МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Комп'ютерна система оптимізації фізичних навантажень спортсменів**

Виконав: студент 4 курсу, групи 6.1210-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
О.С. Швець
(ініціали та прізвище)
Керівник доцент кафедри ЕІС та ПЗ
А.І. Безверхий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дісітел»
П.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____перший (бакалавський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Програмне забезпечення систем_____
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Тетяна КРИТСЬКА
“_01_” __березня_____2024_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Швецю Олександру Сергійовичу
(прізвище, ім’я, по батькові)

1.Тема роботи Комп’ютерна система оптимізації фізичних навантажень спортсменів
керівник роботи _____Безверхий Анатолій Ігорович_____
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від 26.12.2023 р. № 2215-с
2. Строк подання студентом кваліфікаційної роботи_____07.06.2024_____
3. Вихідні дані кваліфікаційної роботи бакалавра
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- вивчення та збір матеріалів, пов'язаних з темою дипломної роботи;
- вивчення та оцінка наявних рішень і аналогів;
- аналіз та оцінка сучасних технологій для створення клієнтської частини веб-застосунків;
- аналіз PosgreSql СУБД;
- огляд алгоритму машинного навчання для прогнозування майбутніх навантажень.
- розробка та уточнення функціональних вимог до системи;
- створення архітектури системи та проєктування системи оптимізації фізичних навантажень;
- розробка програмного забезпечення та його тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	01.03 – 10.03.24	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	11.03 – 12.03.24	виконано
3	Аналіз існуючих методів рішення	13.03 – 14.03.24	виконано
4	Вивчення та збір матеріалів, пов'язаних з темою дипломної роботи	15.03 – 20.03.24	виконано
5	Вивчення та оцінка наявних рішень і аналогів	21.03 – 26.03.24	виконано
6	Аналіз та оцінка сучасних технологій для створення клієнтської частини веб-застосунків	27.03 – 28.03.24	виконано
7	Аналіз PostgreSql СУБД	29.03 – 13.04.24	виконано
8	Огляд алгоритму машинного навчання для прогнозування майбутніх навантажень	14.04 – 16.04.24	виконано
9	Розробка та уточнення функціональних вимог до системи	17.04 – 19.04.24	виконано
10	Створення архітектури системи та проєктування системи оптимізації фізичних навантажень	20.04 – 01.05.24	виконано
11	Розробка програмного забезпечення та його тестування	02.05 – 13.05.24	виконано
12	Оформлення звіту	13.05 – 20.05.24	виконано

Студент _____ О.С. Швець
(підпис) (прізвище та ініціали)

Керівник роботи _____ А.І. Безверхий
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено
Нормоконтролер _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок – 69

Рисунків – 23

Джерел – 9

Швець О.С. Комп’ютерна система оптимізації фізичних навантажень спортсменів: кваліфікаційна робота бакалавра спеціальності 121 «Програмне забезпечення систем» / наук. керівник А.І. Безверхий. Запоріжжя : ЗНУ, 2024. 69 с.

Одним з ключових аспектів успішного тренувального процесу є ефективне планування фізичних навантажень. Використання даних, алгоритмів та аналітичних методів допомагає тренерам і спортсменам приймати обґрунтовані рішення щодо тренувального режиму. Впровадження комп'ютерної системи для оптимізації фізичних навантажень дозволяє аналізувати та коригувати навантаження з урахуванням індивідуальних потреб і можливостей спортсмена, покращуючи їхні результати та знижуючи ризик травм.

Проведено порівняльний аналіз існуючих систем-конкурентів. Виходячи з результатів цього аналізу, було розроблено комп'ютерну систему для оптимізації фізичних навантажень спортсменів. Реалізована система дозволяє тренерам і спортсменам аналізувати навантаження, планувати тренування та здійснювати коригування в режимі реального часу, що допомагає спортсменам досягати кращих результатів.

Мета дослідження – створення та оптимізація комп'ютерної системи для ефективної оптимізації фізичних навантажень спортсменів.

Ключові слова: *Машинне навчання, тренер, оптимізація фізичних навантажень, спортсмен вправа, комплекс, прогрес.*

ABSTRACT

Pages – 69

Figures – 23

References – 9

Shvets O.S. Computer System for Optimizing Athletes' Physical Loads: Bachelor's Qualification Work, specialty 121 "Software Systems" / scientific supervisor A.I. Bezverkhyi. Zaporizhzhia: ZNU, 2024. 69 p.

One of the key aspects of a successful training process is the effective planning of physical loads. The use of data, algorithms, and analytical methods helps coaches and athletes make informed decisions about the training regime. The implementation of a computer system for optimizing physical loads allows for the analysis and adjustment of loads based on the individual needs and capabilities of the athlete, improving their performance and reducing the risk of injuries.

A comparative analysis of existing competitor systems was conducted. Based on the results of this analysis, a computer system for optimizing athletes' physical loads was developed. The implemented system allows coaches and athletes to analyze loads, plan training sessions, and make real-time adjustments, helping athletes achieve better results.

The aim of the study is to create and optimize a computer system for the effective optimization of athletes' physical loads.

Keywords: Machine learning, coach, optimization of physical loads, athlete, exercise, complex, progress.

ЗМІСТ

ВСТУП

Актуальність теми

Створення комп'ютерної системи для оптимізації фізичних навантажень спортсменів є надзвичайно актуальною у сучасному спортивному світі з численних причин.

По-перше, інтерес до використання технологій у спорті постійно зростає. Спортсмени та тренери шукають нові способи покращення результатів, а комп'ютерні системи можуть стати потужним інструментом у цьому процесі. Розробка системи оптимізації навантажень може допомогти аналізувати та оптимізувати тренувальні процеси, що сприятиме підвищенню ефективності тренувань та досягненню кращих результатів.

По-друге, така система може забезпечити індивідуальний підхід до кожного спортсмена. Кожен атлет має унікальні потреби та можливості, і використання комп'ютерної системи дозволить створити персоналізовані програми тренувань, які оптимально відповідатимуть їхнім цілям та потребам.

По-третє, використання комп'ютерних технологій дозволяє здійснювати більш точний та об'єктивний аналіз фізичного стану спортсменів. Збір та обробка даних про фізичні параметри, показники тренувального процесу та реакції на навантаження може допомогти виявити слабкі місця, уникнути перевантаження та травм.

Комп'ютерна система оптимізації навантажень може сприяти збереженню часу та ресурсів. Автоматизована обробка даних та генерація рекомендацій щодо тренувального процесу дозволить тренерам та спортсменам ефективно використовувати свій час та зосередитися на досягненні результатів.

Більш того, після пандемії COVID-19 зросла потреба у віддалених тренуваннях, що зробило використання такого додатку для тренерів ще більш корисним та зручним. Тренери зможуть створювати індивідуальні програми

тренувань для своїх учнів та відстежувати їх прогрес онлайн. Це не лише забезпечить безпеку та здоров'я учасників, але й відкриє нові можливості для тренування та розвитку, незалежно від місця проживання або доступності спортивних закладів.

Загалом, розробка комп'ютерної системи для оптимізації фізичних навантажень спортсменів є актуальною темою, оскільки вона відповідає потребам у покращенні спортивних результатів, надає індивідуальний підхід до тренувань, допомагає в ефективному використанні часу та ресурсів та використовує потенціал сучасних технологій для досягнення максимальних результатів у спорті.

Мета дослідження

Створення та оптимізація комп'ютерної системи для ефективної оптимізації фізичних навантажень спортсменів.

Завдання дослідження

Аналіз особливостей та існуючих рішень для розробки комп'ютерної системи оптимізації фізичних навантажень спортсменів з використанням сучасних технологій. Вивчення доступних підходів та методик розробки системи оптимізації навантажень для вибору найбільш відповідної для втілення концепції системи.

Об’єкт дослідження

Об’єктом дослідження є процес розробки комп'ютерної системи оптимізації фізичних навантажень спортсменів з використанням сучасних технологій.

Предмет дослідження

Предметом дослідження є технології та методики створення програмного забезпечення для ефективного аналізу та оптимізації фізичних тренувань спортсменів з метою поліпшення їхніх спортивних результатів.

Методи дослідження

Теоретичний аналіз різноманітних підходів до оптимізації тренувань спортсменів; вивчення класифікацій та принципів проектування комп'ютерних систем для фізичної підготовки; порівняльний аналіз методів та технологій для вибору найбільш ефективних засобів реалізації системи оптимізації навантажень.

Практичне значення одержаних результатів

Практичне значення одержаних результатів дослідження полягає у можливості розробленої комп'ютерної системи оптимізації фізичних навантажень спортсменів забезпечити тренерам та спортсменам зручний та ефективний інструмент для досягнення спортивних цілей. Це дозволить тренерам створювати індивідуальні програми тренувань, а спортсменам - відстежувати їхній прогрес та отримувати персоналізовані рекомендації для поліпшення результатів. Отримані результати дослідження можуть бути використані для оптимізації тренувального процесу, зменшення ризику травм та підвищення ефективності підготовки спортсменів, що в свою чергу сприятиме досягненню високих спортивних досягнень та покращенню загального фізичного стану.

Глосарій

Машинне навчання - галузь штучного інтелекту, яка розробляє алгоритми і методи, що дозволяють комп'ютерним системам автоматично навчатися та вдосконалюватися на основі даних, без необхідності явного програмування.

Комплекс - у контексті фізичних тренувань, це набір вправ, виконуваних у певному порядку, спрямований на досягнення конкретних тренувальних цілей, таких як покращення сили, витривалості або гнучкості.

Вправа - фізичне завдання або рух, виконуване з метою тренування певних груп м'язів або покращення загальної фізичної форми. Вправи можуть бути частиною комплексу або виконуватися окремо.

Тренер - спеціаліст, який надає керівництво, інструкції та підтримку спортсменам або індивідуальним особам під час тренувального процесу. Тренер допомагає розробляти тренувальні програми, коригувати техніку виконання вправ і мотивувати на досягнення спортивних цілей.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ ОПТИМІЗАЦІЇ ФІЗИЧНИХ НАВАНТАЖЕНЬ СПОРТСМЕНІВ

1.1 Огляд літературних джерел

Оптимізація фізичних навантажень у спорті є одним із ключових аспектів досягнення високих результатів. Комп'ютерні системи, спрямовані на оптимізацію цих навантажень, стали невід'ємною частиною сучасного тренувального процесу. Їх роль полягає в аналізі та моніторингу фізіологічних показників спортсменів, що дозволяє тренерам та медичним працівникам надавати індивідуальні рекомендації щодо оптимального навантаження, відпочинку та відновлення [1].

Однією з ключових переваг використання комп'ютерних систем оптимізації фізичних навантажень є можливість індивідуалізації тренувальних планів. Кожен спортсмен має унікальні фізіологічні особливості, цілі та обмеження. Завдяки збору та аналізу данних про фізіологічний стан спортсмена, комп'ютерна система може надавати персоналізовані рекомендації, які враховують його індивідуальні потреби.

Крім того, використання таких систем дозволяє попереджати травми та перевантаження. Шляхом постійного моніторингу фізіологічних показників спортсмена та аналізу динаміки його тренувань, можна вчасно виявляти ознаки перевантаження або неправильного навантаження, що дозволяє тренерам вчасно втручатися та уникнути можливих травм.

Подальший розвиток комп'ютерних систем оптимізації фізичних навантажень спортсменів передбачає впровадження нових технологій та методик аналізу даних. Машинне навчання та штучний інтелект стають все більш важливими компонентами таких систем, що дозволяє автоматизувати процес аналізу даних та надавати більш точні та зрозумілі рекомендації спортсменам та тренерам [2].

Загальний аналіз літературних джерел свідчить про значущість та перспективність використання комп’ютерних систем у сучасному спорті. Вони стають необхідним інструментом для досягнення високих результатів та підвищення рівня підготовки спортсменів у всіх галузях спортивної діяльності.

