну інформацію для оптимізації своїх тренувань і покращення результатів.

Актори: Аналітик

Передумови: Користувач має бути зареєстрованим і увійшовши у систему. У системі повинні бути зібрані дані про тренування, харчування та інші показники.

Оптимістичний сценарій:

1. користувач відкриває розділ аналізу даних у додатку;
2. користувач обирає тип даних для аналізу, наприклад: тренувальні сесії, харчування, загальний прогрес;
3. користувач надає параметри для аналізу, включаючи: період (дані початку і закінчення), вибір спеціальних вправ або раціону;
4. користувач підтверджує запит на проведення аналізу;
5. система обробляє запит, виконує виконання обчислення та формує результат;
6. користувач отримує звіт, що містить: графіки та діаграми, які ілюструють прогрес, статистичні дані про тренування і харчування, рекомендації на основі отриманих даних;
7. користувач може зберегти результати аналізу в системі або експортувати їх у формати PDF, Excel та ін.

Прагматичний сценарій:

1. користувач вибирає неправильний формат дати або недоступний період для аналізу.
2. система перевіряє введені параметри на коректність;

3. у разі помилки система повідомляє користувача про невірно введені дані та пропонує їх виправити;
4. користувач виправляє помилки і повторно подає запит на аналіз;
5. система обробляє нові параметри та формує результат.

2.2.2 Діаграма послідовності. Діаграма послідовності служить візуальним зображенням взаємодії між об’єктами в системі протягом визначеного періоду часу. Він підкреслює хронологічний порядок повідомлень, якими обмінюються різні компоненти для виконання певної функції. Ця діаграма є частиною Уніфікованої мови моделювання (UML) і зазвичай використовується в розробці програмного забезпечення для ілюстрації динамічної поведінки системи[13].

Ключові елементи діаграми послідовності включають лінії життя, панелі активації, повідомлення, зворотні повідомлення та примітки. Лінії життя представляють кожного учасника взаємодії у вигляді вертикальної пунктирної лінії, яка може символізувати об’єкти, акторів або системи, що беруть участь у процесі. Смуги активації, зображені у вигляді прямокутників на лінії життя, вказують тривалість, протягом якої об’єкт активний або керує взаємодією.

Діаграма послідовностей, зображена на рис. 4, містить такі об’єкти:

- “Спортсмен”;
- “Аналітик”;
- “Тренування”;
- “Статистика”.

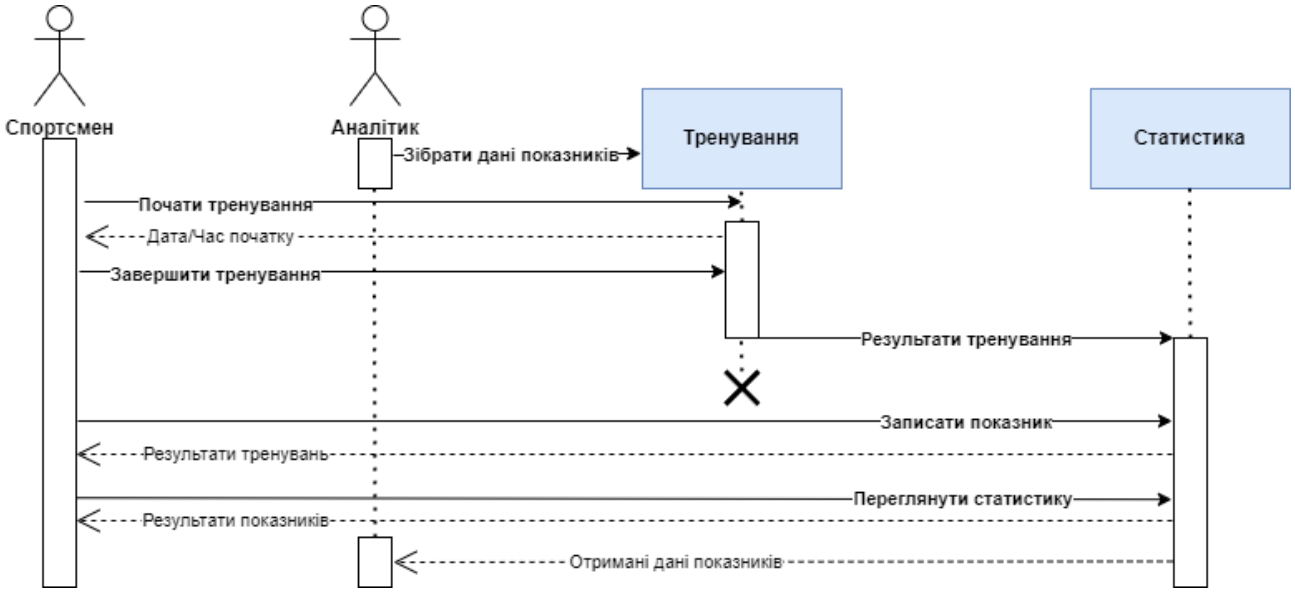


Рис. 4 Діаграма послідовності

Об’єкт «Спортсмен» ініціює запит до об’єкта «Тренування» на початок тренування та отримує відповідну дату та час сеансу. Після цього спортсмен надсилає запит на завершення тренування, що запускає передачу даних тренування в об’єкт «Статистика», де вони будуть зберігатися для подальшого

використання. Після завершення цього процесу об'єкт «Навчання» завершує свою роботу.

Далі об’єкт «Спортсмен» надсилає запит об’єкту «Статистика» для запису показників ефективності, отримуючи у відповідь результати своєї статистики. Крім того, спортсмен просить переглянути свою статистику та отримує відповідні статистичні дані на основі свого запиту.

2.2.3 Діаграма активності. Діаграма активності — це спеціалізована діаграма UML, яка ілюструє потік дій у системі чи процесі. Він надає візуальне уявлення про те, як різні завдання взаємопов’язані, фіксуючи динамічні аспекти системи. Ключові компоненти діаграми активності включають дії, представлені заокругленими прямокутниками, які позначають виконувані завдання або операції. Стрілки, що з’єднують ці дії, вказують на потік керування, окреслюючи послідовність, у якій відбуваються дії.

Діаграма починається із зафарбованого кола, що позначає початок процесу, і завершується зафарбованим колом, оточеним рамкою, що символізує кінець. Точки прийняття рішень, показані у вигляді ромбоподібних форм, дозволяють потоку розгалужуватися на основі конкретних умов, що веде до різних шляхів залежно від результату[16].

Крім того, діаграми активності можуть містити розгалуження, які розділяють потік на одночасні дії, і з’єднання, які синхронізують кілька потоків назад в один шлях. Доріжки, які є вертикальними або горизонтальними поділами, також можуть бути включені для представлення різних учасників або компонентів, відповідальних за певну діяльність. Це допомагає уточнити ролі та відповідальність у процесі.

Потоки керування керують порядком виконання дій, гарантуючи, що діаграма точно відображає заплановану послідовність операцій. Діаграми активності є безцінними для моделювання робочих процесів, бізнес-процесів і системних операцій, оскільки вони дають зацікавленим сторонам чітке розуміння того, як завдання пов’язані одне з одним. Ця візуалізація допомагає

виявити потенційні вузькі місця та сприяє покращенню спілкування між членами команди, що зрештою сприяє більш ефективному підходу до управління процесами.

Розроблена діаграма активності представлена на рис.5.

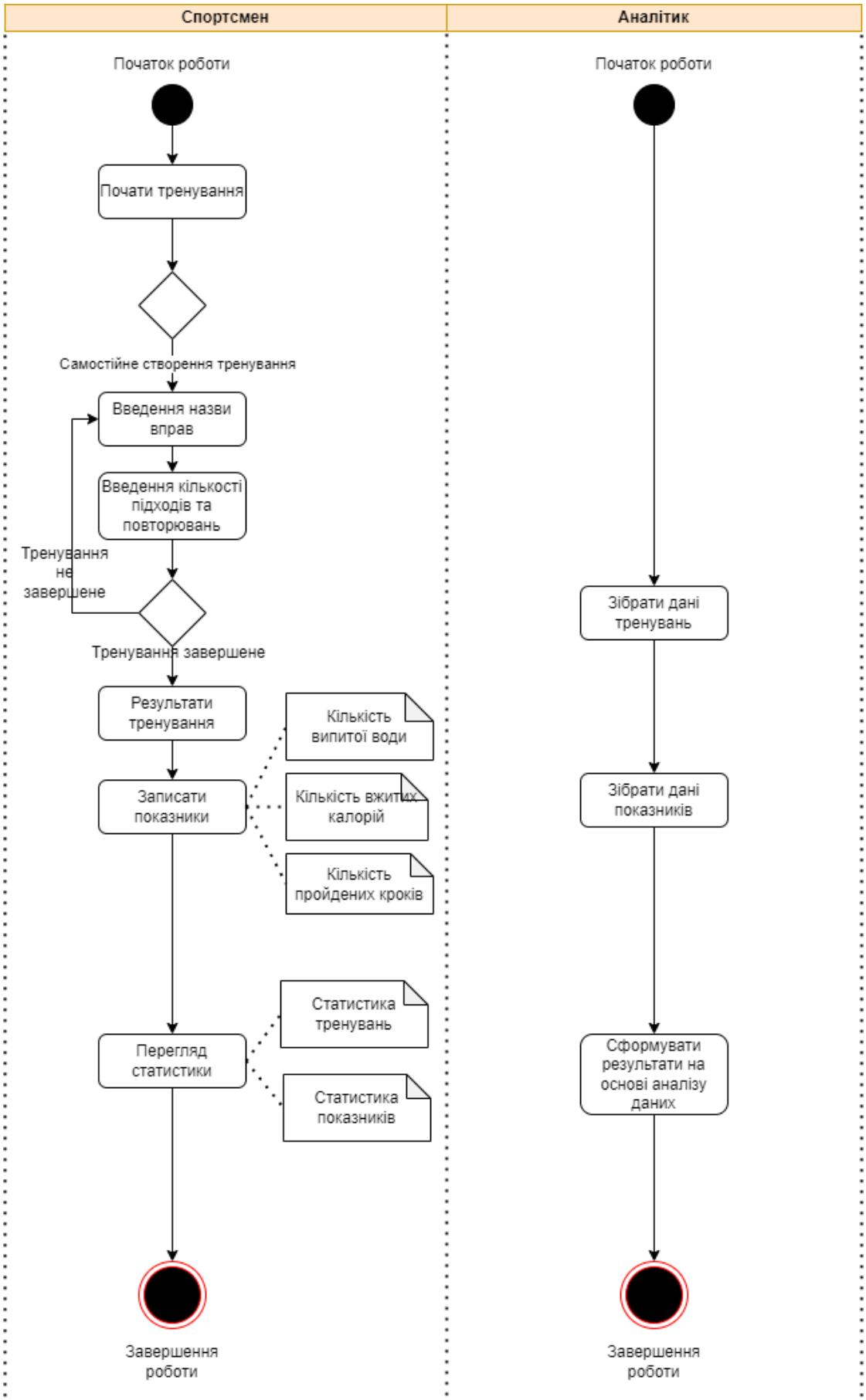


Рис. 5 Діаграма активності

2.3 Огляд інструментів для реалізації завдань Data Mining

Оцінюючи найкращі інструменти інтелектуального аналізу даних, я познайомлю RapidMiner, WEKA, Orange, KNIME та SAS. Користувачі часто використовують кілька програм, оскільки кожен інструмент має різні переваги, які можуть ефективно доповнювати один одного. Крім того, серед інструментів інтелектуального аналізу даних поширена сумісність, що сприяє бездоганній інтеграції. Тим не менш, навіть один комплексний інструмент може запропонувати широкий функціонал, особливо для новачків.

RapidMiner, раніше відомий як YALE («Ще одне навчальне середовище»), є одним із найбільш широко використовуваних інструментів інтелектуального аналізу даних. У опитуванні KDnuggets у 2014 році він був найпопулярнішим перед R. RapidMiner підтримує всі аспекти процесу інтелектуального аналізу даних, включаючи представлення результатів. Його структура складається з трьох основних модулів: RapidMiner Studio, RapidMiner Server і RapidMiner Radoop, кожен з яких призначений для виконання різних методів інтелектуального аналізу даних. Крім того, RapidMiner попередньо обробляє дані, оптимізуючи їх для швидкої подальшої обробки. Примітно, що він чудово підходить для прогнозної аналітики, прогнозування майбутніх подій на основі зібраних даних, що робить його сильним конкурентом серед програмного забезпечення інтелектуального аналізу даних[20].

WEKA, або Waikato Environment for Knowledge Analysis, — це програмне забезпечення з відкритим кодом, розроблене Університетом Ваїкато. Створений на Java, він бездоганно працює в Windows, MacOS і Linux. WEKA, визнана своїми надійними можливостями машинного навчання, підтримує різні завдання інтелектуального аналізу даних, такі як кластеризація, асоціація, регресія та класифікація. Його зручний графічний інтерфейс полегшує використання та пропонує підключення до баз даних SQL для подальшої обробки даних. Сильна сторона WEKA полягає в класифікації,

яка може похвалитися численними алгоритмами, включаючи штучні нейронні мережі, дерева рішень, ID3 і C4.5. Однак він не відповідає іншим методам, таким як кластерний аналіз, пропонуючи лише основні процедури.

Orange, який існує понад два десятиліття, виник як проект Люблянського університету. Спочатку написаний на C++, пізніше він включив Python як мову запитів, пропонуючи потужне поєднання можливостей. Orange — це комплексне програмне забезпечення для інтелектуального аналізу даних, яке славиться своїм зручним інтерфейсом і широкими можливостями для аналізу даних і тексту, а також машинного навчання. Це полегшує такі операції, як класифікація, регресія, кластеризація тощо, поряд із візуальним програмуванням. Примітно, що користувачі вважають Orange привабливим і ефективним завдяки привабливій візуалізації даних і швидкій обробці. Це робить його ідеальним вибором як для новачків, так і для досвідчених користувачів, які шукають інтуїтивно зрозумілий інструмент аналізу даних.

SAS, продукт SAS Institute, є провідним інструментом інтелектуального аналізу даних у бізнес-аналізі, хоч і з високою ціною. Розроблений для великих підприємств, SAS чудово підходить для прогностичної аналітики та інтерактивної візуалізації даних, що робить його ідеальним для вражаючих презентацій. Завдяки комплексному підходу до інтелектуального аналізу даних SAS пропонує високу масштабованість, дозволяючи підвищити продуктивність за допомогою додаткового обладнання або ресурсів. Ця масштабованість робить його потужним рішенням для надійних бізнес-додатків, особливо відданих технічно менш досвідченим користувачам завдяки його інтуїтивно зрозумілому графічному інтерфейсу користувача.

Однак для використання SAS зазвичай потрібна платна ліцензія, хоча існують винятки для державних установ, які можуть пропонувати її безкоштовно або за зниженою ціною. Параметри налаштування дозволяють адаптувати функції інструменту до конкретних потреб, що впливає на загальну ціну. SAS, який переважно використовується у фармацевтичному та банківському секторах, став галузевим стандартом, пропонуючи оптимізовані рішення для бізнес-

аналітики (BI) та веб-майнінгу. Його інтегроване програмне забезпечення BI ще більше покращує його корисність, позиціонуючи SAS як один із найпотужніших інструментів аналізу даних, доступних сьогодні[21].