1.2 Аналіз програмних продуктів-аналогів

Створення комп'ютерної системи для оптимізації фізичних навантажень спортсменів є надзвичайно актуальною темою в спортивному світі. У зв'язку з цим, аналіз програмних продуктів-аналогів може бути важливим етапом для визначення найкращих рішень у розробці та впровадженні такої системи. Давайте розглянемо деякі з них для отримання інсайтів та порівняння з нашим проектом.

TrainingPeaks — це інтернет-платформа, що спеціалізується на наданні сервісів для планування, аналізу та моніторингу тренувальних програм для спортсменів різних спортивних дисциплін. Заснована у 1999 році, ця платформа швидко здобула популярність серед професійних спортсменів та їхніх тренерів завдяки своїм потужним функціональним можливостям та надійності.

Інтерфейс сайту можемо побачити на рисунку 1.

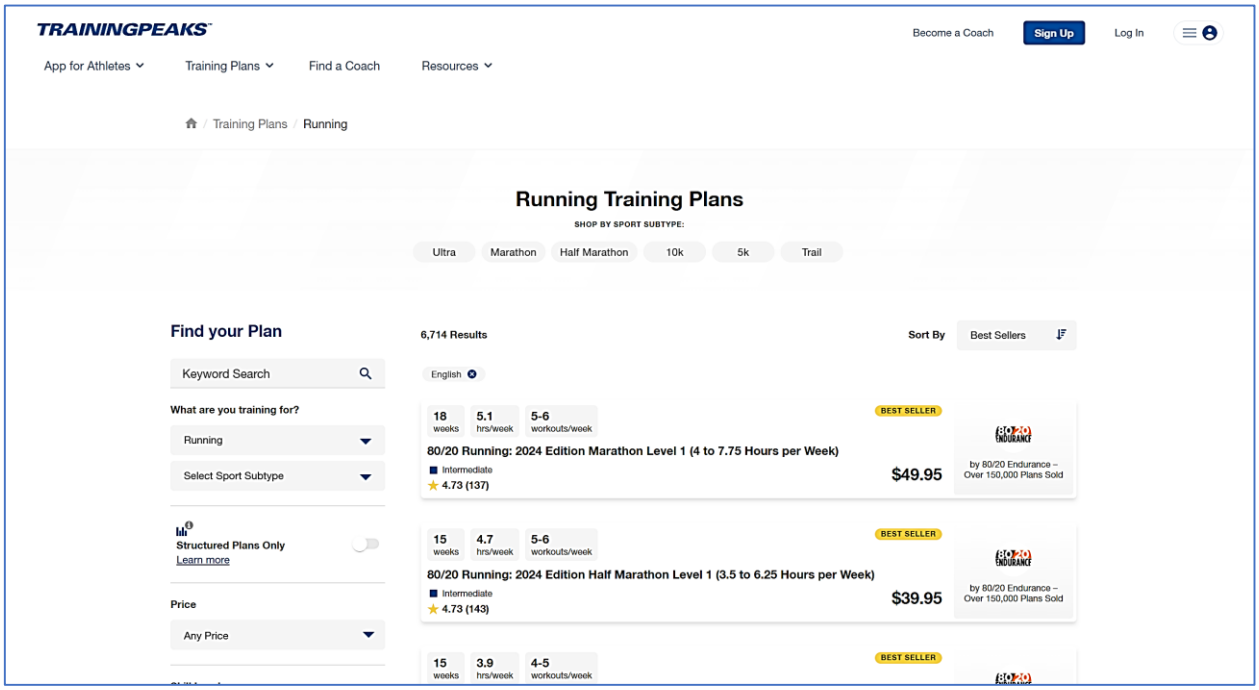

Рисунок 1 — Інтерфейс TrainingPeaks

Переваги:

1. TrainingPeaks має велику базу користувачів, включаючи професійних спортсменів та їхніх тренерів, що підтверджує його авторитет на ринку.
2. Веб-платформа та мобільний додаток TrainingPeaks є добре організованими та інтуїтивно зрозумілими у використанні.
3. Спортсмени можуть легко створювати та налаштовувати свої тренувальні програми, а тренери мають можливість моніторити та аналізувати прогрес своїх підопічних.
4. Користувачі можуть залишати відгуки про ефективність тренувань та результати, що дозволяє іншим спортсменам та тренерам отримувати додаткову інформацію перед плануванням тренувальних програм.

Недоліки:

1. Деякі користувачі можуть відзначити складність використання платформи на початковому етапі через велику кількість функцій та опцій.

2. TrainingPeaks може бути не досить інтегрованим з іншими спортивними додатками або пристроями, що може створювати додаткові труднощі для користувачів, які використовують інші сервіси.
3. Деякі спортсмени можуть відчувати перевантаження інформацією, оскільки платформа може надавати велику кількість аналітичних даних, які потребують обробки та інтерпретації.

TeamBuildr - це інноваційна платформа, спеціалізована на плануванні та виконанні тренувальних програм для спортсменів та тренерів. Заснована у XXI столітті, TeamBuildr швидко завоювала популярність серед різних спортивних команд і спортсменів завдяки своїм передовим можливостям та інтуїтивному інтерфейсу.

Інтерфейс сайту можемо побачити на рисунку 2.

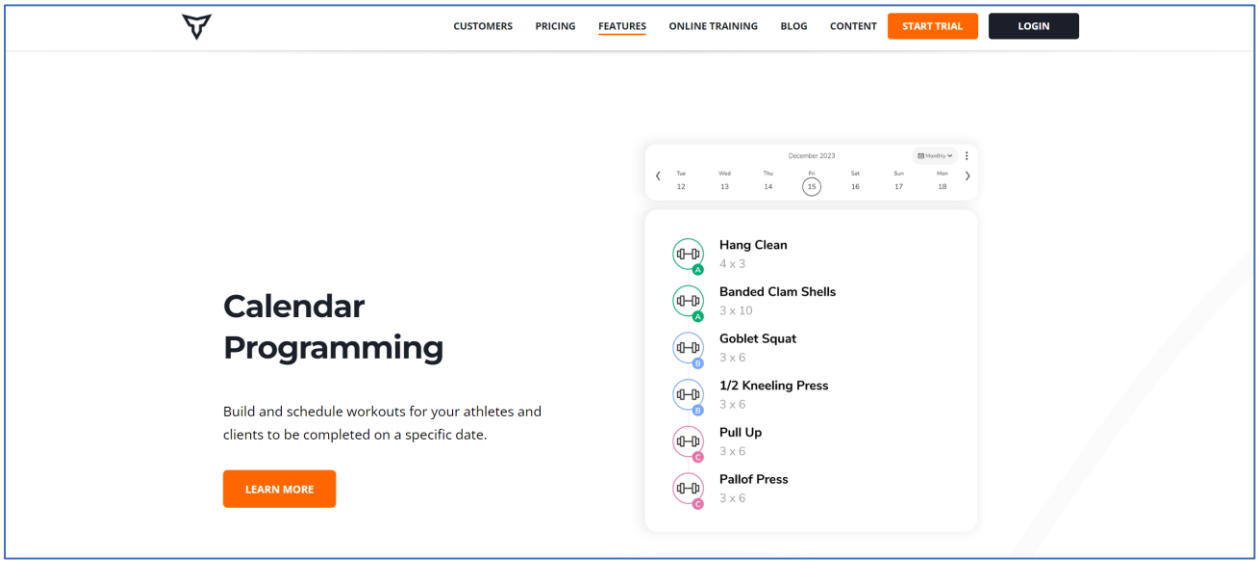

Рисунок 2 — Інтерфейс TeamBuilder

Переваги:

1. TeamBuildr має велику базу користувачів, серед яких професійні спортсмени та їхні тренери, що свідчить про його вплив та авторитет у спортивній громадськості.

2. Інтерфейс веб-платформи та мобільного додатка TeamBuildr відрізняється чіткою організацією та інтуїтивним дизайном, що робить їх легкими у використанні навіть для новачків.
3. Спортсмени можуть швидко створювати та налаштовувати свої тренувальні програми за допомогою TeamBuildr, тоді як тренери мають можливість моніторити та аналізувати прогрес своїх підопічних, сприяючи покращенню спортивних результатів.
4. Користувачі можуть залишати відгуки про ефективність тренувань та досягнення результатів, що створює можливість для інших спортсменів та тренерів отримувати додаткову інформацію перед плануванням власних тренувальних програм.

Недоліки:

1. Деякі користувачі можуть відзначити, що початкова настройка та освоєння функцій TeamBuildr можуть бути складними через велику кількість опцій, що вимагає додаткового часу для оволодіння.
2. TeamBuildr може бути обмеженим в інтеграції з іншими спортивними додатками або пристроями, що ускладнює спільне використання з іншими сервісами та пристроями.
3. Деякі спортсмени можуть відчувати перевантаження інформацією через велику кількість аналітичних даних, які надає TeamBuildr, що може вплинути на їхній спортивний розвиток.|

Trainerize - це інноваційна платформа, спеціалізована на плануванні та виконанні тренувальних програм для спортсменів та тренерів. Заснована у XXI столітті, Trainerize швидко завоювала популярність серед спортивних команд і спортсменів завдяки своїм передовим можливостям та інтуїтивному інтерфейсу.

Інтерфейс сайту можемо побачити на рисунку 3.

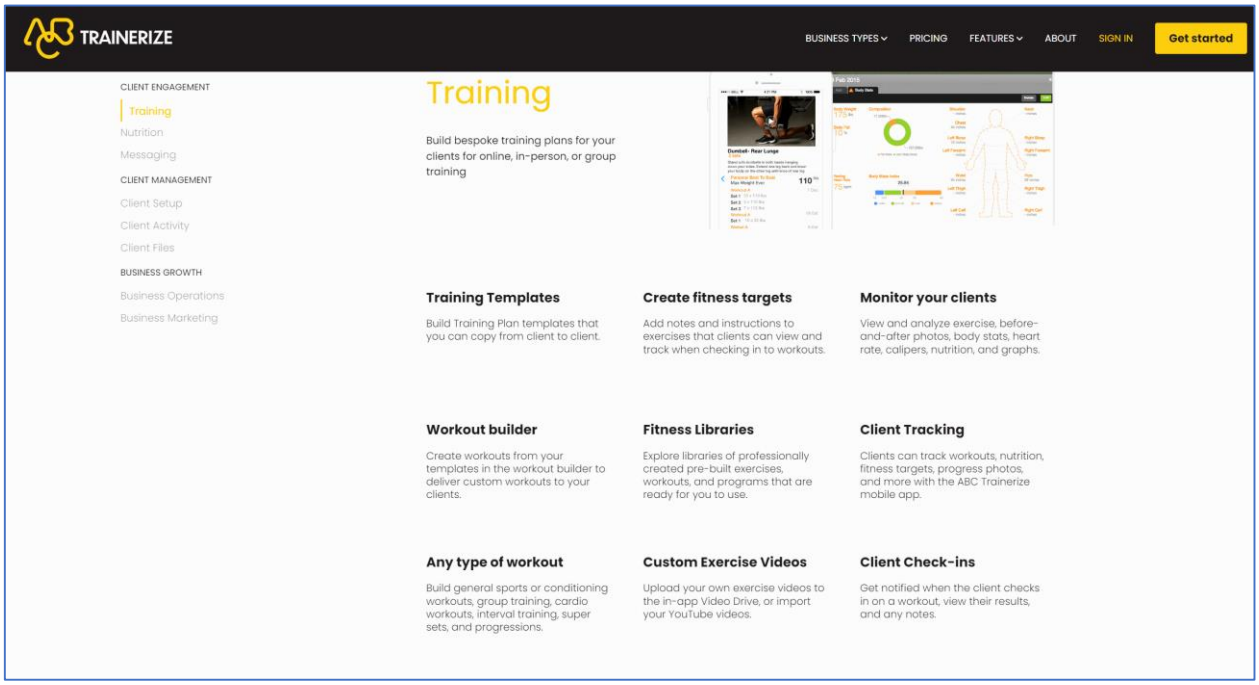

Рисунок 3 — Інтерфейс Trainerize

Переваги:

1. Trainerize пропонує широкий вибір інструментів для створення та налаштування індивідуальних тренувальних програм, які враховують потреби кожного спортсмена.
2. Платформа має зручний веб-інтерфейс та мобільний додаток, що дозволяє легко взаємодіяти з нею як тренерам, так і спортсменам з будь-якого пристрою.
3. Спортсмени можуть зручно виконувати тренувальні завдання, записувати прогрес та спілкуватися з тренерами через платформу, що сприяє покращенню їхніх спортивних результатів.
4. Trainerize надає можливість відстежувати фізичні показники та аналізувати дані про тренування, що дозволяє тренерам налаштовувати програми для максимальної ефективності.