2.4 Структура джерела інформації для інтелектуального аналізу

Структура джерела інформації для інтелектуального аналізу зазвичай включає кілька ключових компонентів. По-перше, дані збираються з різних джерел, включаючи бази даних, текстові документи, соціальні мережі та веб-сторінки. Після збору дані об’єднуються в уніфікований формат, усуваючи невідповідності та об’єднуючи дані з різних джерел. Потім дані проходять очищення та попередню обробку, яка включає виправлення помилок, обробку відсутніх значень і видалення викидів. Потім виконується вибір функцій і розробка для визначення відповідних функцій і покращення продуктивності моделі. Різні методи аналізу даних, як-от статистичний аналіз і алгоритми машинного навчання, застосовуються для вилучення розуміння та знань із підготовленого набору даних. Прогнозні моделі можна створювати за допомогою алгоритмів машинного навчання для прогнозування або класифікації на основі даних. Результати аналізу інтерпретуються та передаються за допомогою методів візуалізації даних, таких як діаграми та графіки. Зрештою, результати документуються та повідомляються зацікавленим сторонам у письмових звітах, презентаціях або інтерактивних візуалізаціях, інформуючи процеси прийняття рішень. Цей структурований підхід забезпечує ефективний аналіз даних для отримання цінної інформації.

Сховище даних служить уніфікованим, структурованим сховищем інформації, ретельно організованим для полегшення оперативного аналізу та прийняття рішень. Він втілює в собі предметно-орієнтовані, інтегровані та незмінні дані, ретельно розташовані в хронологічному порядку. Цей всеосяжний резервуар надійних даних є основою бізнес-аналітики, що базується на системах

онлайн-обробки даних (OLTP) і системах підтримки прийняття рішень (DSS). Дихотомія між джерелами операційних даних (ODS) і сховищем даних лежить в основі цієї парадигми, оптимізуючи структури даних як для операційних завдань, так і для аналітичних пошуків.

По суті, сховища даних служать наріжним каменем для побудови систем підтримки прийняття рішень, спрямованих на стандартизацію та централізацію всіх відповідних бізнес-даних для ефективного аналізу та звітності. Для цього необхідна гармонійна інтеграція даних із різноманітних внутрішніх і зовнішніх джерел, що забезпечує уніфікованість і доступність. Незалежно від конкретної реалізації, сховища даних мають спільні фундаментальні характеристики: предметна орієнтація, інтеграція, тимчасове посилання та незмінність.

Перед асиміляцією в сховище даних оперативні дані проходять ретельну перевірку, очищення та агрегацію в необхідному обсязі. Ці дані, отримані з оперативних баз даних, проходять суворий процес перевірки, очищення, уніфікації та агрегації, що завершується узгодженим набором даних, що сприяє поглибленому аналізу[25].

На етапі розробки системи сховище даних було ретельно розроблено для полегшення багатогранного аналізу в різних областях. Архітектурний план цього сховища даних зображено на малюнку 1, що забезпечує візуальне представлення його структури.

Щоб задовольнити різноманітні вимоги до даних, у сховищі даних було ретельно розроблено набір таблиць. Сховище даних зображено на Рис. 6.

Ці таблиці служать репозиторіями для зберігання важливих даних, кожна з яких адаптована до конкретних аналітичних потреб і функціональних областей:

- AccountUserDim;
- DateDim;
- ExcersiseDim;
- WaterDim;
- CaloriesDim;
- StepsDim;

- WorkoutFact;
- IndicatorsFact.

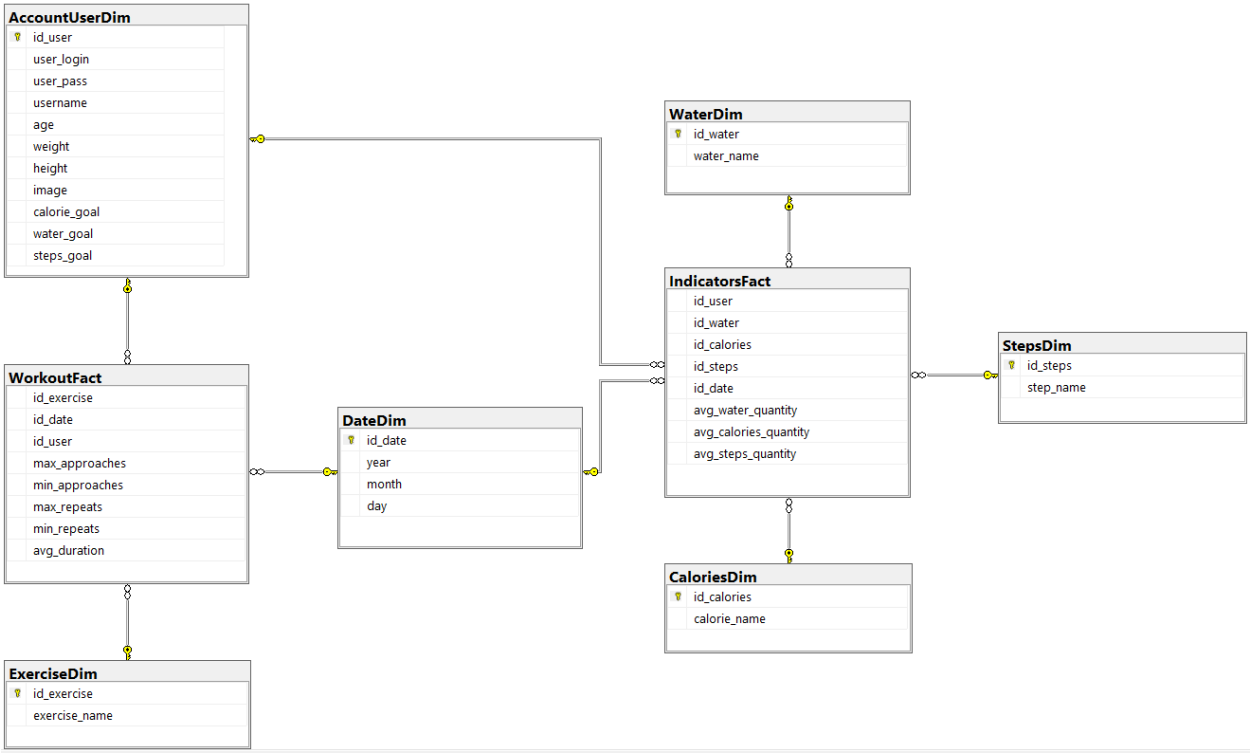


Рис. 6 Сховище даних системи моніторингу та аналізу показників фітнес-асистента

3 РОЗРОБКА СИСТЕМИ

3.1 Логічна модель даних

ERwin — надійний інструмент моделювання даних, який відіграє вирішальну роль у проектуванні, візуалізації та управлінні архітектурами даних. Це особливо корисно для адміністраторів баз даних, архітекторів даних і аналітиків даних, яким потрібно створити чітке представлення зв’язків і структур даних у своїх організаціях[17].

Програмне забезпечення чудово підходить як для логічного, так і для фізичного моделювання даних, дозволяючи користувачам розробляти діаграми сутності та зв’язку (ERD), які ілюструють, як різні сутності, наприклад таблиці, взаємодіють через зв’язки. Ці візуальні представлення покращують розуміння потоку даних і допомагають забезпечити відповідність структури бази даних потребам бізнесу.

Однією з видатних особливостей ERwin є його інтуїтивно зрозумілий графічний інтерфейс, який дозволяє користувачам візуально проектувати моделі даних і керувати ними. Функція перетягування спрощує процес створення та зміни сутностей, атрибутів і зв’язків, роблячи його доступним для користувачів із різними технічними знаннями.

Окрім своїх можливостей проектування, ERwin пропонує інструменти перевірки моделей, щоб переконатися, що моделі даних відповідають встановленим стандартам і найкращим практикам. Ця функція допомагає виявити помилки або невідповідності перед впровадженням, тим самим підвищуючи надійність моделі.

Співпраця є ще одним ключовим аспектом ERwin, оскільки вона дозволяє декільком членам команди працювати над моделями даних одночасно. Включення функцій контролю версій дозволяє відстежувати зміни та підтримує організований робочий процес, полегшуючи командну роботу в проектах архітектури даних.

ERwin також підтримує інтеграцію з різними системами керування базами даних (СУБД). Ця функціональність дозволяє користувачам генерувати сценарії для створення або модифікації баз даних безпосередньо з моделі, спрощуючи перехід від проектування до реалізації та підвищуючи ефективність.

Крім того, програмне забезпечення забезпечує можливості зворотного проектування, дозволяючи користувачам створювати моделі даних з існуючих баз даних. Ця функція є особливо корисною для документування та розуміння застарілих систем, що може допомогти організаціям у модернізації їхніх архітектур даних.

Комплексні функції звітування та документування також є частиною пропозицій ERwin, що дозволяє користувачам створювати детальні звіти, що описують модель даних, включаючи інформацію про сутності, атрибути та зв’язки. Ця документація є цінною для цілей відповідності та управління, пропонуючи чіткий огляд структури даних.

Нарешті, ERwin дотримується галузевих стандартів моделювання даних, таких як ANSI/SPARC і IDEF1X, гарантуючи, що створені моделі сумісні з найкращими практиками в архітектурі даних[17].

Таким чином, ERwin — це потужний і універсальний інструмент, який спрощує процес моделювання даних, сприяє співпраці та покращує точність дизайну бази даних. Його широкі можливості роблять його ідеальним вибором для організацій, які прагнуть ефективно керувати своїми даними та підтримувати прийняття обґрунтованих рішень.

Логічна модель системи представлена на рис. 7.

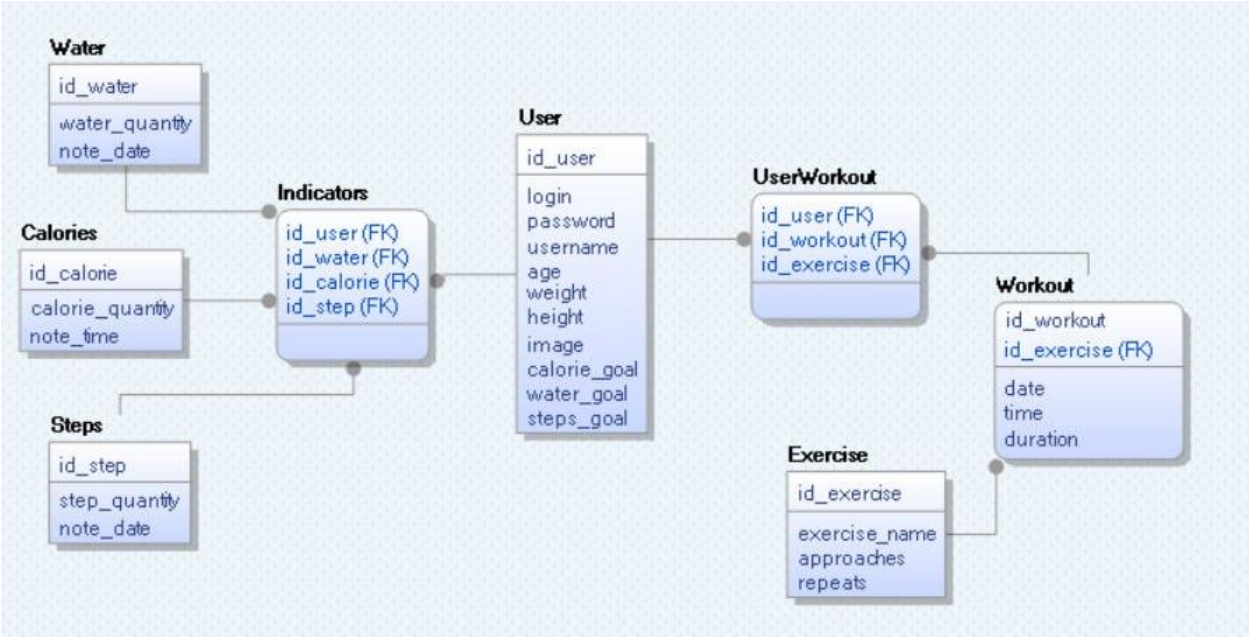


Рис. 7 ER-діаграма

Логічна модель складається з кількох ключових сутностей, які представляють різні аспекти даних і дій користувача. Сутність «User» містить важливу інформацію про особу, а сутність «Indicators» відстежує різні показники, пов’язані з продуктивністю користувача. Сутність «Workout» надає детальну інформацію про конкретні тренування, а «UserWorkout» підключає користувачів до їхніх тренувань. Крім того, сутність «Excercise» описує виконані вправи, а сутності «Water», «Calories» та «Steps» записують інформацію про споживання води, споживання калорій і кількість пройдених кроків відповідно.

Об’єкт «User» містить такі атрибути, як ідентифікатор користувача, облікові дані для входу, пароль, ім’я користувача, вік, вага, зріст, зображення профілю та індивідуальні цілі щодо споживання калорій, споживання води та кількості кроків.

Сутність «Indicators» має атрибути, які складаються з ідентифікатора користувача, ідентифікатора калорій, ідентифікатора води та ідентифікатора кроку, які служать посиланнями на відповідні точки даних.

Для сутності "Workout" його атрибути включають ідентифікатор тренування, ідентифікатор вправи, дату, час і тривалість сеансу тренування. Сутність «UserWorkout» пов’язує користувачів із їхніми тренуваннями за допомогою атрибутів ID користувача, ID тренування та ID вправи.

Що стосується вправ, сутність «Excercise» складається з таких атрибутів, як ID вправи, назва, кількість підходів і повторень.

Об’єкт «Water» фіксує інформацію про водозабір, що характеризується ідентифікатором води, спожитою кількістю та датою входу. Сутність «Calories» відстежує споживання калорій за допомогою атрибутів для ідентифікатора калорій, кількості та відповідної дати. Нарешті, сутність «Steps» записує кількість зроблених кроків з атрибутами, включаючи ідентифікатор кроку, кількість і дату нотатки.

Відносини між цими суб’єктами також важливі. Сутність «User» пов’язана як з «Indicators», так і з «UserWorkout», ілюструючи, як дані користувача пов’язані з показниками продуктивності та тренуваннями. Сутність «Indicators», у свою чергу, підключається до сутностей «Water», «Calories» та «Steps», що дозволяє комплексно відстежувати показники здоров’я користувача. Крім того, сутність «Workour» пов’язана як з «UserWorkout», так і з «Excercise», що відображає зв’язок між конкретними тренуваннями та вправами, які виконуються під час цих сеансів.

3.2 Вибір системи управління базою даних та її реалізація

Система керування базами даних (СУБД) — це програмний інструмент, призначений для полегшення створення баз даних, керування та маніпулювання ними. Діючи як посередник між користувачами та базою даних, СУБД покращує організацію, зберігання, пошук і загальне управління даними.

Ключові компоненти СУБД включають механізм бази даних, який є основною службою, що відповідає за зберігання та керування даними, обробку

запитів і транзакцій, забезпечуючи при цьому цілісність і безпеку даних. Схема бази даних визначає структуру даних у базі даних, окреслюючи такі елементи, як таблиці, поля, зв'язки та обмеження. Більшість СУБД використовують мову структурованих запитів (SQL) як мову запитів, що дозволяє користувачам виконувати важливі операції, такі як запити, оновлення, вставка та видалення даних. Крім того, багато СУБД забезпечують інтерфейс користувача, який може бути графічним або командним рядком, для спрощення взаємодії з базою даних. Різні моделі даних, включаючи реляційну, об'єктно-орієнтовану та документ орієнтовану, визначають, як дані структуруються та як доступ до них здійснюється[19].