Недоліки:

1. Деякі користувачі можуть відчувати перевантаження від великої кількості функцій та опцій, що може стати перешкодою для їхнього швидкого освоєння платформи.
2. Trainerize може бути менш інтегрованим з іншими спортивними додатками порівняно з іншими аналогічними платформами, що може ускладнити взаємодію користувачів з різними сервісами.
3. Деякі функції платформи можуть вимагати додаткової оплати або доступні лише у преміум-версії, що може бути недоліком для деяких користувачів.

Продивившись та проаналізувавши програмні продукти-аналоги можемо побачити які переваги має комп'ютерна система оптимізації фізичних навантажень, яка базується на машинному навчанні:

Персоналізований підхід:

Система, базована на машинному навчанні, може адаптуватися до індивідуальних потреб кожного спортсмена. Вона здатна аналізувати велику кількість даних про фізичні показники, історію тренувань та інші фактори для створення оптимальних тренувальних програм, які враховують унікальні потреби та можливості кожного спортсмена.

Адаптивність до змін:

Система може автоматично адаптуватися до змін у фізичному стані спортсмена, його результатів тренувань та інших факторів. Це дозволяє швидко коригувати тренувальні програми для досягнення найкращих результатів та попередження можливих перенавантажень або травм.

Інтеграція з даними в реальному часі:

Система може інтегруватися з різноманітними датчиками та пристроями для вимірювання фізичних параметрів спортсменів в реальному часі. Це дозволяє отримувати точні дані про навантаження та реакцію організму спортсмена під час тренувань.

Аналіз великих обсягів даних:

Система може обробляти великі обсяги даних про тренування, фізичні параметри та результати, використовуючи машинне навчання для виявлення патернів та трендів. Це дозволяє зробити більш точні прогнози щодо оптимальних тренувальних програм та результатів.

Вплив на результати:

Завдяки персоналізованому підходу та аналізу великих обсягів даних, ваша система може значно підвищити ефективність тренувань та досягнення результатів для кожного спортсмена. Це дозволяє підвищити конкурентоспроможність та досягати високих спортивних досягнень.

Загалом, комп'ютерна система оптимізації фізичних навантажень спортсменів, розроблена на основі машинного навчання, має потенціал стати ключовим інструментом у сфері спортивної підготовки. Вона надає персоналізовані тренувальні програми, адаптується до змін в фізичному стані спортсменів та надає можливість аналізу великих обсягів даних для досягнення найкращих результатів. Таким чином, ця система може підвищити ефективність тренувань та допомогти спортсменам досягти високих спортивних досягнень.

1.3 Постановка завдання

Завданням даного дослідження є розробка комп'ютерної системи оптимізації фізичних навантажень для спортсменів з використанням методів машинного навчання. Основною метою є створення ефективного та інноваційного інструменту, який дозволить спортсменам та їх тренерам максимально оптимізувати тренувальний процес з урахуванням індивідуальних фізіологічних особливостей та потреб кожного спортсмена.

Для досягнення цієї мети необхідно розв'язати наступні завдання:

1. Аналіз предметної області: Дослідити сучасний стан підходів до тренування спортсменів та методів оптимізації фізичних навантажень.

2. Огляд існуючих методів рішення: Проаналізувати існуючі підходи та програмні рішення для оптимізації фізичних навантажень у спорті.
3. Вивчення та збір матеріалів, пов'язаних з темою дослідження: Провести детальний аналіз наукової та практичної літератури з питань оптимізації фізичних навантажень та використання методів машинного навчання у спорті.
4. Аналіз та оцінка сучасних технологій: Дослідити сучасні технології та інструменти, які можна використовувати для створення комп'ютерної системи оптимізації фізичних навантажень.
5. Розробка та уточнення функціональних вимог: Визначити основні функції та вимоги до системи, зокрема, можливості адаптації до індивідуальних потреб спортсменів та їх тренерів.
6. Створення архітектури системи та проєктування програмного забезпечення: Розробити структуру системи та вибрати найбільш підходящі технології для реалізації комп'ютерної системи оптимізації фізичних навантажень.
7. Розробка програмного забезпечення та його тестування: Реалізувати програмне забезпечення, яке відповідає вимогам та функціональним можливостям, та провести його тестування для перевірки працездатності та ефективності.

Оформлення звіту: Описати процес розробки, використані технології та інструменти, результати тестування та висновки щодо подальшого вдосконалення системи.

Таким чином, результатом дослідження буде комп'ютерна система оптимізації фізичних навантажень для спортсменів, яка буде базуватися на сучасних методах машинного навчання та забезпечуватиме ефективний моніторинг та аналіз тренувального процесу.

2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Аналіз сучасних технологій для створення клієнтської частини веб застосунків

Перед створенням веб-застосунку важливо обрати фреймворк для розробки його клієнтської частини. На сьогоднішній день доступно багато фреймворків, але основні — React, Vue та Angular. Для кращого розуміння, який з них найкраще підійде для створення Комп'ютерної системи оптимізації фізичних навантажень спортсменів, розглянемо кожен з них ближче.

ReactJS, один з найбільш популярних JavaScript-фреймворків для розробки інтерфейсу користувача (UI), створений Facebook для швидкої, ефективної та простої розробки веб-додатків.

Основні особливості ReactJS:

1. Віртуальний DOM: ReactJS використовує віртуальний DOM, що дозволяє ефективно оновлювати лише необхідні елементи веб-сторінки, зменшуючи навантаження на браузер та покращуючи продуктивність додатка.
2. Компонентна архітектура: ReactJS базується на компонентній архітектурі, яка дозволяє розбивати інтерфейс на невеликі, повторно використовувані компоненти. Це спрощує розробку, тестування та підтримку додатків.
3. JSX: ReactJS використовує JSX (розширений синтаксис JavaScript), що дозволяє писати HTML-подібний код безпосередньо в JavaScript-файлах. Це спрощує створення інтерфейсу та полегшує розробку.
4. Односторінкові додатки: ReactJS добре підходить для створення односторінкових додатків, оскільки дозволяє

швидко змінювати вміст сторінок без перезавантаження сторінки.

Переваги ReactJS:

1. Ефективність: Використання віртуального DOM та розподілення обов'язків між клієнтом і сервером забезпечує швидку реакцію додатка і поліпшує продуктивність.
2. Компонентна архітектура: Модульна компонентна архітектура ReactJS сприяє повторному використанню коду, зручному управлінню станом та підтримці додатків.
3. Велика спільнота розробників: ReactJS має велику спільноту розробників, що підтримують інструменти, бібліотеки та документацію. Це дозволяє швидко знаходити рішення на випадки, коли виникають питання або проблеми.
4. Розширюваність: ReactJS дозволяє легко інтегрувати сторонні бібліотеки та інструменти для розширення можливостей фреймворку.

Недоліки ReactJS:

1. Високе використання пам'яті: Використання віртуального DOM може привести до високого використання пам'яті, особливо для великих додатків. Однак це зазвичай не є серйозною проблемою, якщо застосовуються ефективні підходи до оптимізації.
2. Відсутність стандартних рішень: ReactJS не має вбудованих рішень для багатьох завдань, таких як маршрутизація, керування станом або форми. Це може вимагати використання сторонніх бібліотек або написання власного коду.
3. Час ініціалізації: Ініціалізація ReactJS може займати певний час через завантаження додаткових файлів та виконання певних операцій.

Незважаючи на недоліки, ReactJS залишається одним з найпопулярніших фреймворків завдяки своїм перевагам у швидкості, ефективності та модульності. Він широко використовується в індустрії веб-розробки та має велику екосистему інструментів та бібліотек, які допомагають розробникам ефективно створювати складні та масштабовані додатки.

У порівнянні з іншими фреймворками, React виявляється досить конкурентоспроможним. Його ефективність та зручність у використанні роблять його першим вибором для багатьох розробників. Однак важливо враховувати конкретні потреби проекту та рівень володіння командою, оскільки інші фреймворки, такі як Vue та Angular, також можуть бути дуже ефективними залежно від контексту.

Vue.js є сучасним JavaScript-фреймворком для створення інтерфейсу користувача (UI). Він створений для спрощення розробки веб-додатків та створення ефективних інтерфейсів. Основні особливості Vue.js:

1. Прогресивність: Vue.js розроблений з урахуванням прогресивного підходу, що дозволяє поетапно впроваджувати його в існуючі проекти.
2. Компонентна архітектура: Vue.js має гнучку компонентну архітектуру, що дозволяє розбивати інтерфейс на невеликі, повторно використовувані компоненти.
3. Двостороннє зв'язування даних: Vue.js надає можливість зв'язувати дані між компонентами та шаблонами за допомогою двостороннього зв'язування.
4. Директиви: Vue.js має великий набір вбудованих директив, які дозволяють легко маніпулювати DOM та реагувати на події.

Переваги Vue.js:

1. Легкість вивчення: Vue.js має простий синтаксис, що спрощує процес вивчення та розробки.
2. Флексібельність: Vue.js дозволяє використовувати тільки необхідні функції та бібліотеки, що полегшує розробку.

3. Екосистема: Vue.js має широку екосистему додаткових бібліотек та інструментів, що полегшує розробку та підтримку додатків.
4. Висока продуктивність: Vue.js забезпечує високу продуктивність завдяки використанню Virtual DOM та оптимізованих алгоритмів оновлення. Недоліки Vue.js:
5. Менша популярність: У порівнянні з іншими фреймворками, Vue.js має меншу кількість розробників та активність спільноти.
6. Обмежені можливості: У деяких аспектах Vue.js може мати обмежені можливості порівняно з іншими фреймворками.
7. Відсутність готових рішень: У порівнянні з деякими іншими фреймворками, Vue.js може мати менше готових рішень для певних завдань. Загалом, Vue.js є потужним і гнучким фреймворком, який підходить для широкого спектру проєктів від невеликих до складних.

Переваги Vue.js:

1. Легкість вивчення: Vue.js має простий синтаксис, що дозволяє швидко оволодіти його основами навіть початківцям у розробці.
2. Флексібельність: Фреймворк дозволяє розробникам використовувати лише необхідні функції та бібліотеки, зменшуючи витрати ресурсів на непотрібні компоненти.
3. Екосистема: Vue.js має широку екосистему з багатьма додатковими бібліотеками та інструментами, що допомагають спростити розробку та підтримку додатків.
4. Висока продуктивність: Завдяки використанню Virtual DOM та оптимізованим алгоритмам оновлення, Vue.js забезпечує високу продуктивність та швидкодію додатків.

Недоліки Vue.js:

1. Менша популярність: Порівняно з іншими фреймворками, Vue.js має меншу кількість розробників та меншу активність у спільноті.
2. Обмежені можливості: У деяких аспектах Vue.js може мати обмежені можливості порівняно з іншими фреймворками.
3. Відсутність готових рішень: У порівнянні з деякими іншими фреймворками, Vue.js може мати менше готових рішень для певних завдань.

Vue.js є потужним і гнучким фреймворком, який легко вивчається, має високу продуктивність та велику екосистему. Він підходить для широкого спектру проєктів, від невеликих односторінкових додатків до складних корпоративних рішень. Однак, необхідно враховувати його меншу популярність та обмежені можливості у порівнянні з деякими конкуруючими фреймворками.