Функціональні можливості СУБД охоплюють різні важливі завдання. Визначення даних дозволяє користувачам створювати та змінювати структуру бази даних, тоді як маніпуляції даними полегшують такі операції, як вставка, оновлення, видалення та отримання даних за допомогою мов запитів. Механізми безпеки даних контролюють доступ до інформації, гарантуючи, що лише авторизовані користувачі можуть переглядати або змінювати дані. Щоб підтримувати точність і послідовність, СУБД забезпечує цілісність даних за допомогою правил і обмежень, таких як первинні та зовнішні ключі. Крім того, він надає інструменти резервного копіювання та відновлення для захисту від втрати чи пошкодження даних. Управління паралелізмом є ще однією важливою функцією, керуючи одночасним доступом користувачів до бази даних, щоб запобігти втручанню в транзакції.

Існує кілька типів СУБД. Реляційна СУБД (RDBMS) організовує дані в таблицях із рядками та стовпцями та використовує SQL для обробки даних, із популярними прикладами, такими як MySQL, PostgreSQL і Oracle. СУБД NoSQL обслуговує неструктуровані або напівструктуровані дані, пропонуючи гнучкість і масштабованість, зокрема MongoDB, Cassandra та Redis. Об'єктно-орієнтована СУБД (OODBMS) містить принципи об'єктно-орієнтованого програмування, що дозволяє зберігати складні типи даних як об'єкти, наприклад у db4o та ObjectDB. Крім того, ієрархічні та мережеві СУБД представляють старіші типи баз даних,

які зберігають дані в деревоподібних структурах або допускають численні зв’язки, наприклад IMS IBM та Integrated Data Store (IDS).

Підсумовуючи, СУБД відіграє вирішальну роль в ефективному управлінні та маніпулюванні великими обсягами даних. Він забезпечує системний підхід для користувачів і програм для взаємодії з базами даних, забезпечуючи цілісність даних, безпеку та доступність.

На основі оцінки цих факторів було прийнято рішення обрати СУБД для розробки програмного забезпечення фітнес-помічника MS SQL Server.

Рішення обрати MS SQL Server як систему керування базами даних для проекту можна пояснити кількома переконливими факторами, які відповідають вимогам і цілям проекту.

Перш за все, MS SQL Server пропонує надійну продуктивність і масштабованість, що робить його придатним для обробки великих обсягів даних і підтримки великих транзакційних навантажень. Ця можливість має важливе значення для додатків, які потребують швидкого пошуку та обробки даних, гарантуючи, що користувачі відчувають мінімальну затримку під час операцій.

Іншим важливим фактором є розширені функції безпеки MS SQL Server. Завдяки вбудованим механізмам шифрування, автентифікації та контролю доступу він забезпечує безпечне середовище для конфіденційних даних. Це особливо важливо для організацій, які повинні дотримуватися суворих нормативних стандартів і захищати інформацію клієнтів.

Крім того, MS SQL Server легко інтегрується з різними продуктами та службами Microsoft, такими як Azure, Visual Studio та Power BI. Ця сумісність забезпечує плавний процес розробки та спрощене розгортання, дозволяючи командам використовувати існуючі технології та навички Microsoft[21].

Зручний інтерфейс і комплексні інструменти, надані MS SQL Server, сприяють ефективному управлінню та розробці бази даних. SQL Server

Management Studio (SSMS) пропонує інтуїтивно зрозумілі функції для запитів, керування та моніторингу баз даних, підвищуючи продуктивність як для розробників, так і для адміністраторів баз даних.

Крім того, MS SQL Server підтримує різні типи даних і структури, включаючи реляційні дані, дані JSON і XML. Ця універсальність дає змогу працювати з різноманітними моделями даних, що є корисним для проектів, які вимагають гнучкого зберігання та доступу до даних.

Нарешті, широка підтримка спільноти та документація, що оточує MS SQL Server, надають цінні ресурси для усунення несправностей, передових практик та оптимізації. Ця мережа підтримки допомагає гарантувати швидке й ефективне вирішення будь-яких проблем, які виникають під час розробки чи розгортання.

Підсумовуючи, рішення обрати MS SQL Server було обумовлено його продуктивністю, функціями безпеки, сумісністю з технологіями Microsoft, зручними інструментами керування, універсальністю обробки різних типів даних і доступністю підтримки спільноти. Ці атрибути роблять його придатним вибором для проектів, які потребують надійного та ефективного рішення для керування базами даних.

Під час проєктування таблиць у фізичній моделі основою слугували сутності з логічної моделі. Атрибути цих сутностей були використані для розробки структури таблиці. Отриману схему бази даних зображено на рис. 8.

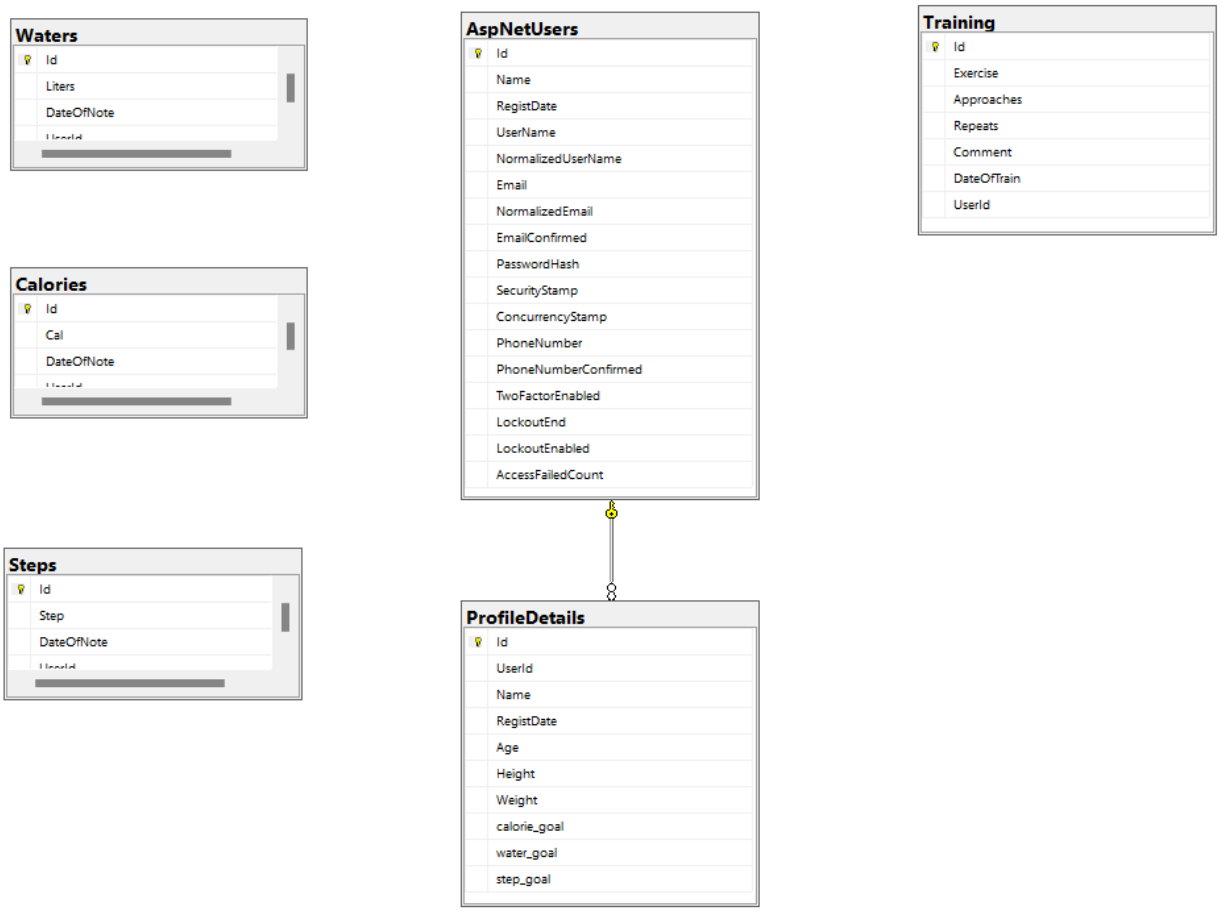


Рис. 8 База даних системи

3.3 Архітектура програмного забезпечення

У веб-додатках запускаються три основні шари: інтерфейс користувача, бізнес-логіка та доступ до даних. У такій архітектурі користувачі дізнаються запити через інтерфейс, який взаємодіє з бізнес-логікою. Ця логіка, у свою чергу, може відкрити доступ до даних для обробки запитів. Важливо, щоб інтерфейс користувача не проводив запити після до шару доступу до даних або взаємодіяв з функціями іншим способом збереження. Аналогічно, бізнес-логіка повинна взаємодіяти з функціями збереження лише через шар доступу до даних, що чітко виконує обов'язки кожного з цих шарів.

З цього можна зробити висновок, що багатошарова архітектура є оптимальним вибором для цієї системи. Однак традиційний багатошаровий підхід має свій недолік: обробка залежностей під час компіляції відбувається

вниз. Це означає, що інтерфейс користувача залежить від бізнес-логіки, яка, у свій час, залежить від шару доступу до даних. В результаті бізнес-логіка, що містить ключові функції програми, залежить від деталей реалізації доступу до даних, а також від наявності самої бази даних. Таке взаємозалежне структурування ускладнює тестування бізнес-логіки, після чого воно часто потребує тестової бази даних.

Надалі буде представлено чотиришарову архітектуру даного програмного забезпечення, що ілюструє цю концепцію (Рис. 9).

Рис. 9 Багатошарова архітектура веб-сайту

Візуалізація системи демонструє, що вона побудована на основі чотирирівневої багатошарової архітектури, яка складається з презентаційного, бізнесового, рівня збереження даних та баз даних. Інформація про збільшення передається від одного рівня до наступного.

Перший рівень представляє інтерфейс користувача, який відображається на екрані комп'ютера або смартфона.

Другий рівень містить основну функціональну програму, з якою взаємодіє користувач. Тут він може вводити свої дані, підтримувати тренування та отримувати персоналізовані звіти.

На третьому рівні відновлення передачі та збереження інформації від користувачів до баз даних.

Четвертий рівень складається з бази даних, в якій зберігається інформація користувача. Ця структура забезпечує ефективну організацію даних та їх збереження в системі.

3.4 Використання 1-Rule для класифікації

1-Rule, також відомий як алгоритм OneR, є простим та інтуїтивно зрозумілим алгоритмом класифікації, який використовується в машинному навчанні. Незважаючи на свою простоту, алгоритм 1-Rule може надати цінну інформацію та служити основою для більш складних моделей класифікації.

Основний принцип алгоритму 1-Rule передбачає вибір одного атрибута (або функції) з набору даних і його використання для створення правила класифікації. Цей вибраний атрибут, як правило, демонструє найсильніший зв’язок із цільовим класом у наборі даних.

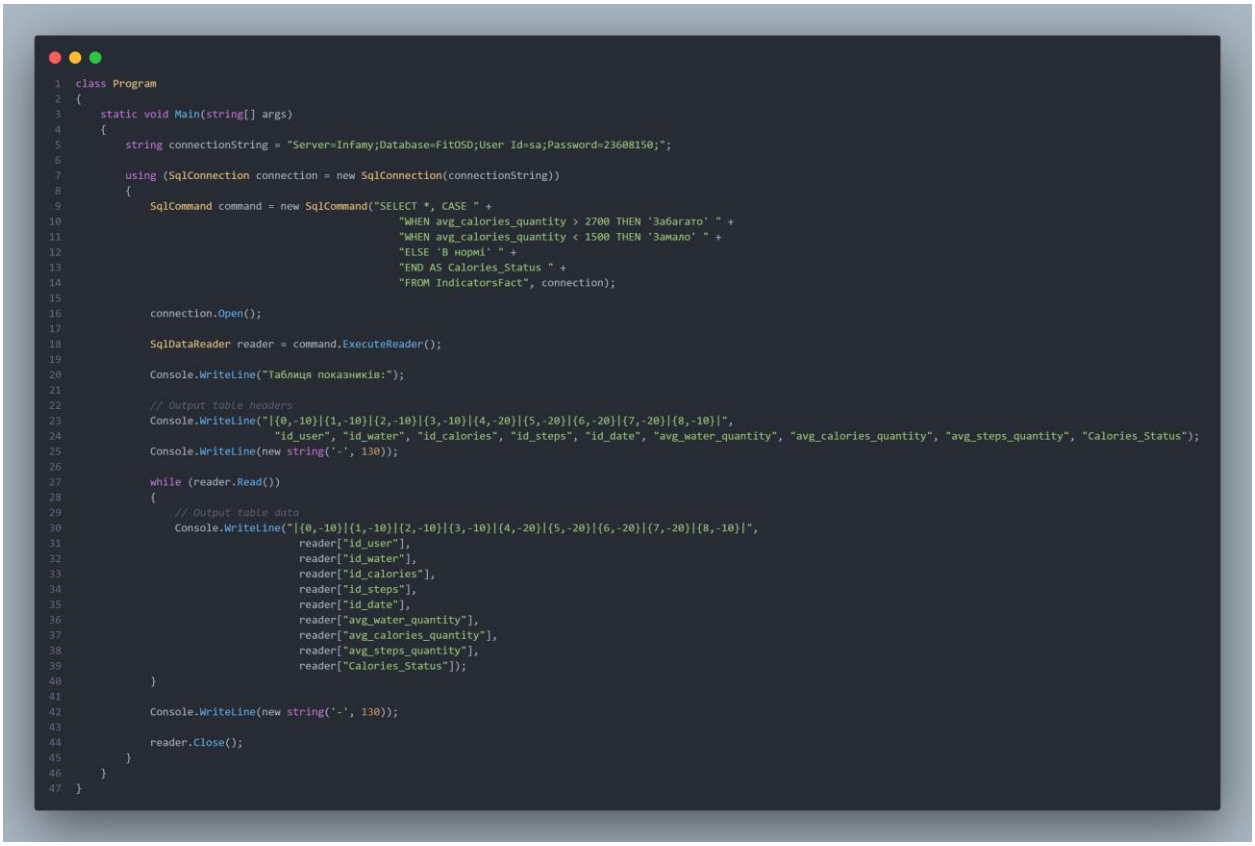
Після вибору атрибута алгоритм визначає найпоширеніший клас (або категорію) для кожного унікального значення вибраного атрибута. Потім на основі цих асоціацій створюється правило, яке призначає найпоширеніший клас екземплярам, які відповідають певному значенню атрибута.

Наприклад, якщо ми класифікуємо квіти на основі їхньої довжини пелюсток, алгоритм 1-правила може визначити, що якщо довжина пелюстки менше 2,5 см, квітка, швидше за все, належить до класу "setosa"[24].

Незважаючи на свою простоту, алгоритм 1-Rule може забезпечити швидку та інтерпретовану базову лінію для завдань класифікації. Однак він не завжди може фіксувати складні зв’язки в даних, а його продуктивність може бути нижчою від більш просунутих алгоритмів на певних наборах даних.