Angular, розроблений зусиллями Google, представляє собою високорівневий інструмент для тих, хто прагне створювати потужні та витончені веб-додатки. Він вражає своєю розмаїтістю функціоналу та гнучкістю в налаштуванні.

Особливості Angular:

1. TypeScript: В основі Angular лежить TypeScript, який пропонує розширені можливості і типізацію, що забезпечує статичний аналіз коду та підвищує його надійність.
2. Компонентна архітектура: Angular побудований на принципах компонентної архітектури, де додаток складається з повторно використовуваних компонентів, що спрощує його розробку та підтримку.
3. Директиви: Фреймворк має потужну систему директив для маніпулювання DOM та виконання додаткових функцій.

4. Відділена розмітка та логіка: Angular розділяє розмітку та логіку, використовуючи окремі шаблони для відображення даних.

Переваги Angular:

1. Широкий функціонал: Angular надає великий набір інструментів для розробки, включаючи готові рішення для різних завдань.
2. Чітка структура: Фреймворк визначає чіткі правила організації коду, що полегшує спільну роботу в команді та підтримку проєктів.
3. Висока продуктивність: Завдяки використанню віртуального DOM та інших оптимізацій, Angular забезпечує високу швидкість додатків.
4. Сильна спільнота: Фреймворк має велику спільноту розробників, яка активно підтримує розвиток і надає різноманітні ресурси.

Недоліки Angular:

1. Великий розмір: Розмір фреймворку може впливати на швидкість завантаження додатків, особливо в малих проєктах.
2. Складність вивчення: Висока складність може затримати нових розробників, які потребують більше часу для оволодіння фреймворком.
3. Жорстка структура: Angular може накладати обмеження на розробку, що ускладнює адаптацію до вже існуючих проєктів.

У підсумку, Angular є неабияким інструментом для розробки веб-додатків, здатним забезпечити якість та продуктивність. Він відрізняється широким спектром можливостей та глибокою інтеграцією з іншими інструментами, що робить його першим вибором для багатьох проєктів. Однак, для деяких випадків можуть бути значними недоліками його великий

розмір та складність вивчення, що потребує додаткових зусиль для успішного впровадження.

Для вирішення проблеми оптимізації фізичних навантажень у комп'ютерній системі спортивних досягнень було визначено, що важливим фактором є вибір підходящого фреймворка. В даному випадку було вирішено використовувати підхід ReactJS через його здатність до компонентної архітектури, що сприяє розробці, тестуванню та підтримці системи. За допомогою цієї архітектури можна створювати повторно використовувані елементи інтерфейсу, що дозволяє швидше вирішувати завдання оптимізації фізичних навантажень та забезпечує легку масштабованість проєкту.

2.2 Аналіз PostgreSql СУБД

У рамках сучасного інформаційного ландшафту, де обробка великих обсягів даних відіграє ключову роль, використання традиційних реляційних баз даних може бути недостатнім для ефективної обробки та масштабування даних. У такому контексті, PostgreSQL, як одна з найпопулярніших відкритих реляційних баз даних, відіграє важливу роль у забезпеченні ефективності та надійності обробки даних.

Реляційна база даних PostgreSQL, як база даних з відкритим вихідним кодом, відома своєю потужною реляційною моделлю та широким спектром функціональності. Її можливості включають підтримку складних запитів, транзакційну безпеку та розширені можливості індексації, що робить її привабливим вибором для різноманітних застосувань, включаючи великі корпоративні системи, веб-додатки та аналітичні платформи.

Реляційна база даних - це структурована колекція даних, організована у вигляді таблиць, які містять рядки і стовпці. Кожна таблиця представляє собою сутність (або тип даних), а кожний рядок у таблиці представляє конкретний запис даних. Реляційні бази даних базуються на принципах реляційної алгебри та теорії множин.

Основні поняття реляційної бази даних включають:

1. Таблиці: Це основні структурні одиниці реляційної бази даних, що зберігають дані у вигляді двовимірних таблиць. Кожна таблиця складається з рядків і стовпців, де кожен стовпець визначає тип даних, а кожен рядок представляє конкретний запис даних.
2. Ключі: Ключі використовуються для унікальної ідентифікації кожного рядка в таблиці. Основними типами ключів є первинний ключ (який однозначно ідентифікує кожен запис в таблиці) та зовнішній ключ (який встановлює зв'язок між двома таблицями).
3. Відносини: Відносини визначають зв'язки між різними таблицями в базі даних. Це може бути один до одного, один до багатьох або багато до багатьох відношення.
4. SQL (Structured Query Language): Це мова запитів, яка використовується для взаємодії з реляційною базою даних. З його допомогою можна створювати, змінювати та видаляти дані в таблицях, а також виконувати складні запити для отримання необхідної інформації.
5. Нормалізація: Це процес структурування бази даних для уникнення аномалій та забезпечення ефективного зберігання та оновлення даних. Нормалізація допомагає уникнути дублювання даних та забезпечує консистентність бази даних.
6. У загальному, реляційні бази даних є одним з основних типів баз даних, які використовуються в сучасних системах для організації та управління даними. Вони забезпечують надійність, ефективність та легкість використання для різноманітних потреб бізнесу та розробки програмного забезпечення.

PostgreSQL — це потужна реляційна база даних з відкритим вихідним кодом, яка використовує мову запитів SQL для керування та маніпулювання даними. Основні особливості PostgreSQL включають:

1. Розширеність: PostgreSQL підтримує широкий спектр функцій та типів даних, що дозволяє розробникам створювати складні бази даних з різноманітними вимогами.
2. Надійність: Безпека даних та надійність є пріоритетом у PostgreSQL. Вона має механізми резервного копіювання, транзакцій та відновлення даних для запобігання втратам інформації.
3. Розширені можливості: PostgreSQL підтримує ряд розширень та додаткових модулів, які розширюють його функціональність. Це дозволяє адаптувати базу даних до конкретних потреб проєкту.
4. Підтримка стандартів: PostgreSQL дотримується стандартів SQL та включає в себе багато функцій, що роблять роботу з даними більш зручною та ефективною.
5. Спільнота та підтримка: У PostgreSQL активна спільнота розробників, яка надає підтримку, документацію та рішення для різних проблем та питань, пов'язаних з використанням бази даних.

У цілому, PostgreSQL є потужним і надійним інструментом для роботи з реляційними базами даних, який використовується у різних сферах від веб-розробки до аналізу даних.

2.3 Огляд REST API

REST API є важливим інструментом у веб-розробці, що забезпечує зв'язок між клієнтськими та серверними додатками. Ця технологія відіграє велике значення для програмістів, оскільки дозволяє створювати ефективну та гнучку взаємодію між компонентами системи. У нашому огляді ми розглянемо основні принципи та переваги використання REST API.

REST API (Representational State Transfer Application Programming Interface) - це архітектурний стиль, що використовується для створення веб-служб на базі протоколу HTTP. Він визначає набір правил та обмежень, які регулюють структуру та спосіб взаємодії між клієнтами та серверами.

Основною ідеєю REST API є використання різних HTTP-методів (GET, POST, PUT, DELETE) для виконання операцій з ресурсами, які представлені в URL-адресі.

Основні концепції REST API:

1. Ресурси: Ресурси є ключовими елементами REST API і представляють собою конкретні об'єкти чи дані, доступ до яких можна отримати через URL-адресу. Кожен ресурс має свій унікальний ідентифікатор і може перебувати в різних станах (наприклад, "користувач" або "продукт").
2. HTTP-методи: REST API використовує різні HTTP-методи для виконання операцій з ресурсами. Наприклад, метод GET використовується для отримання даних, метод POST - для створення нових ресурсів, PUT - для оновлення існуючих ресурсів, DELETE - для видалення ресурсів.
3. Представлення: Дані передаються між клієнтом та сервером у форматі, який може бути легко розуміти та обробити. Зазвичай використовуються формати, такі як JSON або XML, які забезпечують зручну інтерпретацію та обробку даних.

Переваги REST API:

1. Стандартизація: REST API використовує HTTP, що є стандартом веб-розробки, що спрощує розуміння та використання його принципів.
2. Масштабованість: REST API легко масштабується для обслуговування багатьох клієнтів зі збереженням стану сервера, що робить його ідеальним для великих та розподілених систем.
3. Кешування: REST API підтримує використання механізмів кешування HTTP, що дозволяє зменшити навантаження на сервер та підвищити продуктивність застосунків.

REST API є незамінним інструментом для веб-розробки, який дозволяє ефективно забезпечувати комунікацію між клієнтськими та серверними

додатками. Він спрощує розробку та підтримку програмного забезпечення, створюючи стандартизований інтерфейс для обміну даними за допомогою HTTP-протоколу. REST API дозволяє створювати масштабовані та надійні системи, а також підтримує різні формати даних, такі як JSON або XML. Завдяки своїм перевагам, REST API широко використовується у різних сферах веб-розробки, що робить його невід'ємною складовою сучасного програмного забезпечення.

2.4 Огляд алгоритму машинного навчання для прогнозування майбутніх навантажень

Машинне навчання - це галузь штучного інтелекту, яка досліджує розробку алгоритмів і моделей, які дають комп'ютерам здатність вчитися з даних і виконувати завдання без явного програмування. Основна ідея машинного навчання полягає в тому, щоб розвивати алгоритми, які можуть розпізнавати закономірності в даних та використовувати їх для прийняття рішень.

Існує три основні типи завдань у машинному навчанні:

Навчання з учителем (Supervised Learning): У цьому типі навчання алгоритм навчається на основі пари вхідних даних та відповідних міток. Наприклад, якщо маємо набір зображень автомобілів, і для кожного зображення є мітка, що вказує, чи є це зображення автомобілем чи ні, то алгоритм навчання навчатиме модель класифікації, яка може передбачати, чи є автомобіль на новому зображенні.

Навчання без учителя(Unsupervised Learning): Тут модель навчається без міток або будь-якої форми учителя. Алгоритм намагається знайти приховані структури в даних. Наприклад, алгоритми кластеризації групують схожі об'єкти разом без будь-яких заздалегідь відомих категорій.

Підготовлене навчання (Reinforcement Learning): У цьому типі навчання агент навчається взаємодіяти з оточенням, проводячи послідовні дії. Він отримує нагороду або покарання в залежності від своїх дій, і його метою є максимізація нагороди. Цей тип навчання застосовується, наприклад, у розробці ігрових алгоритмів.

Машинне навчання здатне розв'язувати широкий спектр завдань, від простих класифікаційних задач до складних прогнозувань та аналізу великих обсягів даних. Воно застосовується у багатьох галузях, таких як медицина, фінанси, транспорт, маркетинг, техніка, наука та багато інших.

Згадуючи про ML.NET, ми відразу переходимо до світу машинного навчання — цієї галузі штучного інтелекту, яка створює системи, здатні навчатися на основі даних та виконувати завдання без явного програмування. ML.NET виходить за межі лише розробки алгоритмів, вона є також інтеграційною платформою, яка спрощує процес створення, навчання та використання моделей машинного навчання, особливо для розробників, які працюють у середовищі .NET.

Ця бібліотека підтримує широкий спектр завдань машинного навчання, включаючи класифікацію, регресію, кластеризацію та інші. Це означає, що з ML.NET можна створювати моделі для різноманітних сценаріїв, починаючи від передбачення майбутніх значень до виявлення складних закономірностей у даних.

Прогнозування ваги (PredictWeight):

Цей метод, код якого можна побачити на Лістингу 1, призначений для передбачення ваги на основі вхідних даних про кількість повторень, рівень сприйняття навантаження та день тренування.

Ось як він працює:

1. Спочатку створюється двигун прогнозування за допомогою моделі WeightPrediction, яка містить у собі пайплайн обробки даних та навчену модель.
2. Потім метод передає вхідні дані про кількість повторень, рівень сприйняття навантаження та день тренування у двигун прогнозування.
3. Двигун використовує навчену модель для передбачення ваги на основі цих вхідних даних.
4. Останнім кроком є повернення передбаченої ваги.

Лістинг *1 Код передбачення ваги*

```
  public float PredictWeight(int numberOfRepetitions,
float rateOfPerceivedExertion, float day)
  {
```

```
      var predictionEngine = 
_mlContext.Model.CreatePredictionEngine<TrainingData, 
WeightPrediction>(_weightModel);
      var prediction = predictionEngine.Predict(new 
TrainingData
      {
          NumberOfRepetitions = numberOfRepetitions,
          RateOfPerceivedExertion = 
rateOfPerceivedExertion,
          Day = day
      });

      return prediction.PredictedWeight;
  }
```

Прогнозування кількості повторень (PredictRepetitions):

Цей метод, код якого можна побачити на Лістингу 2, призначений для передбачення кількості повторень на основі вхідних даних про вагу, рівень сприйняття навантаження та день тренування.

Ось як він працює:

1) Спочатку створюється двигун прогнозування за допомогою моделі RepetitionsPrediction, яка містить у собі пайплайн обробки даних та навчену модель.
2) Потім метод передає вхідні дані про вагу, рівень сприйняття навантаження та день тренування у двигун прогнозування.
3) Двигун використовує навчену модель для передбачення кількості повторень на основі цих вхідних даних.
4) Останнім кроком є повернення передбаченої кількості повторень.

Лістинг *2. Код передбачення кількості повторень*

```
public float PredictRepetitions(float weight, float 
rateOfPerceivedExertion, float day)
```

```
    {
        var predictionEngine =
_mlContext.Model.CreatePredictionEngine<TrainingData,
RepetitionsPrediction>(_repetitionsModel);
        var prediction = predictionEngine.Predict(new
TrainingData
        {
            Weight = weight,
            RateOfPerceivedExertion =
rateOfPerceivedExertion,
            Day = day
        });

        return prediction.PredictedRepetitions;
    }
```

Ці методи дозволяють легко використовувати навчені моделі для прогнозування ваги та кількості повторень залежно від введених характеристик тренування.

3 РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ ОПТИМІЗАЦІЇ ФІЗИЧНИХ НАВАНТАЖЕНЬ СПОРТСМЕНІВ.

3.1 Опис предметної області

Комп'ютерна система оптимізації фізичних навантажень спортсменів є інноваційним рішенням для автоматизації та оптимізації тренувальних процесів у спортивних дисциплінах. Ця система використовує сучасні технології для підтримки тренувального процесу та досягнення максимальних результатів для спортсменів та їх тренерів.

Однією з ключових складових комп'ютерної системи є алгоритмічна модель, яка аналізує дані про фізичні параметри спортсменів, їх стан готовності та історію тренувань. Ця модель дозволяє генерувати індивідуальні тренувальні програми та рекомендації з метою оптимізації навантажень та підвищення ефективності тренувань.

Крім того, комп'ютерна система використовує інтерактивний інтерфейс для взаємодії з тренерами та спортсменами. Цей інтерфейс дозволяє спортсменам вносити дані про своє фізичне становище та результати тренувань, а також отримувати рекомендації щодо оптимальних навантажень та вправ.

Комп'ютерна система оптимізації фізичних навантажень також включає в себе модуль моніторингу за фізичними показниками спортсменів. Цей модуль відстежує зміни у фізичних показниках та реагує на них, щоб адаптувати тренувальні програми та максимізувати результати.

Загалом, комп'ютерна система оптимізації фізичних навантажень спортсменів дозволяє підвищити ефективність тренувань, зменшити ризики травм та підвищити результативність спортивних досягнень. Це інноваційне рішення в сфері спортивної науки, яке відкриває нові можливості для досягнення високих спортивних результатів.

3.2 Архітектура системи

Система "Оптимізація фізичних навантажень спортсменів", реалізована з використанням технологій .NET Web API, EF Core, PostgreSQL та React, втілює в собі сучасний підхід до організації та автоматизації тренувальних процесів спортсменів.

Клієнтська частина системи розроблена з використанням ReactJS, що надає високу швидкість роботи та гнучкість в управлінні користувацьким інтерфейсом. ReactJS використовує реактивний підхід, що дозволяє плавно оновлювати вміст сторінок без перезавантаження, забезпечуючи при цьому позитивний досвід користувачам під час взаємодії з системою.

Серверна частина системи базується на .NET Web API, яка використовує EF Core для взаємодії з базою даних PostgreSQL. Цей рівень забезпечує надійну та ефективну обробку запитів користувачів, використовуючи масштабованість та швидкість роботи .Net.

Система реалізована з використанням архітектурного підходу з чотирма рівнями (4-layer architecture), що дозволяє відокремити функціональні обов'язки та забезпечити легкість розширення та підтримки коду. (див. Рисунок 4) Ось детальний опис кожного з цих рівнів:

Happy.WebApi:

- Цей рівень відповідає за веб-інтерфейс та взаємодію з клієнтами через API.
- Він включає в себе контролери, які обробляють HTTP-запити та взаємодіють з бізнес-логікою системи.
- Контролери перетворюють HTTP-запити на внутрішні об'єкти та передають їх до сервісного рівня для подальшої обробки.

Happy.Service:

- Цей рівень містить бізнес-логіку додатку та виконує всі операції, пов'язані з управлінням фізичними навантаженнями спортсменів.

- Сюди входять класи та сервіси, які реалізовують функціональність додатку, таку як оптимізація навантажень, обробка даних та взаємодія з іншими компонентами системи.

Happy.Infrastructure:

- Цей рівень відповідає за доступ до даних та роботу з базою даних.
- Включає у себе класи, що взаємодіють з EF Core або іншими ORM-інструментами для доступу до даних.
- Тут також можуть знаходитись різноманітні інструменти для роботи з різними джерелами даних (наприклад, зовнішніми API).

Happy.Domain:

- Цей рівень містить моделі даних та бізнес-об'єкти, які представляють собою сутності та логічні компоненти системи.
- Він є серцем системи і визначає структуру даних, бізнес-правила та логіку обробки даних.
- Моделі в цьому рівні використовуються всіма іншими рівнями для обміну даними та взаємодії між компонентами системи.

Кожен з цих рівнів має свої чітко визначені обов'язки та взаємодіє з іншими рівнями згідно з принципами чистої архітектури. Це дозволяє системі бути масштабованою, легко змінювати та підтримувати, оскільки кожен рівень відповідає лише за конкретний аспект функціональності.

Допоміжні бібліотеки Happy.Common та Happy.Utils забезпечують загальні функції та утиліти для спільного використання та полегшення розробки.

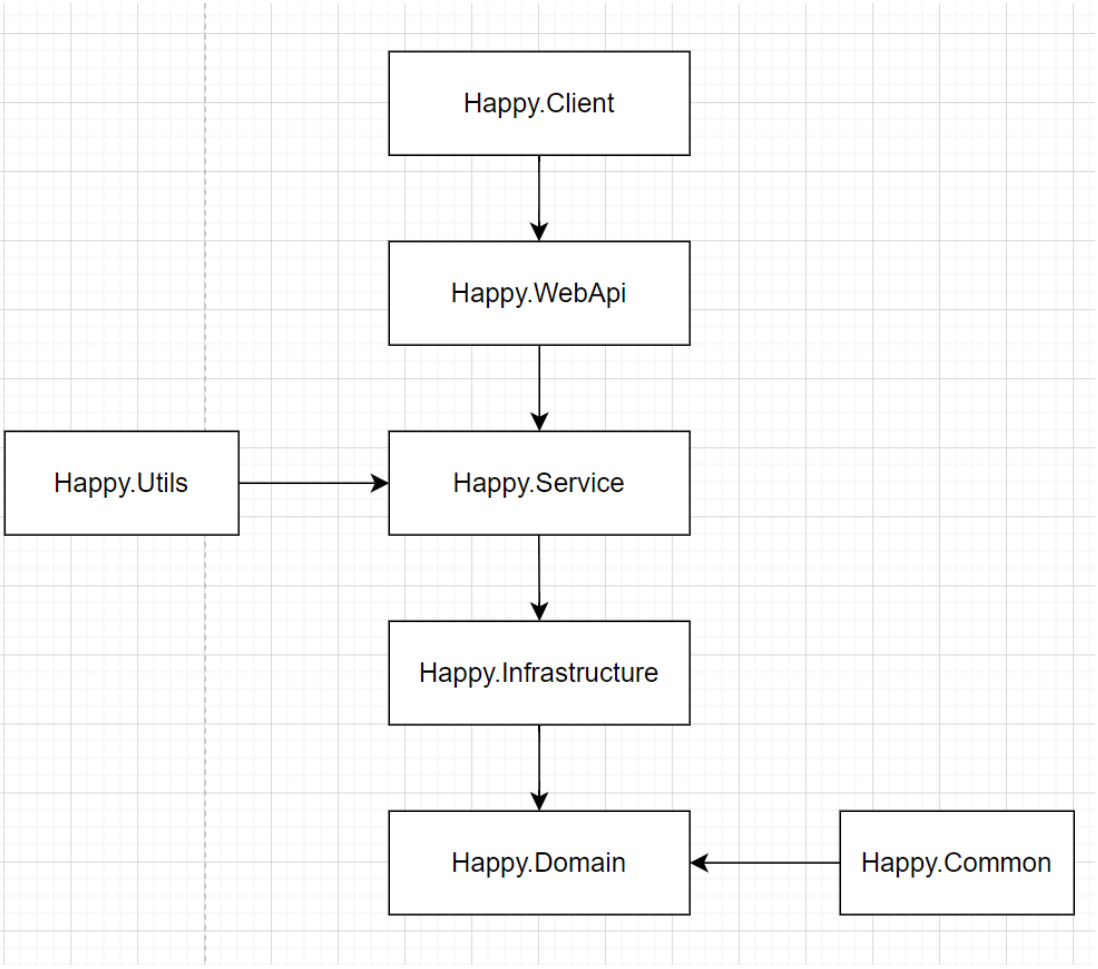

Рисунок 4 — Архітектура системи

Ця архітектура дозволяє створювати ефективну та масштабовану систему для оптимізації фізичних навантажень спортсменів, забезпечуючи високу продуктивність, надійність та зручність використання для користувачів.

Користувач може взаємодіяти з системою "Оптимізація фізичних навантажень спортсменів" через веб-інтерфейс, який надається за допомогою Web API. Для цього він може виконувати HTTP-запити з браузера або іншого клієнтського застосунку. Протокол HTTP використовується для комунікації між користувачем і сервером.

Після виконання запиту до Web API, користувач отримає відповідь у форматі JSON(див. Рис 6), яка містить інформацію про результат виконання запиту. Наприклад, у відповіді може бути зазначено статус успішного

створення нового навантаження або список усіх наявних навантажень, залежно від типу запиту.

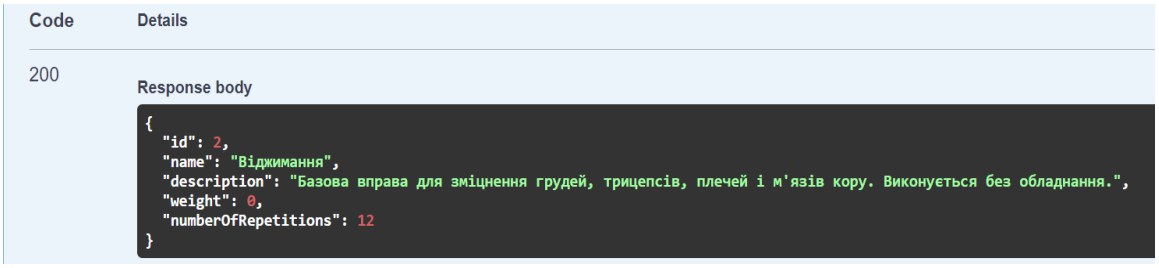

Рисунок 6 — Відповідь в Json форматі на запит користувача

3.3 Функціональні вимоги системи

Нижче наведені вимоги для розробки та імплементації застосунку "Оптимізація фізичних навантажень спортсменів" з використанням .NET Web API, ReactJS, EF Core та PostgreSQL.