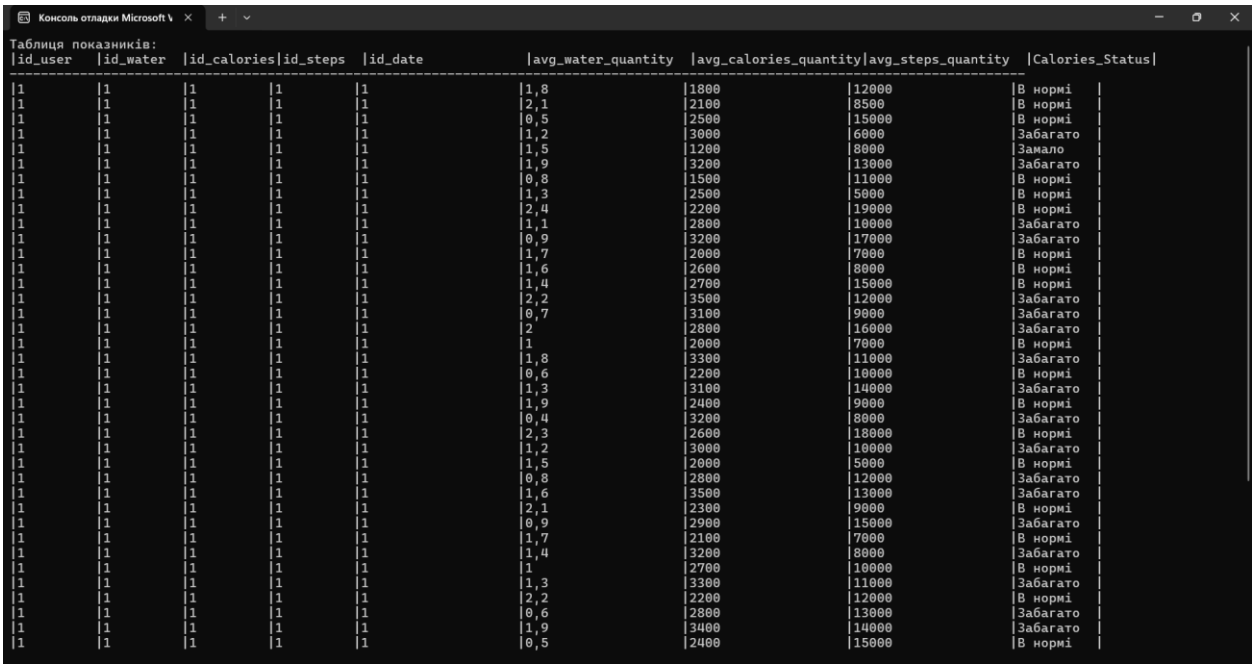
Щоб вирішити проблему класифікації, ми використали Microsoft Visual Studio та дані з SD, що містяться в Microsoft SQL Management Studio. Написано програму та використано алгоритм 1R.

На малюнку 10 показано запит на отримання спожитих калорій, води та кроків із таблиці IndicatorsFact для кожного користувача.



```
class Program
{
    static void Main(string[] args)
    {
        string connectionString = "Server=Infamy;Database=FitOSD;User Id=sa;Password=23608150;";

        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            SqlCommand command = new SqlCommand("SELECT *, CASE " +
                                                "WHEN avg_calories_quantity > 2700 THEN 'Забагато' " +
                                                "WHEN avg_calories_quantity < 1500 THEN 'Замало' " +
                                                "ELSE 'В нормі' " +
                                                "END AS Calories_Status " +
                                                "FROM IndicatorsFact", connection);

            connection.Open();

            SqlDataReader reader = command.ExecuteReader();

            Console.WriteLine("Таблиця показників:");

            // Output table headers
            Console.WriteLine("|{0,-10}|{1,-10}|{2,-10}|{3,-10}|{4,-20}|{5,-20}|{6,-20}|{7,-20}|{8,-10}|",
                              "id_user", "id_water", "id_calories", "id_steps", "id_date", "avg_water_quantity", "avg_calories_quantity", "avg_steps_quantity", "Calories_Status");
            Console.WriteLine(new string('-', 130));

            while (reader.Read())
            {
                // Output table data
                Console.WriteLine("|{0,-10}|{1,-10}|{2,-10}|{3,-10}|{4,-20}|{5,-20}|{6,-20}|{7,-20}|{8,-10}|",
                                  reader["id_user"],
                                  reader["id_water"],
                                  reader["id_calories"],
                                  reader["id_steps"],
                                  reader["id_date"],
                                  reader["avg_water_quantity"],
                                  reader["avg_calories_quantity"],
                                  reader["avg_steps_quantity"],
                                  reader["Calories_Status"]);
            }

            Console.WriteLine(new string('-', 130));

            reader.Close();
        }
    }
}
```

Рис. 10 Запит для отримання спожитих калорій, води та кроків із таблиці IndicatorsFact для кожного користувача



```
Консоль отладки Microsoft \
Таблиця показників:
|id_user   |id_water  |id_calories|id_steps  |id_date            |avg_water_quantity  |avg_calories_quantity|avg_steps_quantity  |Calories_Status|
-----------------------------------------------------------------------------------------------------------------------------------------------
|1         |1         |1         |1         |1                  |1,8                 |1800                 |12000               |В нормі        |
|1         |1         |1         |1         |1                  |2,1                 |2100                 |8500                |В нормі        |
|1         |1         |1         |1         |1                  |0,5                 |2500                 |15000               |В нормі        |
|1         |1         |1         |1         |1                  |1,2                 |3000                 |6000                |Забагато       |
|1         |1         |1         |1         |1                  |1,5                 |1200                 |8000                |Замало         |
|1         |1         |1         |1         |1                  |1,9                 |3200                 |13000               |Забагато       |
|1         |1         |1         |1         |1                  |0,8                 |1500                 |11000               |В нормі        |
|1         |1         |1         |1         |1                  |1,3                 |2500                 |5000                |В нормі        |
|1         |1         |1         |1         |1                  |2,4                 |2200                 |19000               |В нормі        |
|1         |1         |1         |1         |1                  |1,1                 |2800                 |10000               |Забагато       |
|1         |1         |1         |1         |1                  |0,9                 |3200                 |17000               |Забагато       |
|1         |1         |1         |1         |1                  |1,7                 |2000                 |7000                |В нормі        |
|1         |1         |1         |1         |1                  |1,6                 |2600                 |8000                |В нормі        |
|1         |1         |1         |1         |1                  |1,4                 |2700                 |15000               |В нормі        |
|1         |1         |1         |1         |1                  |2,2                 |3500                 |12000               |Забагато       |
|1         |1         |1         |1         |1                  |0,7                 |3100                 |9000                |Забагато       |
|1         |1         |1         |1         |1                  |2                   |2800                 |16000               |Забагато       |
|1         |1         |1         |1         |1                  |1                   |2000                 |7000                |В нормі        |
|1         |1         |1         |1         |1                  |1,8                 |3300                 |11000               |Забагато       |
|1         |1         |1         |1         |1                  |0,6                 |2200                 |10000               |В нормі        |
|1         |1         |1         |1         |1                  |1,3                 |3100                 |14000               |Забагато       |
|1         |1         |1         |1         |1                  |1,9                 |2400                 |9000                |В нормі        |
|1         |1         |1         |1         |1                  |0,4                 |3200                 |8000                |Забагато       |
|1         |1         |1         |1         |1                  |2,3                 |2600                 |18000               |В нормі        |
|1         |1         |1         |1         |1                  |1,2                 |3000                 |10000               |Забагато       |
|1         |1         |1         |1         |1                  |1,5                 |2000                 |5000                |В нормі        |
|1         |1         |1         |1         |1                  |0,8                 |2800                 |12000               |Забагато       |
|1         |1         |1         |1         |1                  |1,6                 |3500                 |13000               |Забагато       |
|1         |1         |1         |1         |1                  |2,1                 |2300                 |9000                |В нормі        |
|1         |1         |1         |1         |1                  |0,9                 |2900                 |15000               |Забагато       |
|1         |1         |1         |1         |1                  |1,7                 |2100                 |7000                |В нормі        |
|1         |1         |1         |1         |1                  |1,4                 |3200                 |8000                |Забагато       |
|1         |1         |1         |1         |1                  |1                   |2700                 |10000               |В нормі        |
|1         |1         |1         |1         |1                  |1,3                 |3300                 |11000               |Забагато       |
|1         |1         |1         |1         |1                  |2,2                 |2200                 |12000               |В нормі        |
|1         |1         |1         |1         |1                  |0,6                 |2800                 |13000               |Забагато       |
|1         |1         |1         |1         |1                  |1,9                 |3400                 |14000               |Забагато       |
|1         |1         |1         |1         |1                  |0,5                 |2400                 |15000               |В нормі        |
```

Рис. 11 Результат класифікації за станом калорійності

Дані розраховані з 01.10.2023-01.11.2023

Дані стану калорій поділяються на три класи:

- занадто багато;

- не вистачає;
- у нормі.

Ґрунтуючись на цій вибірці, ми можемо зробити висновок, що більшість людей переїдають зайві калорії, ніж ті, хто недоїдає або нормально харчується.

3.5 Вибір інструментарію для створення програмного забезпечення

При виборі інструментів для розробки програм зазвичай розглядаються кілька ключових технологій і програмних додатків, кожна з яких служить окремим цілям у процесі розробки.

C# служить основною мовою програмування для створення програми. C#, відомий своєю універсальністю та чіткою типізацією, є ідеальним вибором для розробки надійних програм, які можуть працювати на різних платформах. Його багатий набір функцій дозволяє розробникам писати зрозумілий код, який зручно підтримувати, що підвищує продуктивність і співпрацю.

ASP.NET вибрано як веб-платформу для полегшення розробки динамічних веб-додатків. Він дозволяє розробникам створювати інтерактивні веб-інтерфейси та надає широкий спектр вбудованих функцій для безпеки, керування сеансами та продуктивності програм. За допомогою ASP.NET розробники можуть використовувати архітектуру MVC (Model-View-Controller), яка сприяє організованому коду та поділу проблем, полегшуючи керування складними програмами.

MS SQL Server є обраною системою керування базою даних для зберігання та отримання даних програми. Відомий своєю надійністю, масштабованістю та функціями безпеки, MS SQL Server надає потужні інструменти для керування даними, включаючи розширені можливості запитів через SQL. Це дозволяє розробникам ефективно обробляти транзакції даних і підтримувати цілісність даних.

Photoshop CS6 входить до набору інструментів для графічного дизайну та розробки UI/UX. Це програмне забезпечення дозволяє дизайнерам створювати візуально привабливі інтерфейси користувача, оптимізувати зображення та створювати привабливу графіку, яка покращує загальну естетику програми. Використовуючи Photoshop, розробники можуть гарантувати, що візуальні елементи програми відповідають бажаному досвіду користувача.

Для інтерфейсної розробки HTML, CSS і JavaScript є важливими технологіями. HTML забезпечує структуру веб-сторінок, дозволяючи розробникам створювати семантичний і доступний вміст. CSS використовується для стилізації веб-сторінок, дозволяючи налаштовувати макети, кольори та шрифти для досягнення цілісного дизайну. JavaScript додає веб-програмі інтерактивність і динамічність, підвищуючи залучення користувачів і загальну функціональність.

Таким чином, поєднання C#, ASP.NET, MS SQL Server, Photoshop CS6, HTML, CSS і JavaScript створює комплексне середовище розробки. Цей набір інструментів підтримує різні аспекти програми, від серверної логіки та керування базами даних до зовнішнього дизайну та взаємодії з користувачем, забезпечуючи комплексний та ефективний процес розробки додатків.

3.6 Використання методу Наївного Байєса

Метод Naive Bayes — це простий, але потужний алгоритм, який використовується в машинному навчанні для класифікаційних завдань. Він ґрунтується на теоремі Байєса, яка кількісно визначає ймовірність події на основі попередніх знань про відповідні умови. Незважаючи на свою простоту, Naive Bayes може давати вражаючі результати, особливо в сценаріях класифікації тексту, таких як виявлення спаму та аналіз настроїв[27].

Ось як працює Наївний Байєс.

Теорема Байєса лежить в її основі, виражаючи ймовірність події, якщо відбулася інша подія. У класифікації він обчислює ймовірність мітки класу з огляду на спостережувані особливості.

Наивний Байєс припускає, що ознаки є умовно незалежними з урахуванням мітки класу. Це спрощує розрахунки, хоча на практиці це часто не зовсім вірно.

Під час навчання алгоритм обчислює попередню ймовірність кожного класу та ймовірність кожної функції, заданої кожному класу, використовуючи навчальні дані.

Для передбачення він обчислює апостеріорну ймовірність кожного класу з огляду на спостережувані особливості за допомогою теореми Байєса, вибираючи клас із найвищою ймовірністю як прогноз.

Оцінка простого класифікатора Байєса передбачає оцінку таких показників, як точність, точність, запам’ятовування та оцінка F1 шляхом порівняння прогнозованих міток класу з справжніми мітками класу в даних тесту.

Наївні байєсівські класифікатори цінуються за їх простоту, швидкість і ефективність, особливо коли припущення про незалежність досить добре. Однак вони можуть не працювати так ефективно, як складніші алгоритми на наборах даних із висококорельованими характеристиками.

На малюнку 12 показано структуру куба (побудованого за допомогою SSAS), на основі якого в майбутньому буде виконуватися інтелектуальний аналіз даних. Цей куб побудований для вирішення завдань аналізу використаних калорій[28].

Програмна реалізація алгоритму: Алгоритм реалізовано за допомогою засобів SQL та C#.

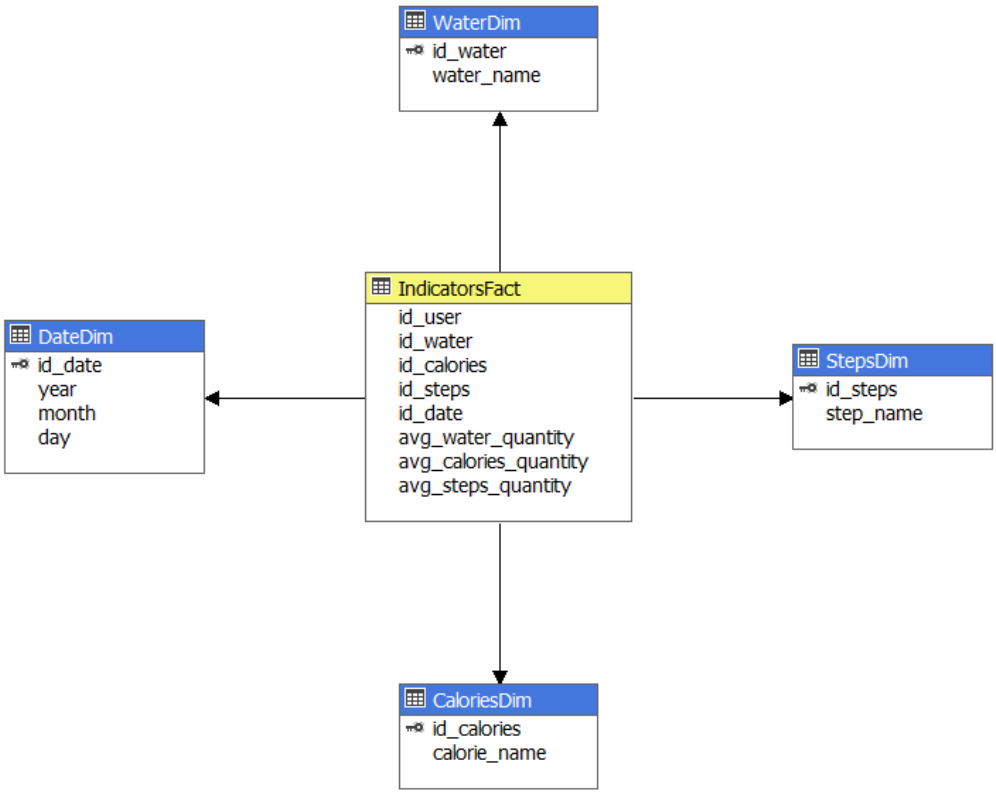


Рис. 12 Будова куба

Реалізація розв’язання задачі класифікації методом Наївного Байєса з використанням засобів SSAS.