1. Аутентифікація, реєстрація та перегляд інформації про поточного користувача:
   - Система повинна надавати можливість користувачам аутентифікуватися та реєструватися для отримання доступу до функцій додатку, перегляду особистих даних, налаштувань та історії тренувань.
   - Користувачі повинні мати можливість створювати облікові записи з унікальними ідентифікаторами та паролями.
2. Отримання поточного тижня:
   - Система повинна надавати користувачам інформацію про поточний тиждень тренувань, включаючи дати та заплановані навантаження.
3. Створення, редагування, видалення та перегляд вправ:
   - Користувачі повинні мати можливість створювати нові вправи, редагувати наявні, видаляти та переглядати деталі кожної вправи.

- Вправи повинні містити інформацію про назву, опис, тривалість, інтенсивність та інші характеристики.

4. Створення, редагування, видалення та перегляд комплексів:
   - Система повинна дозволяти користувачам створювати тренувальні комплекси, редагувати їх, видаляти та переглядати деталі кожного комплексу.
   - Тренувальні комплекси повинні включати набір вправ та їх послідовність.
5. Створення, редагування, видалення та перегляд прогресу:
   - Користувачі повинні мати можливість створювати записи про свій прогрес, редагувати їх, видаляти та переглядати історію тренувань.
   - Прогрес повинен включати дані про виконані вправи, кількість повторень, вагу, рівень навантаження та інші важливі метрики.
6. Отримання тренувальних рекомендацій:
   - Система повинна надавати користувачам тренувальні рекомендації на основі майбутніх навантажень, використовуючи моделі машинного навчання.
   - Рекомендації повинні враховувати поточний стан користувача, його прогрес та інші релевантні дані для оптимізації тренувального процесу.

Ці функціональні вимоги визначають основні можливості застосунку для оптимізації фізичних навантажень спортсменів, які необхідно реалізувати для задоволення потреб користувачів та забезпечення ефективності роботи системи.

3.4 Проектування комп’ютерної системи оптимізації фізичних навантажень спортсменів

Одним з етапів проєктування застосунку "Оптимізація фізичних навантажень спортсменів" є створення діаграм використання (Use-Case) для визначення функціональності системи.

Для візуалізації функціональності та взаємодії між акторами (користувачами) і системою була створена Use-Case діаграма.

Елементи Use-Case діаграми включають:

1. Актори:
    - Користувач: основний актор, який взаємодіє з системою.
    - Адміністратор: актор, який відповідає за керування вмістом системи.
2. Використані сценарії:
    - Аутентифікація та реєстрація: користувач реєструється в системі або входить в неї, надаючи необхідну інформацію (наприклад, ім'я, електронну пошту тощо).
    - Перегляд інформації про поточного користувача: користувач має можливість переглядати свої особисті дані та налаштування.
    - Отримання поточного тижня: користувач отримує інформацію про поточний тиждень тренувань.
    - Перегляд вправ: користувач може переглядати доступні вправи.
    - Створення, редагування та видалення вправ: адміністратор може створювати нові вправи, редагувати їх або видаляти.
    - Перегляд комплексів: користувач може переглядати доступні комплекси вправ.

- Створення, редагування та видалення комплексів: адміністратор може створювати нові комплекси вправ, редагувати їх або видаляти.
- Перегляд прогресу: користувач може переглядати свій прогрес.
- Створення, редагування та видалення прогресу: адміністратор може створювати записи про прогрес, редагувати їх або видаляти.
- Отримання тренувальних рекомендацій: користувач отримує рекомендації щодо майбутніх навантажень на основі машинного навчання.

3. Взаємодія між акторами та сценаріями:
   - Користувач взаємодіє з системою, щоб аутентифікуватися, переглядати інформацію про себе, отримувати інформацію про поточний тиждень тренувань, переглядати вправи, комплекси та прогрес, а також отримувати тренувальні рекомендації.
   - Адміністратор взаємодіє з системою для створення, редагування та видалення вправ, комплексів і записів про прогрес.
   - Система обробляє запити користувача і адміністратора, надає необхідну інформацію про тренування та поточний стан, дозволяє керувати вправами, комплексами та прогресом, а також генерує рекомендації з використанням алгоритмів машинного навчання.

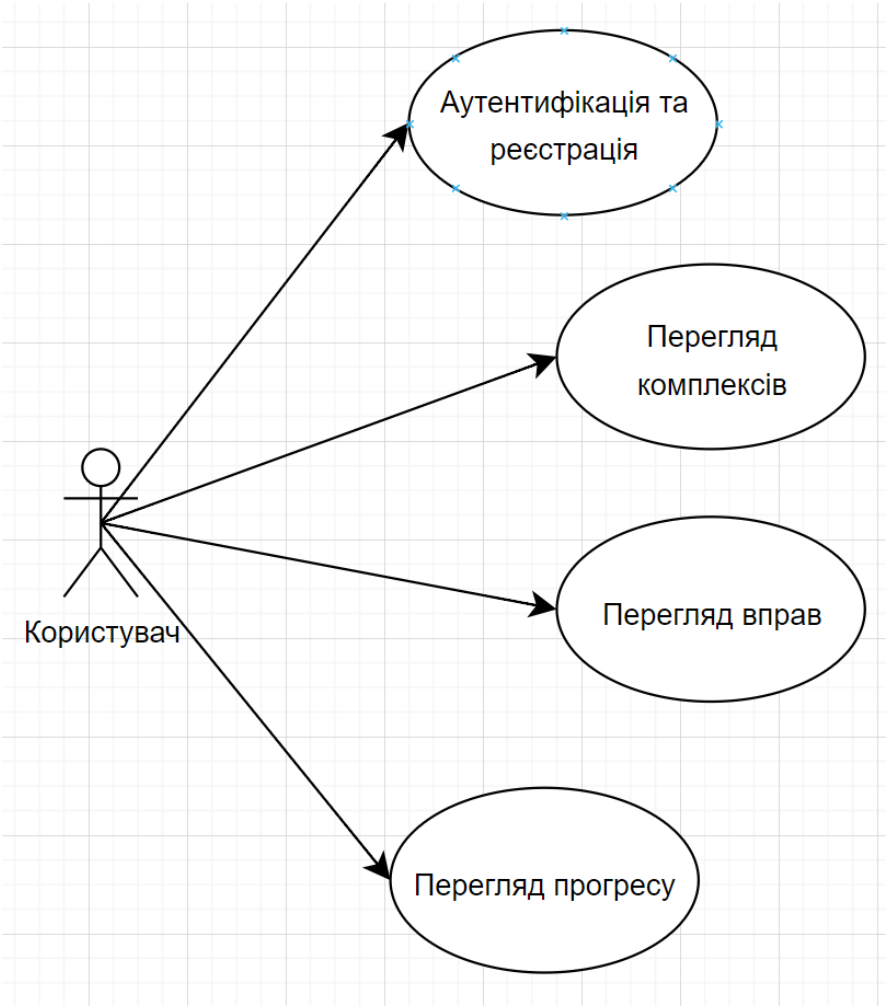

Рисунок 7 — UseCase діаграма можливостей взаємодії користував с системою

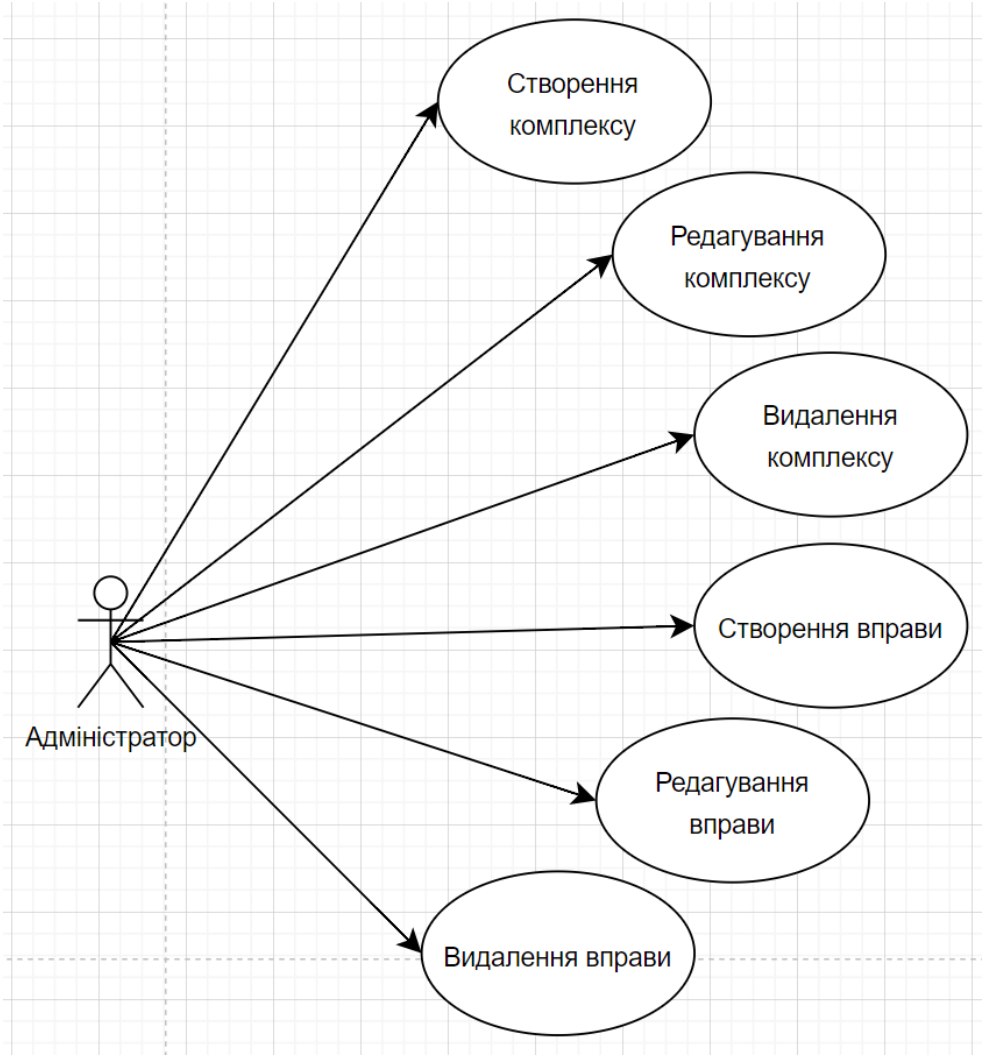

Рисунок 8 — UseCase діаграма можливостей взаємодії адміністратора с системою

База даних PostgreSQL для застосунку "Оптимізація фізичних навантажень спортсменів" містить декілька таблиць, які забезпечують збереження необхідної інформації. Нижче наведено структуру основних таблиць бази даних.

Таблиця AspNetUsers

- id: унікальний ідентифікатор користувача.
- userName: ім'я користувача.
- NormalizedUserName: нормалізоване ім'я користувача.
- Email: електронна пошта користувача.
- NormalizedEmail: нормалізована електронна пошта.

- EmailConfirmed: підтвердження електронної пошти.
- PasswordHash: захешований пароль користувача.
- SecurityStamp: штамп безпеки для користувача.
- ConcurrencyStamp: штамп для обробки одночасних змін.
- PhoneNumber: номер телефону користувача.
- PhoneNumberConfirmed: підтвердження номера телефону.
- TwoFactorEnabled: увімкнення двофакторної аутентифікації.
- LockoutEnd: дата закінчення блокування користувача.
- LockoutEnabled: увімкнення блокування користувача.
- AccessFailedCount: кількість невдалих спроб доступу.

Таблиця AspNetUserRoles

- userId: ідентифікатор користувача.
- roleId: ідентифікатор ролі.

Таблиця AspNetRoles

- Name: назва ролі.
- NormalizedName: нормалізована назва ролі.
- ConcurrencyStamp: штамп для обробки одночасних змін.