Створіть структуру інтелектуального аналізу за допомогою майстра інтелектуального аналізу даних. Виберіть таблицю вимірювань, за якою буде відображатися та прогнозуватися зміна фактичних показників. У вирішуваній задачі цією таблицею буде CaloriesDim, в якій зберігається інформація про існуючі продукти з їх характеристиками:

Далі потрібно визначити параметри вибраних атрибутів

- input - вхідна зміна, яка суттєво впливає на хід досліджуваного процесу (Продукт, Назва);
- predict - зміна, значення якої будуть передбачені (Середня кількість використаних калорій).

Рис. 13. Вигляд моделі інтелектуального аналізу Naive Bayes in Mining Structure

Рис. 14 Метод результатів Наївний Байєс у структурі видобутку

На завершення можна сказати наступне: згідно з отриманими результатами можна зробити висновок, що SSAS підтверджує правила, які були отримані за допомогою алгоритму внутрішньої розробки

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Вимоги до апаратного та програмного забезпечення

Діаграма розгортання виконує важливу роль у відображенні загальної топології розподіленої системи, наочно демонструючи різні, як артефакти розміщуються на окремих вузлах. Основним призначенням цієї діаграми є візуалізація фізичної структури системи, що показує, яким чином програмні компоненти розгортаються на різних вузлах.

На діаграмах вузли представлені у вигляді прямокутників або значків, які символізують виконання апаратних пристроїв або середовища, в яких функціонує система. Програмні артефакти, такі як програми, модулі та служби, зображені у вигляді прямокутників або піктограм, що розміщуються в рамках вузлів для відображення їхнього розгортання.

Зв’язки розгортання, включаючи асоціації, відносини та інші види взаємодій, ілюструють зв’язки між компонентами програмного забезпечення та апаратними вузлами. Ці зв’язки допомагають зрозуміти, як компоненти спілкуються і співпрацюють у системній інфраструктурі[17].

У серверній частині системи обробляються запити, забезпечуються взаємодія з базою даних, обробляються дані, введені користувачем, а також створюється структура веб-додатків. Клієнт представляє собою користувача, який взаємодіє з додатком через веб-браузер. Сервер бази даних є програмним ресурсом, призначеним для зберігання даних у системі.

На діаграмі, зображеній на рисунку 15, представлена структура розгортання даної системи, яка реалізує клієнт-серверний підхід на двох окремих серверах.

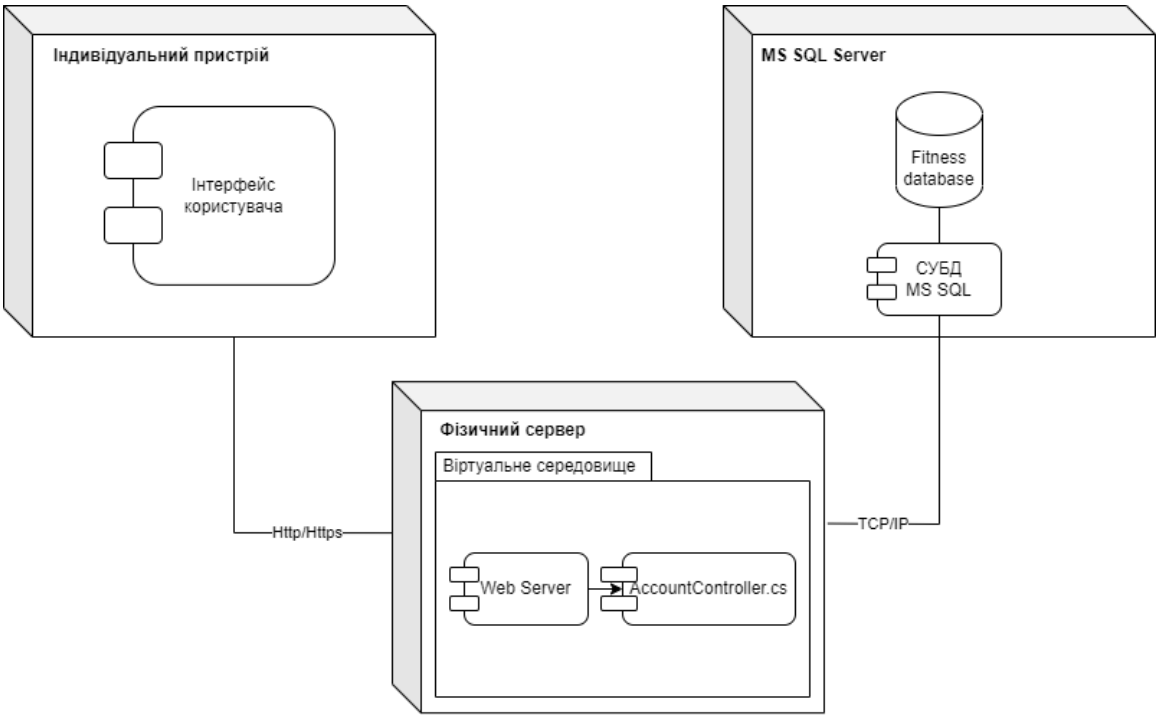


Рис. 15 Діаграма розгортання

Апаратні вимоги які необхідні бути у комп'ютера для функціонування програмного забезпечення:

- процесор із принаймні двома ядрами, наприклад Intel Core i3 або AMD Ryzen 3l
- оперативна пам'ять 4 ГБ;
- SSD 10 ГБ або більше.

Вимоги до програмного забезпечення

Інструменти розробки: Інтегроване середовище розробки (IDE): Приклади включають Visual Studio для розробки на C# або будь-яку іншу бажану IDE, сумісну з мовами програмування, які використовуються.

Система контролю версій: Git для керування вихідним кодом і співпраці.

Фреймворки та бібліотеки: .NET Framework або ASP.NET для веб-додатків, створених на C#.

Фреймворки JavaScript, такі як Vue.js або React для інтерфейсної розробки. Фреймворки CSS, такі як Bootstrap, для адаптивного дизайну.

Система управління базами даних (СУБД): MS SQL Server для ефективного керування реляційними даними.

Інструменти для розробки баз даних і керування ними, наприклад SQL Server Management Studio (SSMS).

Веб-браузер: Останні версії браузерів, як-от Google Chrome, Mozilla Firefox або Microsoft Edge, для тестування веб-додатків.

Додаткове програмне забезпечення: Графічне програмне забезпечення, наприклад Photoshop CS6, для розробки елементів інтерфейсу користувача та графіки. Будь-які необхідні бібліотеки або залежності для певних функцій програми.

Програмне забезпечення безпеки: антивірусне програмне забезпечення та брандмауер для захисту системи від зловмисного програмного забезпечення та несанкціонованого доступу.

4.2 Тестування системи

Запустивши систему, бачимо головну сторінку програмного забезпечення (рис. 16-17).

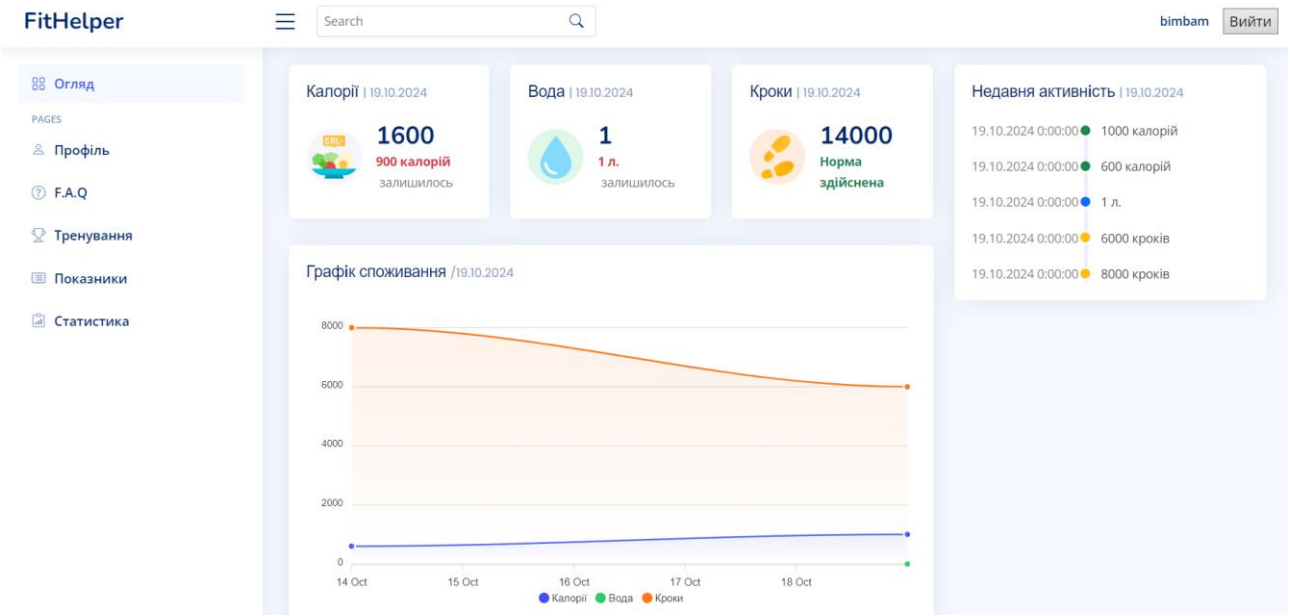


Рис. 16 Головна сторінка

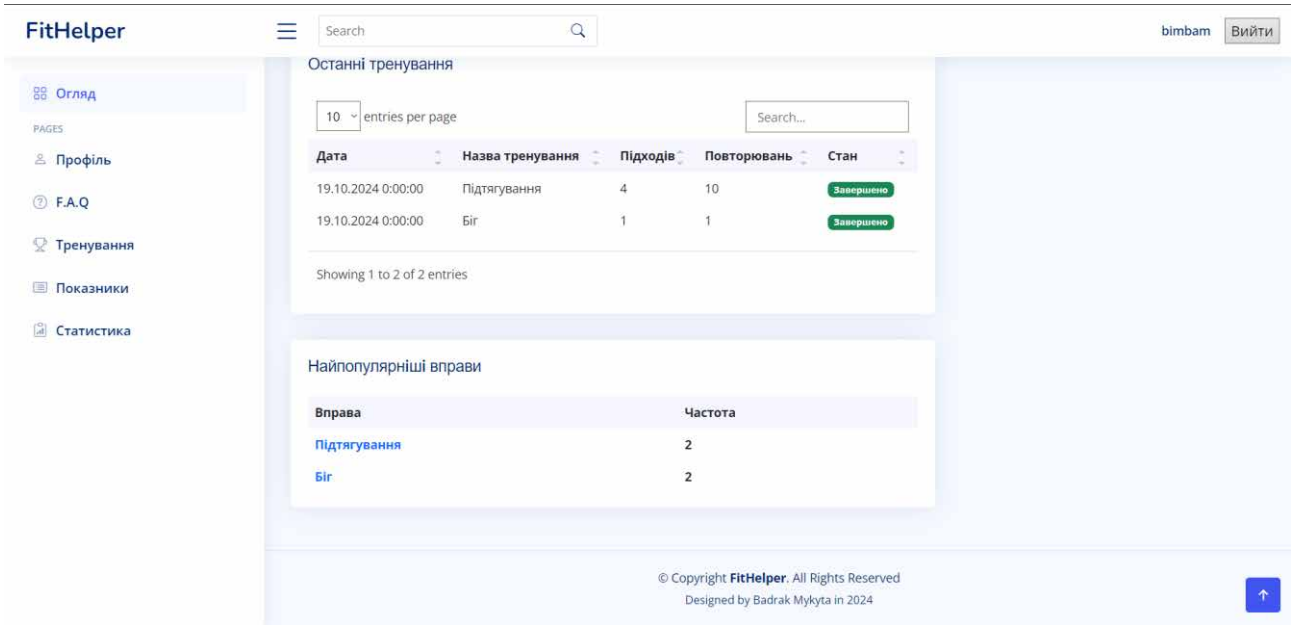


Рис. 17 Головна сторінка

На головній сторінці програми користувач має можливість відразу ознайомитися з ключовими показниками, такими як кількість спожитих калорій, об’єм випитої води та кількість пройдених кроків за поточний день. Ці дані також відображають, скільки ще залишилося до досягнення індивідуальної норми, що робить процес контролю за здоров'ям зручним.

Окрім цього, користувач може переглядати власний графік споживання, а також історію своїх тренувань та досягнень. Важливою частиною інтерфейсу є відображення найбільш популярних вправ, які користувач виконує.

На малюнку 18 показана форма авторизації, яку потрібно заповнити при першому вході в систему.

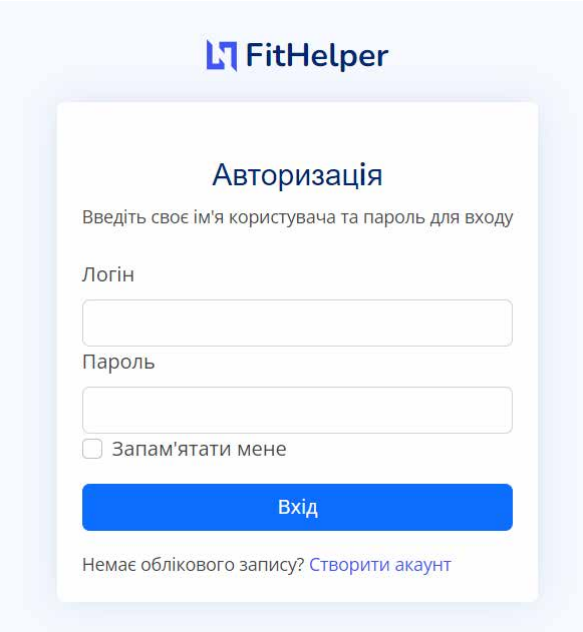


Рис. 18 Форма авторизації

Наступним етапом є реєстрація в системі, що передбачає створення профілю користувача та заповнення його особистими даними. Цю форму можна побачити на рисунку 19.

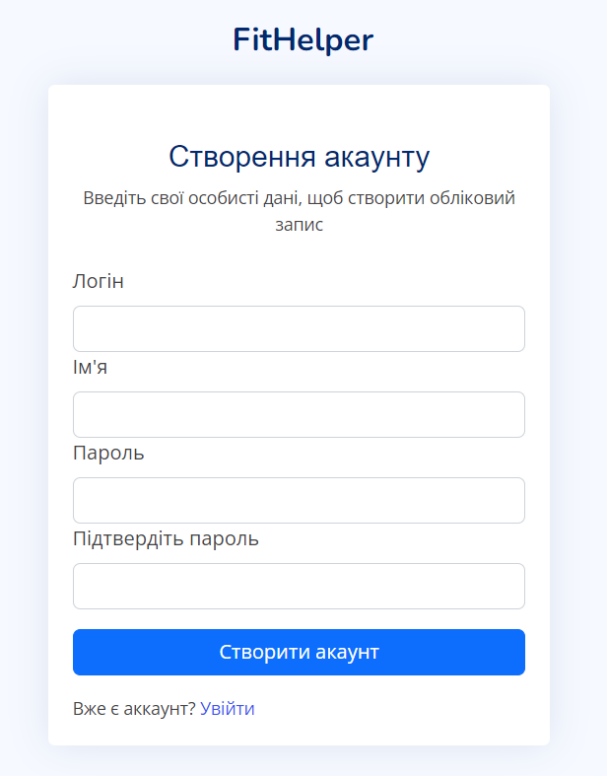


Рис. 19 Форма реєстрації

При вході систему ввівши неправильно дані буде виводитись помилка входу та показуватиметься повідомленню користувачеві (рис. 20).

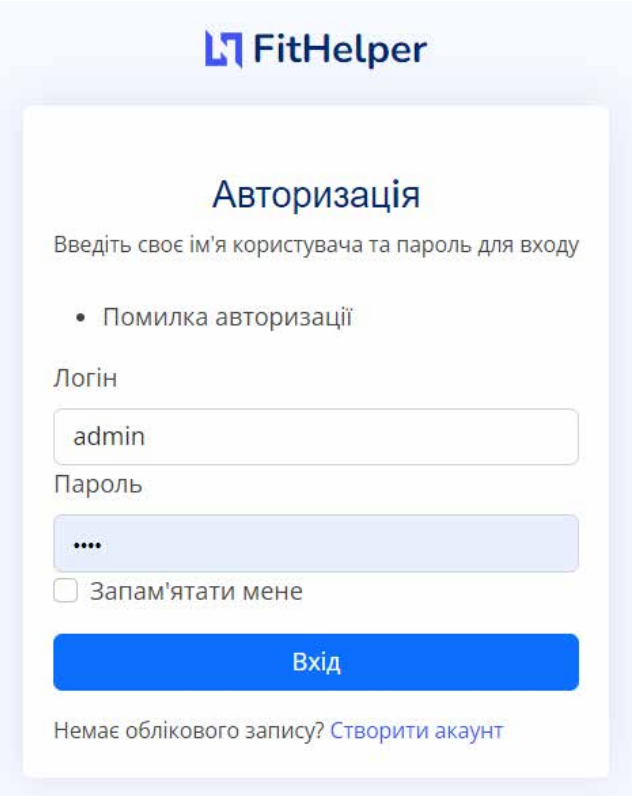


Рис. 20 Помилка при спробі авторизації

Увійшовши в систему можна обрати вкладку Тренування та переглянути тренування (рис. 21).

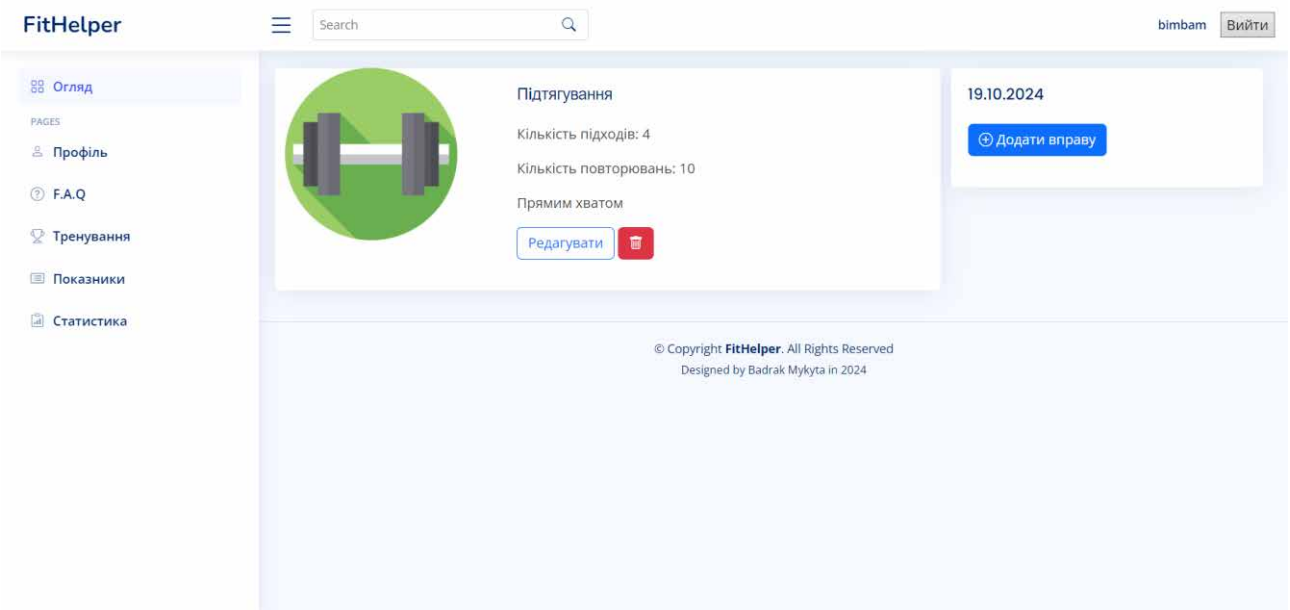


Рис. 21 Сторінка тренувань

Користувач має можливість додати нове тренування, натиснувши кнопку "Додати вправу". Після цього з'являється вікно з формами для заповнення необхідних даних, як показано на рисунку 22.

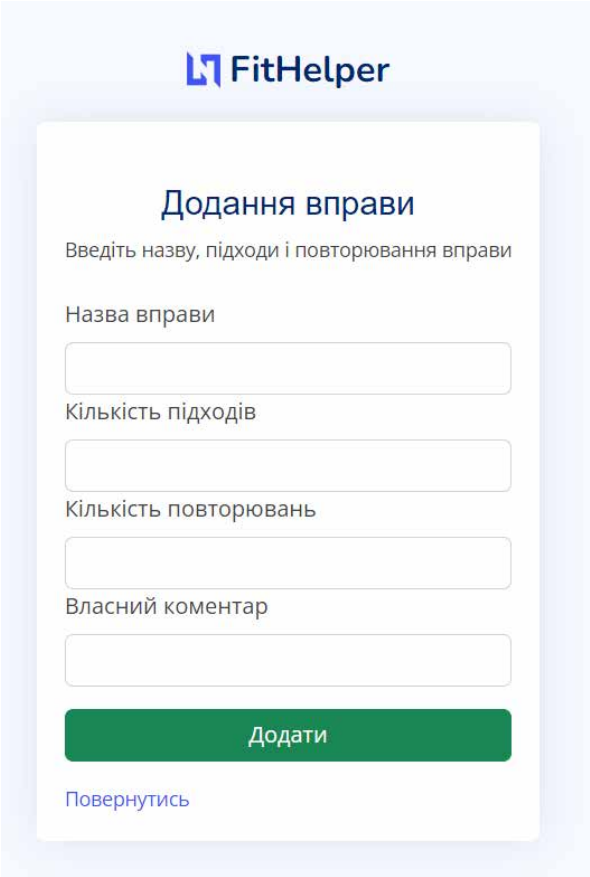


Рис. 22 Форма запису тренувань

Також розроблена система містить можливість видалення тренування, це вікно представлено на рис.23.

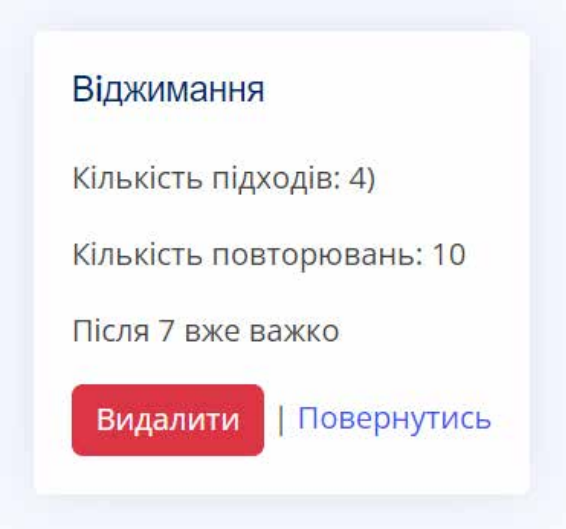


Рис. 23 Підтвердження видалення тренування

Наступною вкладинкою в меню є Показники, ця сторінка представлена на рис. 24.

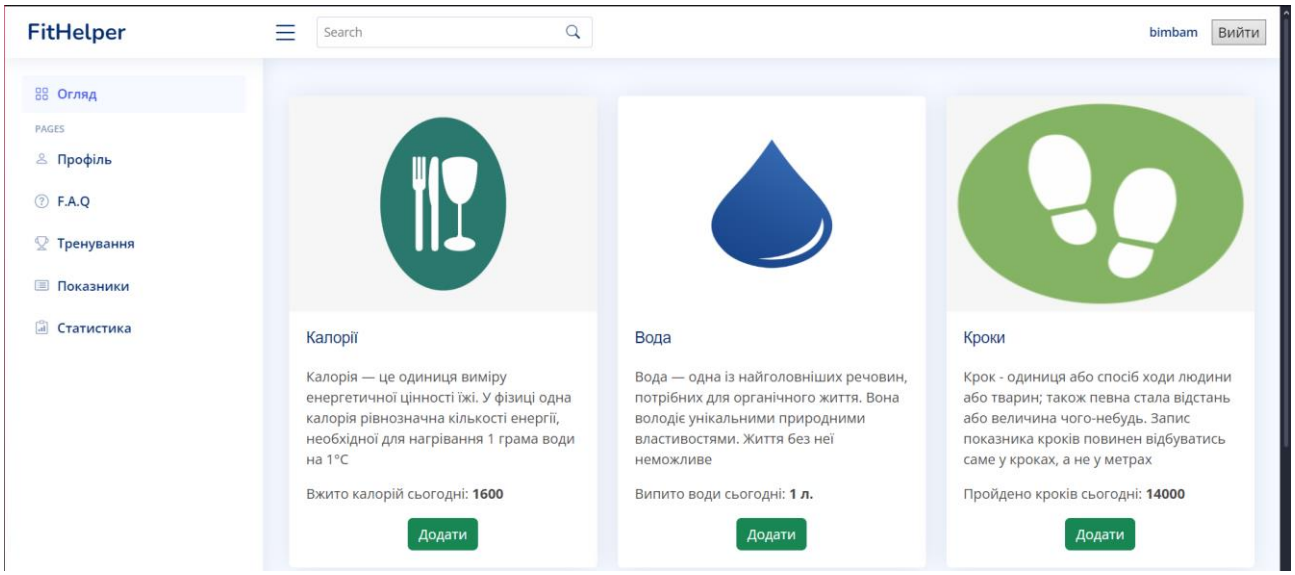


Рис. 24 Сторінка показників

Система містить можливість додавання показників. При натисканні на кнопку “Додати” відкривається відповідне вікно з формами заповнення даних (рис. 25).

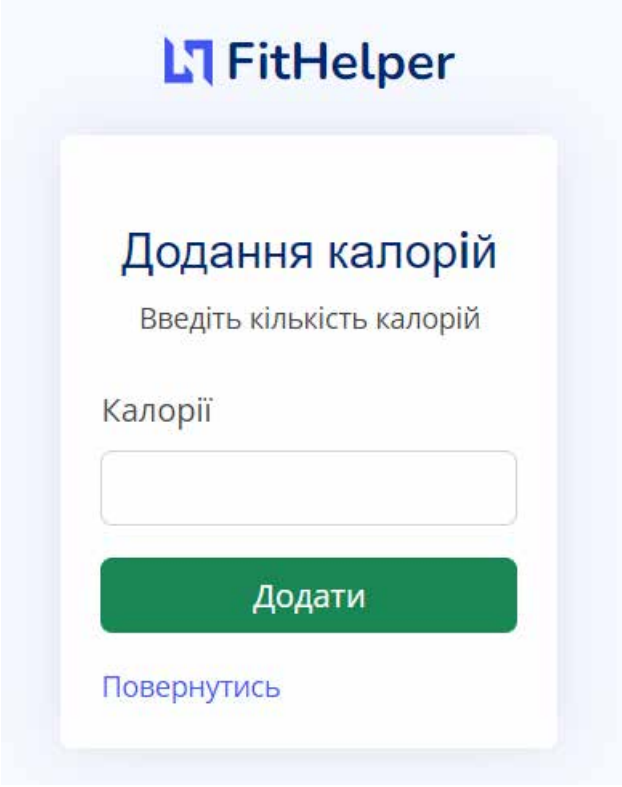


Рис. 25 Форма додавання калорій

Додаємо дані в систему і бачимо на рис.26 додані показники за сьогодні.

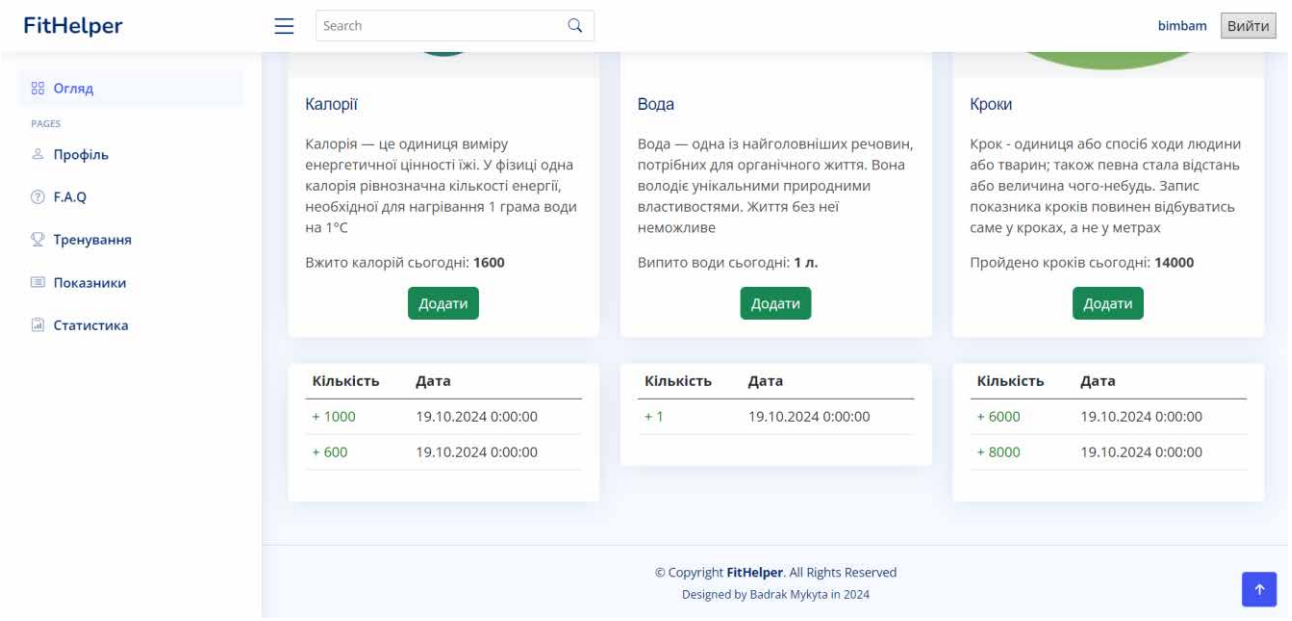


Рис. 26 Сторінка показників після сьогоденних записів

У меню вибираємо пункт "Профіль", де користувач може переглянути свою інформацію. Це вікно показано на рисунку 27.