Таблиця AspNetRoleClaims

- Id: унікальний ідентифікатор.
- roleId: ідентифікатор ролі.
- ClaimType: тип заявки.
- ClaimValue: значення заявки.

Таблиця AspNetUserClaims

- Id: унікальний ідентифікатор.
- UserId: ідентифікатор користувача.
- ClaimType: тип заявки.
- ClaimValue: значення заявки.

Таблиця AspNetUserLogins

- LoginProvider: провайдер входу.

- ProviderKey: ключ провайдера.
- ProviderDisplayName: відображуване ім'я провайдера.
- UserId: ідентифікатор користувача.

Таблиця AspNetUserTokens

- UserId: ідентифікатор користувача.
- LoginProvider: провайдер входу.
- Name: ім'я токена.
- Value: значення токена.

Таблиця Progress

- Id: унікальний ідентифікатор запису.
- UserId: ідентифікатор користувача.
- ExerciseId: ідентифікатор вправи.
- Weight: вага для вправи.
- NumberOfRepetitions: кількість повторень.
- Date: дата виконання вправи.
- RateOfPerceivedExertion: оцінка суб'єктивної складності.

Таблиця Exercises

- Id: унікальний ідентифікатор вправи.
- Name: назва вправи.
- Description: опис вправи.
- Weight: вага для вправи.
- NumberOfRepetitions: кількість повторень.

Таблиця ComplexExercises

- Id: унікальний ідентифікатор запису.
- ExerciseId: ідентифікатор вправи.
- ComplexId: ідентифікатор комплексу.

Таблиця Complexes

- Id: унікальний ідентифікатор комплексу.
- Name: назва комплексу.

- TotalSets: загальна кількість підходів.
- Duration: тривалість виконання комплексу.
- WeekDay: день тижня виконання.

Таблиця ComplexWeeks

- Id: унікальний ідентифікатор запису.
- ComplexId: ідентифікатор комплексу.
- WeekId: ідентифікатор тижня.

Таблиця Week

- Id: унікальний ідентифікатор тижня.
- StartDate: дата початку тижня.
- EndDate: дата закінчення тижня.
- Year: рік.
- WeekNumber: номер тижня.

Ці таблиці, які можна побачити на рисунку 9, забезпечують збереження та управління інформацією про користувачів, їх ролі та права доступу, вправи, комплекси вправ, прогрес тренувань та тижневий розклад. Використання PostgreSQL дозволяє ефективно організувати збереження та доступ до даних, забезпечуючи масштабованість і високу продуктивність системи.

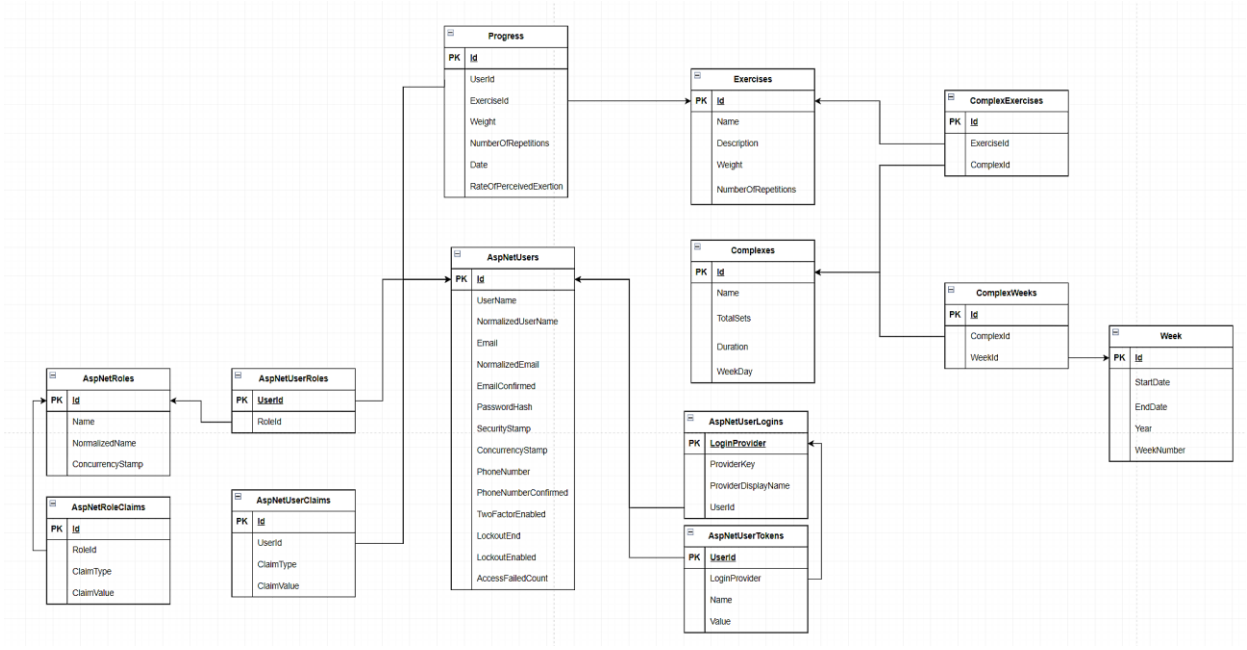

Рисунок 9 — *Схема бази даних*

Діаграма класів для застосунку "Оптимізація фізичних навантажень спортсменів" демонструє взаємозв'язки між основними класами та їх властивостями. Нижче наведено опис основних класів та їх атрибутів:

Клас Base<T>

- Цей базовий клас містить єдину властивість Id, яка є загальною для всіх класів, що його наслідують.

Клас Complex

- Name: назва комплексу.
- TotalSets: загальна кількість підходів.
- Duration: тривалість виконання комплексу.
- WeekDay: день тижня виконання.
- ComplexExercises: колекція зв'язків з вправами.
- ComplexWeeks: колекція зв'язків з тижнями.

Клас ComplexExercise

- ExerciseId: ідентифікатор вправи.
- Exercise: об'єкт вправи.
- ComplexId: ідентифікатор комплексу.
- Complex: об'єкт комплексу.

Клас ComplexWeek

- WeekId: ідентифікатор тижня.
- Week: об'єкт тижня.
- ComplexId: ідентифікатор комплексу.
- Complex: об'єкт комплексу.

Клас Exercise

- Name: назва вправи.
- Description: опис вправи.
- Weight: вага для вправи.
- NumberOfRepetitions: кількість повторень.

- ComplexExercises: колекція зв'язків з комплексами.
- Progresses: колекція записів про прогрес.

Клас Progress

- Weight: вага для вправи.
- NumberOfRepetitions: кількість повторень.
- Date: дата виконання вправи.
- RateOfPerceivedExertion: оцінка суб'єктивної складності.
- UserId: ідентифікатор користувача.
- User: об'єкт користувача.
- ExerciseId: ідентифікатор вправи.
- Exercise: об'єкт вправи.

Клас User

- DisplayName: відображуване ім'я користувача.
- Bio: біографія користувача.
- Progresses: колекція записів про прогрес.

Клас Week

- StartDate: дата початку тижня.
- EndDate: дата закінчення тижня.
- Year: рік.
- WeekNumber: номер тижня.
- ComplexWeeks: колекція зв'язків з комплексами.

Ця діаграма класів, яку можна побачити на рисунку 10, демонструє, як різні сутності системи взаємодіють між собою, забезпечуючи організацію та управління фізичними навантаженнями спортсменів.

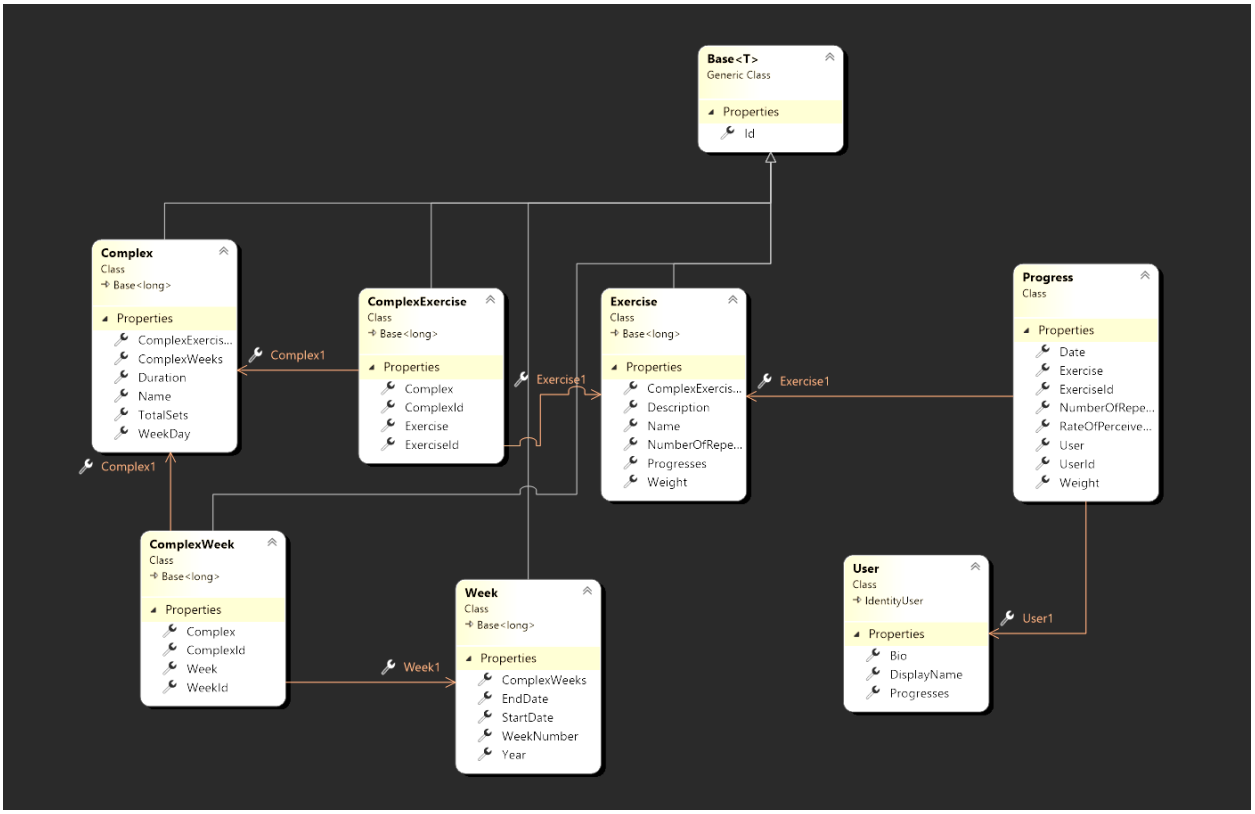

Рисунок 10 — Діаграма класів системи

3.5 Програмна реалізація застосунка

Додаток «Оптимізації фізичних навантажень спортсменів» складається з бекенд та клієнтської частини. Бекенд відповідає за обробку даних та логіку бізнесу, тоді як клієнтська частина реалізує інтерфейс користувача.

Для забезпечення оптимізації фізичних навантажень спортсменів, було розроблено і інтегровано модуль та алгоритм машинного навчання (Див. Рисунок 11).

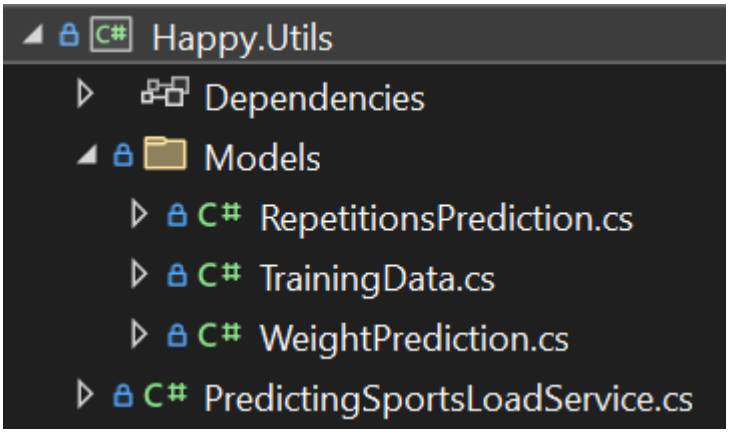

Рисунок 11 – Модуль для оптимізації фізичних навантажень

Для створення цього алгоритму було створено декілька моделей для передбачення майбутніх навантажень спортсмена.