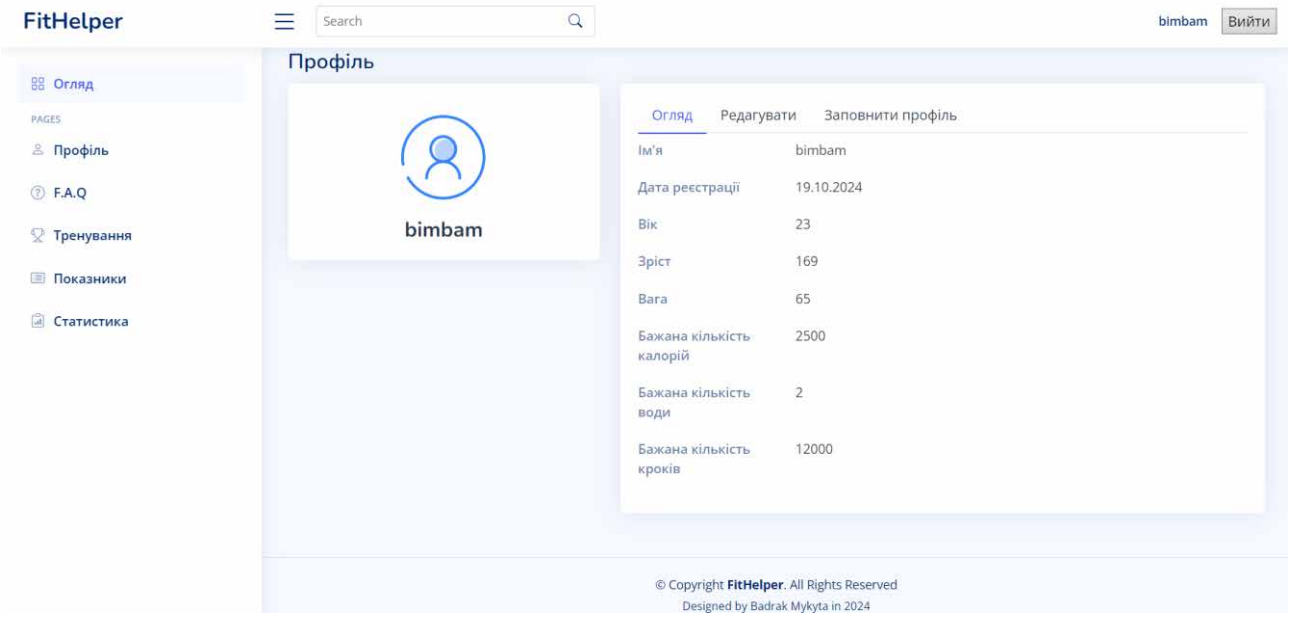


Рис. 27 Сторінка профілю користувача

Інформацію про користувача можна змінити, ця сторінка зображена на рис.28.

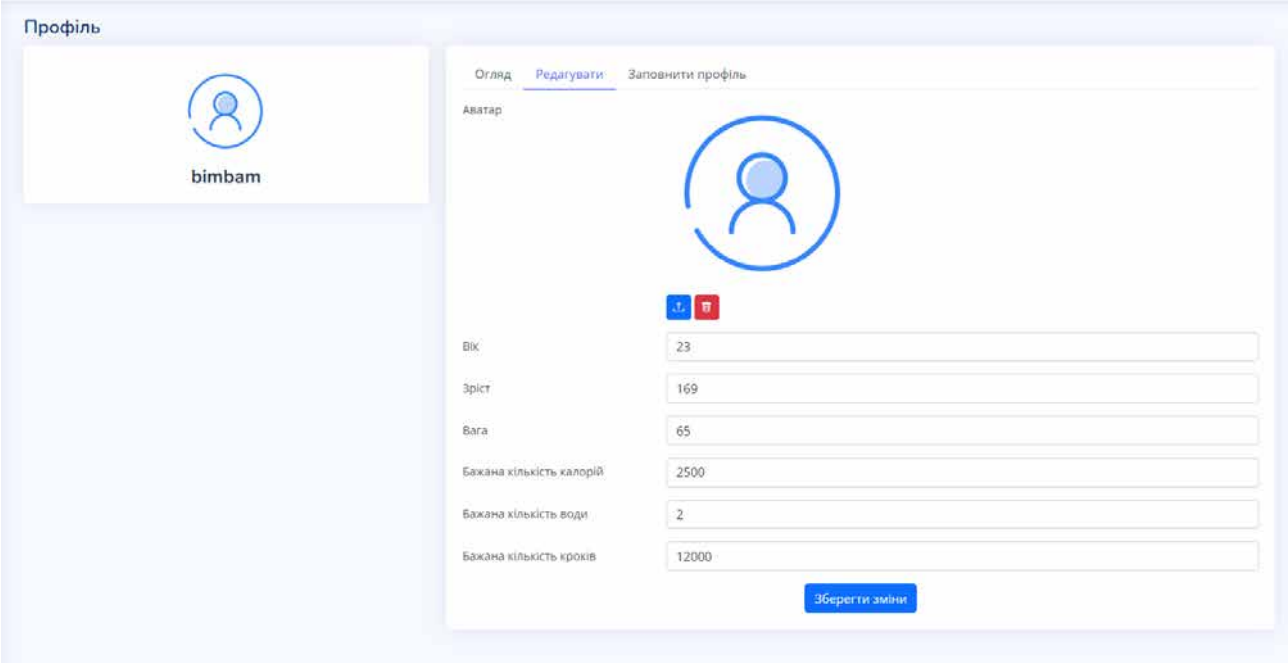


Рис. 28 Редагування профілю

Зайшовши на сторінку Статистики ми можемо побачити 3 колонки за нашими показниками і праворуч селектор дат, за якими ми можемо зробити вибірку (Рис. 29).

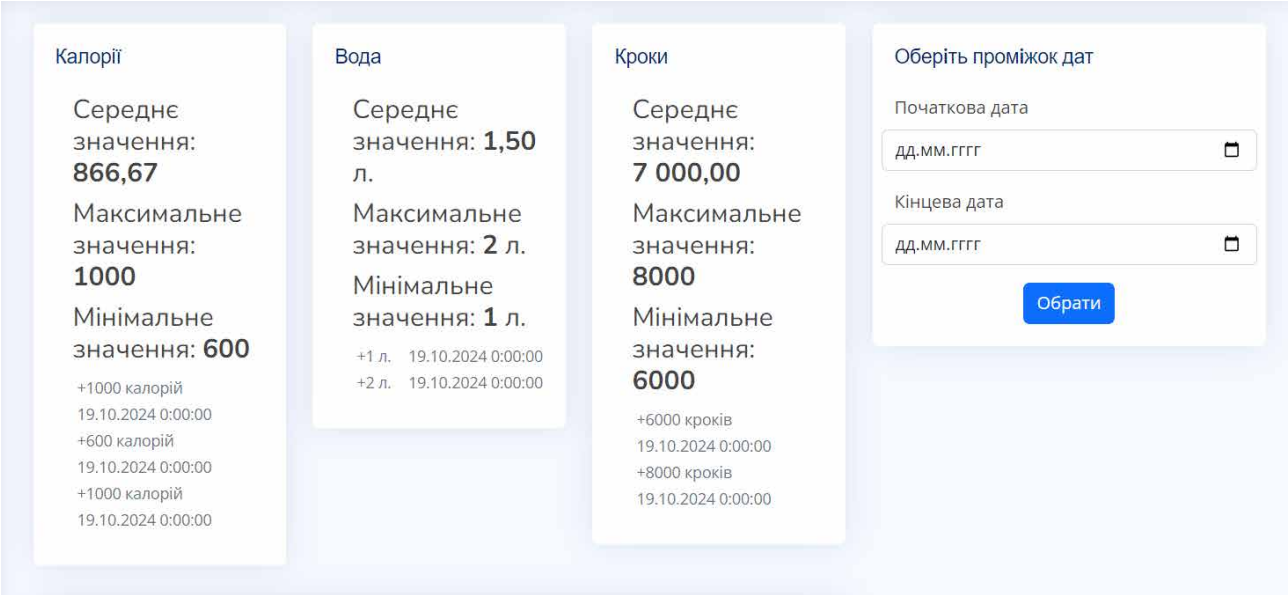


Рис. 29 Сторінка статистики

Після здійснення вибірки у нас згенеруються висновки за всіма показниками та тренуваннями, а також сформуються персональні поради та підказки щодо харчування та тренувань (Рис. 30).

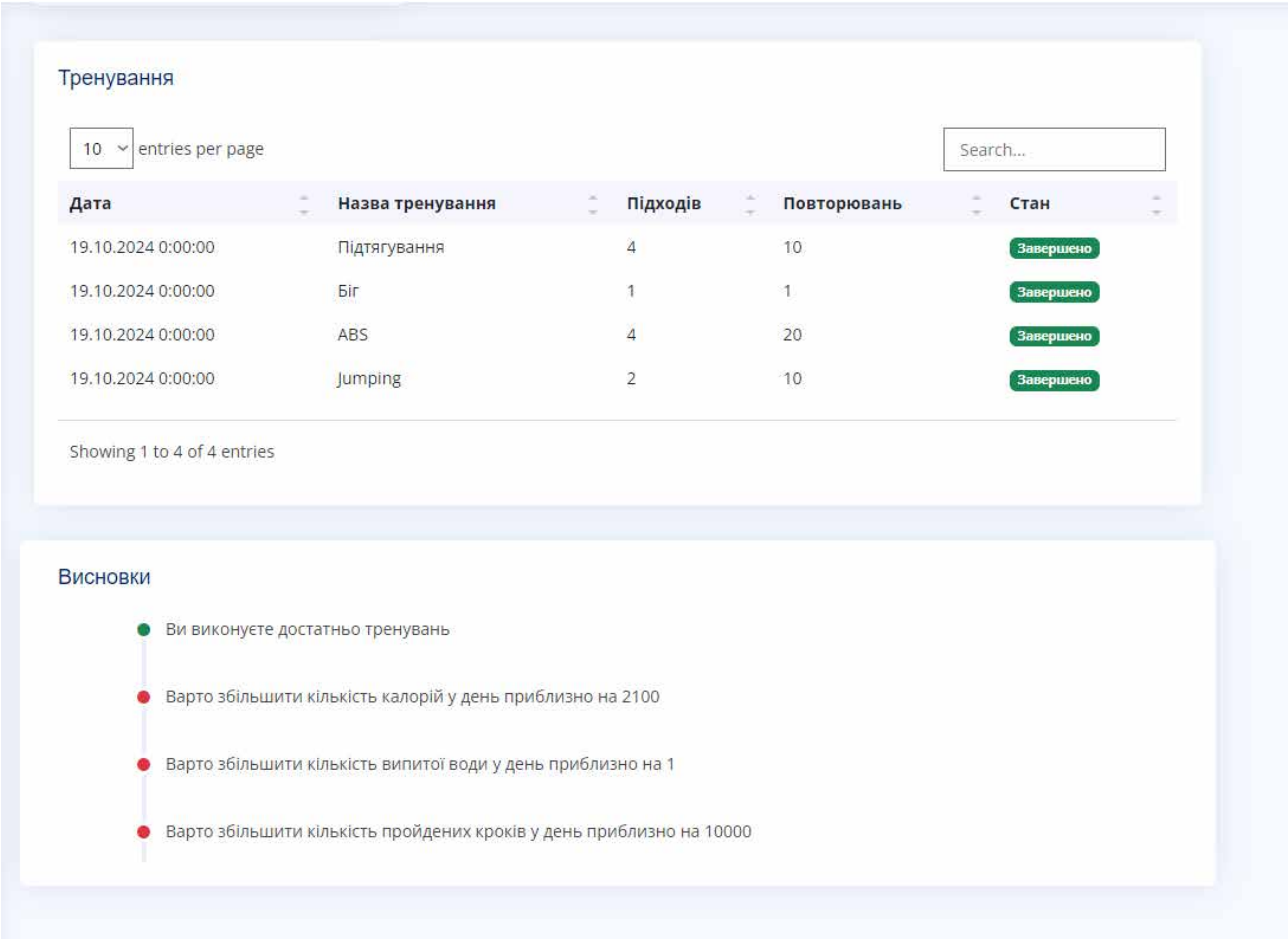


Рис. 30 Висновки

4.3 Інструкція користувача

Посібник користувача – це важливий документ, розроблений для того, щоб допомогти користувачам зрозуміти та максимально використати певний продукт або систему. Його головна мета — розширити можливості людей, пропонуючи детальні вказівки щодо всіх аспектів продукту, від початкового налаштування до розширених функцій. У цьому розділі розглядатимуться найважливіші компоненти та організація посібника користувача, підкреслюючи його роль у покращенні взаємодії з користувачем та мінімізації потенційних проблем під час роботи.

Основна роль посібника користувача полягає в тому, щоб пов’язати складність продукту зі здатністю користувача його використовувати. Перекладаючи технічну термінологію доступною мовою, це гарантує, що навіть

ті, хто не має технічних знань, зможуть впевнено працювати з продуктом. Завдяки чітким інструкціям і докладним поясненням посібник користувача дозволяє користувачам розкрити весь потенціал продукту, мінімізуючи розчарування та плутанину.

Добре структурований посібник користувача зазвичай складається з кількох ключових розділів.

Вступ є основою для посібника, описуючи призначення продукту, основні характеристики та переваги. У цьому розділі також можна коротко описати цільову аудиторію, підкреслюючи, що посібник призначений для користувачів із різним рівнем технічних знань. Встановлюючи контекст, вступ готує читачів до наступних інструкцій.

Розділ «Початок роботи» особливо важливий для новачків. Він пропонує детальні вказівки щодо процесу інсталяції та налаштування, гарантуючи, що користувачі розуміють необхідні кроки для підготовки продукту до використання. Ключові компоненти цього розділу включають системні вимоги, які описують апаратне та програмне забезпечення, необхідне для ефективної роботи; інструкції з інсталяції, які містять покрокову інструкцію з інсталяції програмного забезпечення або монтажу обладнання; і початкові кроки налаштування, які допомагають персоналізувати роботу користувача.

Нижче наведено огляд інтерфейсу користувача з детальним описом його основних компонентів та їхніх функцій. Цей розділ зазвичай пояснює навігаційне меню, допомагаючи користувачам зрозуміти, як отримати доступ до різних функцій. Він також визначає важливі кнопки та піктограми, ілюструючи їхні функції для більш плавної навігації. Візуальні засоби, такі як знімки екрана, можуть використовуватися, щоб допомогти користувачам візуалізувати свою взаємодію з інтерфейсом.

У розділі «Функції та функціональність» розповідається про основні можливості продукту. Кожна функція детально описана, пропонуючи покрокові інструкції для таких завдань, як введення даних, звітування або налаштування. Наведено реальні приклади, щоб продемонструвати, як можна ефективно

застосовувати ці функції, а також передові практики, які містять поради щодо підвищення ефективності та продуктивності користувачів.

Для користувачів, які стикаються з труднощами, розділ усунення несправностей є безцінним. Він вирішує типові проблеми, визначаючи часті проблеми та надаючи прості рішення. У цьому розділі також пояснюються типові повідомлення про помилки та міститься список поширених запитань для вирішення типових проблем і запитів.

Щоб забезпечити довговічність виробу, у розділі обслуговування та підтримки пропонуються вказівки щодо основних практик. Це може включати поради щодо регулярних оновлень для покращення продуктивності та безпеки, процедури резервного копіювання для запобігання втраті даних та інформацію про підтримку клієнтів, таку як контактні дані для технічної допомоги.

Глосарій термінів служить корисною довідкою, що визначає технічний жаргон, який використовується в посібнику. Цей ресурс дає змогу користувачам краще зрозуміти термінологію, сприяючи кращому розумінню інструкцій.

Інформація про безпеку є ще одним важливим компонентом, який містить запобіжні заходи, яких користувачі повинні вживати під час використання продукту. У цьому розділі можуть бути висвітлені потенційні небезпеки та надано вказівки щодо безпечної експлуатації, щоб запобігти нещасним випадкам.

Нарешті, додатки можуть містити додаткові ресурси, такі як технічні характеристики, відомості про відповідність нормативним вимогам або додаткові матеріали, які пропонують додаткові відомості про продукт.

Важливість посібника користувача неможливо переоцінити. Добре складений посібник підвищує задоволеність користувачів і сприяє успішному прийняттю продукту. Надаючи чіткі, стислі інструкції та вказівки щодо усунення несправностей, він зменшує ймовірність помилок і сприяє позитивній взаємодії з користувачем. Крім того, інформативний посібник користувача може значно зменшити кількість запитів у службу підтримки клієнтів, дозволяючи командам підтримки зосередитися на більш складних питання.

ВИСНОВКИ

Розробка програмного забезпечення фітнес-помічника успішно досягла мети магістерської роботи. Протягом усього проекту розглядалися різні аспекти, включаючи ідентифікацію проблем, дослідження, аналіз, проектування, впровадження та оцінку програмної системи.

Робота розпочалася з вивчення предметної області та формулювання проблеми дослідження. Існуючі рішення були переглянуті, щоб виявити прогалини та можливості для вдосконалення програмного забезпечення підтримки фізичної форми, що заклало основу для наступних етапів.

Було створено логічну модель даних для представлення інформаційних вимог і зв’язків у програмному забезпеченні фітнес-асистента. Ця модель послужила основою для проектування програмної системи. Крім того, була розроблена фізична модель даних, щоб забезпечити ефективне зберігання та пошук даних через вибрану систему керування базою даних.

Незважаючи на те, що система є перспективною для вдосконалення моніторингу та аналізу фітнесу, необхідні подальші дослідження для вивчення її масштабованості, сумісності з іншими платформами, пов’язаними зі здоров’ям, і довгострокового впливу на поведінку користувачів і результати для здоров’я.

Крім того, дослідження підкреслює першочергову важливість конфіденційності користувачів і безпеки даних у системах моніторингу фізичної форми. Враховуючи збір конфіденційної інформації, пов’язаної зі здоров’ям, необхідно вжити суворих заходів для захисту даних користувачів і забезпечення дотримання відповідних норм, таких як GDPR або HIPAA

Архітектура програмного забезпечення була ретельно розроблена для задоволення системних вимог і потреб у масштабованості. Для забезпечення надійності, функціональності та зручності було обрано відповідні засоби

розробки, включаючи ASP.NET Core 6 MVC, C#, MS SQL Server, Bootstrap 5, HTML, CSS і JavaScript.

На етапі впровадження дизайн було перетворено на фактичний код, а різні програмні компоненти були інтегровані. Для забезпечення функціональності та надійності програмного забезпечення було проведено ретельне тестування та налагодження. Відгуки користувачів і повторювані вдосконалення відіграли вирішальну роль у підвищенні продуктивності та зручності використання програмного забезпечення.

Оцінка програмного забезпечення фітнес-помічника підкреслила його ефективність у забезпеченні комплексного відстеження фізичної форми, персоналізованих планів тренувань, дієтичних рекомендацій і моніторингу досягнень. Програмне забезпечення продемонструвало свою здатність допомагати користувачам у досягненні фітнес-цілей і підтримці здорового способу життя.

Загалом програмне забезпечення фітнес-асистента робить внесок у сферу фітнес-технологій, пропонуючи практичне та орієнтоване на користувача рішення. Це стало прикладом успішного застосування принципів розробки програмного забезпечення з використанням сучасних інструментів і методологій. Результати цієї розробки служать основою для майбутніх удосконалень фітнес-технологій, заохочуючи людей до більш здорового способу життя.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. FatSecret – [Електронний ресурс] – Режим доступу: https://apps.apple.com/ru/app/счетчик-калорий-от-fatsecret/id347184248
2. MyFitnessPal – [Електронний ресурс] – Режим доступу: https://www.myfitnesspal.com/ru
3. Найкращі програми для занять спортом вдома – [Електронний ресурс] – Режим доступу: https://blog.allo.ua/luchshie-prilozheniya-dlya-zanyatij-sportom-doma_2020-04-53/
4. Багаторівнева архітектура – [Електронний ресурс] – Режим доступу: https://simpleone.ru/glossary/mnogourovnevaya-arhitektura/
5. Багаторівнева архітектура в ASP.NET MVC – [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/mvc5/23.5.php
6. Типи архітектури програмного забезпечення – [Електронний ресурс] – Режим доступу: https://medium.com/nuances-of-programming/4-типа-архитектуры-программного-обеспечения-917133174724
7. What is Unified Modeling Language (UML)? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
8. ПРОЕКТУВАННЯ ER-ДІАГРАМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ru/skills/proektirovanie-er-diagrammy/
9. Діаграма варіантів використання (UseCase diagram) – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ru/fd_use-case-diagram.html
10. ПРОЕКТУВАННЯ USE CASE ДІАГРАМИ. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СИСТЕМИ – [Електронний ресурс] – Режим доступу:

https://nationalteam.worldskills.ru/skills/proektirovanie-use-case-diagrammy-opredelenie-funktsionalnykh-vozmozhnostey-sistemy/

11. What is Sequence Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/
12. UML - Activity Diagrams – Tutorialspoint – [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm
13. Побудова діаграми класів – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ru/gpg_class-diagram.html
14. Entity Relationship Diagram - Data Modeling – [Електронний ресурс] – Режим доступу: https://www.visualparadigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html
15. Документація Bootstrap (офіційний сайт) – [Електронний ресурс] – Режим доступу: www.getbootstrap.com/documentation.
16. W3Schools – [Електронний ресурс] – Режим доступу: www.w3schools.com
17. Сміт, Дж. (2018). «Тенденції додатків для фітнесу та залучення користувачів: огляд наявної літератури». Журнал технологій здоров'я та фітнесу, 12 (3), 45-62.
18. Патель, Р. (2019). «Аналіз дієти та рекомендації щодо харчування для додатків для фітнесу». Міжнародний журнал харчових наук та харчування, 30 (4), 512-526.
19. Джонсон, М. (2019). «Розробка інтерфейсів користувача для некомерційних програм». Journal of Nonprofit Technology, 10(2), 45-62.
20. Олександр Леоненков. Самовчитель UML, 2007. – [Електронний ресурс] – Режим доступу: http://kniga.scienceontheweb.net/samouchitel-uml-2.html.
21. Logical Data Model. Dataedo – [Електронний ресурс] – Режим доступу: https://dataedo.com/kb/data-glossary/logical-data-model
22. Logical Data Model. Techopedia – [Електронний ресурс] – Режим доступу: https://www.techopedia.com/definition/23969/logical-data-model

23. Microsoft Docs. (2021). Layered architecture pattern – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/azure/architecture/patterns/layered
24. What is Data Mining? Key Concepts, How Does it Work? - [Electronic resource]. - Access mode: https://www.upgrad.com/blog/what-is-datamining key-concepts-how-does-it-work/
25. Data mining tools for better data analysis - [Electronic resource]. – Access mode: https://www.ionos.co.uk/digitalguide/online-marketing/webanalytics/a-comparison-of-data-mining-tools/
26. OLAP systems. – [Electronic resource]. - Access mode: https://pidru4niki.com/1670032447784/informatika/olap-sistemi
27. Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques (3rd ed.). Morgan Kaufmann Publishers.
28. Naive Bayes Classifiers - [Electronic resource]. - Access mode: https://www.geeksforgeeks.org/naive-bayes-classifiers/
29. Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. ACM SIGMOD Record, 22(2), 207-216.
30. Cluster Analysis Guide with Examples - [Electronic resource]. – Access mode: https://www.resonio.com/market-research/cluster-analysis

ДОДАТОК А

Фрагменти програмного коду. Функція створення тренування

```
using Microsoft.EntityFrameworkCore;
using YourNamespace.Data;

public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddDbContext<ApplicationDbContext>(options =>
            options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"))); // Update connection string as necessary

        services.AddControllersWithViews();
    }
}

using Microsoft.AspNetCore.Mvc;
using YourNamespace.Data;
using YourNamespace.Models;
using System.Threading.Tasks;

namespace YourNamespace.Controllers
{
    public class WorkoutsController : Controller
    {
        private readonly ApplicationDbContext _context;

        public WorkoutsController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: Workouts/Create
        public IActionResult Create()
        {
            return View();
        }

        // POST: Workouts/Create
```

```
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create(Workout workout)
        {
            if (ModelState.IsValid)
            {
                _context.Workouts.Add(workout);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(workout);
        }

        // GET: Workouts
        public async Task<IActionResult> Index()
        {
            var workouts = await _context.Workouts.ToListAsync();
            return View(workouts);
        }
    }
}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Workouts}/{action=Index}/{id?}");
    });
}
```

ДОДАТОК Б

Фрагменти програмного коду. Функція аналізу даних та створення висновків

```
namespace FitnessHelper.Controllers
{
  public class StatsController : Controller
  {
    private readonly ApplicationDbContext _context;

    public StatsController(ApplicationDbContext context)
    {
      _context = context;
    }
    public IActionResult Index(DateTime StartDate, DateTime EndDate)
    {
      string userId = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;
      var training = _context.Training.Where(t => t.UserId == userId && t.DateOfTrain >= StartDate && t.DateOfTrain <= EndDate).ToList();
      var calories = _context.Calories.Where(t => t.UserId == userId && t.DateOfNote >= StartDate && t.DateOfNote <= EndDate).ToList();
      var waters = _context.Waters.Where(t => t.UserId == userId && t.DateOfNote >= StartDate && t.DateOfNote <= EndDate).ToList();
      var steps = _context.Steps.Where(t => t.UserId == userId && t.DateOfNote >= StartDate && t.DateOfNote <= EndDate).ToList();
      var profile = _context.ProfileDetails.Where(y => y.UserId == userId).ToList();

      var countTraining = _context.Training.Where(t => t.UserId == userId && t.DateOfTrain >= StartDate && t.DateOfTrain <= EndDate).Count();
      var maxCal = calories.Any() ? calories.Max(f => f.Cal) : 0;
      var minCal = calories.Any() ? calories.Min(x => x.Cal) : 0;
      var avgCal = string.Format("{0:N2}", calories.Any() ? calories.Average(f => f.Cal) : 0);
      var maxLiters = waters.Any() ? waters.Max(f => f.Liters) : 0;
      var minLiters = waters.Any() ? waters.Min(f => f.Liters) : 0;
      var avgLiters = string.Format("{0:N2}", waters.Any() ? waters.Average(f => f.Liters) : 0);
      var maxSteps = steps.Any() ? steps.Max(f => f.Step) : 0;
      var minSteps = steps.Any() ? steps.Min(f => f.Step) : 0;
      var avgSteps = string.Format("{0:N2}", steps.Any() ? steps.Average(f => f.Step): 0);

      //Calories
```

```
        var joinedTables = calories.Join(profile, t1 => t1.UserId, t2 => t2.UserId, (t1, t2) => new { Calories = t1, ProfileDetails =
t2 });
        var rowsToCompare = joinedTables.Where(j => j.Calories.UserId == userId && j.ProfileDetails.UserId == userId);
        var difference = rowsToCompare.Select(j => j.ProfileDetails.calorie_goal - maxCal + minCal).FirstOrDefault();
        //Water
        var joinedTables2 = waters.Join(profile, t1 => t1.UserId, t2 => t2.UserId, (t1, t2) => new { Waters = t1, ProfileDetails = t2
});
        var rowsToCompare2 = joinedTables2.Where(j => j.Waters.UserId == userId && j.ProfileDetails.UserId == userId);
        var difference2 = rowsToCompare2.Select(j => j.ProfileDetails.water_goal - maxLiters + minLiters).FirstOrDefault();
        //Steps
        var joinedTables3 = steps.Join(profile, t1 => t1.UserId, t2 => t2.UserId, (t1, t2) => new { Steps = t1, ProfileDetails = t2 });
        var rowsToCompare3 = joinedTables3.Where(j => j.Steps.UserId == userId && j.ProfileDetails.UserId == userId);
        var difference3 = rowsToCompare3.Select(j => j.ProfileDetails.step_goal - maxSteps + minSteps).FirstOrDefault();

        var viewModel = new StatsViewModel
        {
            Training = training,
            Calories = calories,
            Water = waters,
            Steps = steps,
            MaxCal = maxCal,
            MinCal = minCal,
            AvgCal = avgCal,
            MaxLiters = maxLiters,
            MinLiters = minLiters,
            AvgLiters = avgLiters,
            MaxSteps = maxSteps,
            MinSteps = minSteps,
            AvgSteps = avgSteps,
            TrainCount = countTraining,
            CalorieDiff = difference,
            WaterDiff = difference2,
            StepDiff = difference3
        };

        return View(viewModel);
    }
  }
}
  public class StatsViewModel
  {
```

```
        [DataType(DataType.Date)]
        [DisplayFormat(DataFormatString = "{dd-MM-yyyy}", ApplyFormatInEditMode = true)]
        public DateTime StartDate { get; set; }
        [DataType(DataType.Date)]
        [DisplayFormat(DataFormatString = "{dd-MM-yyyy}", ApplyFormatInEditMode = true)]
        public DateTime EndDate { get; set; }
        public string? UserId { get; set; }
        public List<Training> Training { get; set; }
        public int TrainCount { get; set; }
        public List<Calories> Calories { get; set; }
        public int MaxCal { get; set; }
        public int MinCal { get; set; }
        public string AvgCal { get; set; }
        public List<Water> Water { get; set; }
        public double MaxLiters { get; set; }
        public double MinLiters { get; set; }
        public string AvgLiters { get; set; }
        public List<Steps> Steps { get; set; }
        public int MaxSteps { get; set; }
        public int MinSteps { get; set; }
        public string AvgSteps { get; set; }
        public List<ProfileDetails> ProfileDetails { get; set; }
        public double CalorieDiff { get; set; }
        public double WaterDiff { get; set; }
        public double StepDiff { get; set; }
    }
}
```