Лістинг 3 Метод для передбачення ваги

```
public float PredictWeight(int numberOfRepetitions, float rateOfPerceivedExertion, float day)
        {
            var predictionEngine = _mlContext.Model.CreatePredictionEngine<TrainingData, WeightPrediction>(_weightModel);
            var prediction = predictionEngine.Predict(new TrainingData
            {
                NumberOfRepetitions = numberOfRepetitions,
                RateOfPerceivedExertion = rateOfPerceivedExertion,
                Day = day
            });

            return prediction.PredictedWeight;
        }
```

Лістинг 3 представляє метод PredictWeight, який призначений для передбачення ваги з використанням заданих параметрів. Цей метод отримує кількість повторень (numberOfRepetitions), рівень сприйнятого навантаження (rateOfPerceivedExertion) та день (day) як вхідні параметри.

У першому рядку методу створюється об'єкт predictionEngine за допомогою MLContext і моделі передбачення ваги (_weightModel). Наступний рядок виконує передбачення, передаючи нові дані (TrainingData) до predictionEngine. Результат передбачення зберігається у змінній prediction, після чого повертається передбачене значення ваги (prediction.PredictedWeight).

Лістинг 4 Метод для передбачення потрібної кількості вправ

```
      public float PredictRepetitions(float weight, float
rateOfPerceivedExertion, float day)
      {
          var                predictionEngine                =
_mlContext.Model.CreatePredictionEngine<TrainingData,
RepetitionsPrediction>(_repetitionsModel);
          var  prediction  =  predictionEngine.Predict(new
TrainingData
          {
              Weight = weight,
              RateOfPerceivedExertion                        =
rateOfPerceivedExertion,
              Day = day
          });

          return prediction.PredictedRepetitions;
      }
```

Лістинг 4 представляє метод PredictRepetitions, який призначений для передбачення кількості повторень в залежності від заданих параметрів. Цей

метод отримує вагу (weight), рівень сприйнятого навантаження (rateOfPerceivedExertion) та день (day) як вхідні параметри.

У першому рядку методу створюється об'єкт predictionEngine за допомогою MLContext і моделі передбачення кількості повторень (_repetitionsModel). Наступний рядок виконує передбачення, передаючи нові дані (TrainingData) до predictionEngine. Результат передбачення зберігається у змінній prediction, після чого повертається передбачене значення кількості повторень (prediction.PredictedRepetitions).

Лістинг 5 представляє сервіс для передбачення спортивного навантаження на основі навчальних даних, які надходять у вигляді списку об'єктів TrainingData.

Ініціалізація MLContext: У конструкторі створюється новий об'єкт MLContext, який використовується для взаємодії з бібліотекою ML.NET.

Завантаження даних: Дані про тренування завантажуються з переданого списку в dataView.

Створення моделей: Для передбачення ваги і кількості повторень створюються дві моделі: одна для передбачення ваги (_weightModel), інша для передбачення кількості повторень (_repetitionsModel).

Створення пайплайнів: Для кожної моделі створюється пайплайн, який складається з кількох етапів:

Concatenate: Об'єднання різних ознак у вектор ознак для використання в моделі.

Sdca: Вибір алгоритму для тренування моделі. У цьому випадку використовується стохастичний алгоритм оптимізації (Stochastic Dual Coordinate Ascent), який використовується для задач регресії.

Тренування моделей: Пайплайни об'єднуються з набором даних і викликається метод Fit(), що запускає процес тренування моделей на навчальних даних.

Отже, цей код ініціалізує сервіс для передбачення спортивного навантаження на основі введених даних тренувань і тренує дві моделі: одну для передбачення ваги та іншу для передбачення кількості повторень.

Лістинг 5 Сервіс для передбачення спортивного навантаження

```
        private readonly ITransformer _weightModel;
        private readonly ITransformer _repetitionsModel;

        public PredictingSportsLoadService(List<TrainingData>
trainingData)
        {
            _mlContext = new MLContext();

            var dataView =
_mlContext.Data.LoadFromEnumerable(trainingData);

            var weightPipeline =
_mlContext.Transforms.Concatenate("Features",
nameof(TrainingData.NumberOfRepetitions),
nameof(TrainingData.RateOfPerceivedExertion),
nameof(TrainingData.Day))

.Append(_mlContext.Regression.Trainers.Sdca(labelColumnName
: nameof(TrainingData.Weight), maximumNumberOfIterations:
100));
            _weightModel = weightPipeline.Fit(dataView);

            var repetitionsPipeline =
_mlContext.Transforms.Concatenate("Features",
nameof(TrainingData.Weight),
nameof(TrainingData.RateOfPerceivedExertion),
nameof(TrainingData.Day))
```

```
.Append(_mlContext.Regression.Trainers.Sdca(labelColumnName
: nameof(TrainingData.NumberOfRepetitions),
maximumNumberOfIterations: 100));
            _repetitionsModel =
repetitionsPipeline.Fit(dataView);
        }
```

Ще однією ключовою частиною програмного коду є отримання даних про модель на клієнтській стороні. Для цього створено спеціальний API-запит, на стороні бекенду.

Лістинг 6 відображає спеціальний API-запит, який створений на бекенді для отримання даних про модель на клієнтській стороні. Запит має HTTP метод GET та шлях, що містить параметри complexId, exerciseId та rate.

Опис маршруту: У контролері створений метод, який має маршрут [HttpGet("{complexId}/{exerciseId}/{rate}")], що вказує на доступ до методу через HTTP GET за шляхом, що містить параметри complexId, exerciseId та rate.

Обробка запиту: При отриманні запиту викликається метод GetCurrentUserAsync() для отримання інформації про поточного користувача. Після цього викликається метод PredictSportLoadAsync() сервісу _trainingRecommendationService, який передає ідентифікатор користувача, complexId, exerciseId та rate. Результат цього запиту зберігається у змінній response.

Повернення відповіді: Якщо запит успішний, метод повертає об'єкт response за допомогою методу Ok(). У разі виникнення помилки відправляється відповідь з кодом статусу BadRequest та повідомленням про помилку.

Цей код дозволяє клієнтській стороні отримати дані про модель, використовуючи спеціальний API запит на бекенді.

Лістинг 6 API запит для отримання оптимальних навантажень

```
        [HttpGet("{complexId}/{exerciseId}/{rate}")]
        public async Task<IActionResult> Get(long
complexId, long exerciseId, string rate)
        {
            try
            {
                var user = await GetCurrentUserAsync();
                var response = await
_trainingRecommendationService.PredictSportLoadAsync(user.I
d, complexId, exerciseId, rate);

                return Ok(response);
            }
            catch(Exception ex)
            {
                return BadRequest(ex.Message);
            }
        }
```

Що до клієнтьскої частини, Лістинг 7 демонструє функцію getRecommendation призначена для отримання рекомендацій з бекенду за допомогою API-запиту. Параметри complexId, exerciseId, та rate передаються у вигляді рядків. Перед тим, як виконати запит, вони перетворюються на числові значення за допомогою Number(complexId) та Number(exerciseId) для числового представлення.

Далі, виконується сам запит до бекенду за допомогою методу get() об'єкта agent.TrainingRecommendation. У цьому запиті передаються числові значення complexId, exerciseId, та rate. Результат запиту отримується у змінну recommendation.

У випадку успішного виконання запиту, результат зберігається у стані програми. Це здійснюється за допомогою функції runInAction(), яка забезпечує безпечність мутації стану. Отримана рекомендація зберігається у змінній this.recommendation.

У випадку помилки під час виконання запиту, помилка логується до консолі за допомогою console.log(). Також, за допомогою runInAction(), можливо виконати будь-які додаткові дії зі станом програми у випадку помилки.

Лістинг 7 Функція отримання рекомендації з бекенду

```
        getRecommendation = async(complexId:string,
exerciseId: string, rate: string) => {
            const numericId = Number(complexId);
            const numericExerciseId = Number(exerciseId);
            try{
            const recommendation = await
agent.TrainingRecommendation.get(numericId,
numericExerciseId, rate);
            runInAction(() => {
                this.recommendation = recommendation;
            })
            }catch(error){
                runInAction(() => {
                    console.log(error);
                })
            }
        }
```

За допомогою форми, яку користувач заповняє, при отриманні рекомендації, він має можливість отримати дані відносно його результату за минулі тренування.

3.6 Розробка інтерфейсу комп'ютерної системи оптимізації фізичних навантажень спортсменів

Проект інтерфейсу для оптимізації фізичних навантажень спортсменів розроблений з метою забезпечення зручного та ефективного інструменту для аналізу та управління тренувальними режимами. Інтерфейс відзначається простотою та інтуїтивністю використання, а також надає повний спектр функцій для визначення оптимальних навантажень, необхідних для досягнення спортивних цілей.

Як тільки користувач входе в свій акаунт він попадає на головний екран додатку(Див. Рисунок 12), на якому натиснувши кнопку «Go to complexes» зможе перейти до перегляду комлпексів.

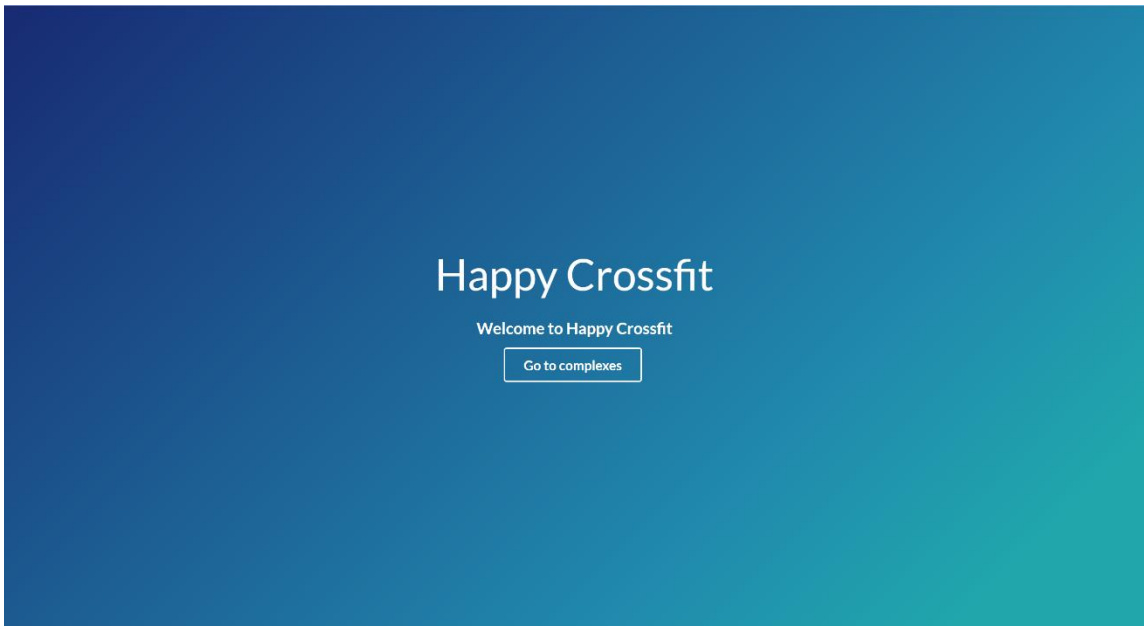

Рисунок 12 — Початковий екран

Перейшовши на екран «Complexes» (Див. Рис користувач має можливість продивитись всі комплекси на певному тижні. Натиснувши на кнопку «Next Week», користувач отримує інформацію про комплекси на другому тижні. Спортсмен має можливість продивитись детальну інформацію про кожен комплекс, натиснувши кнопку «View».

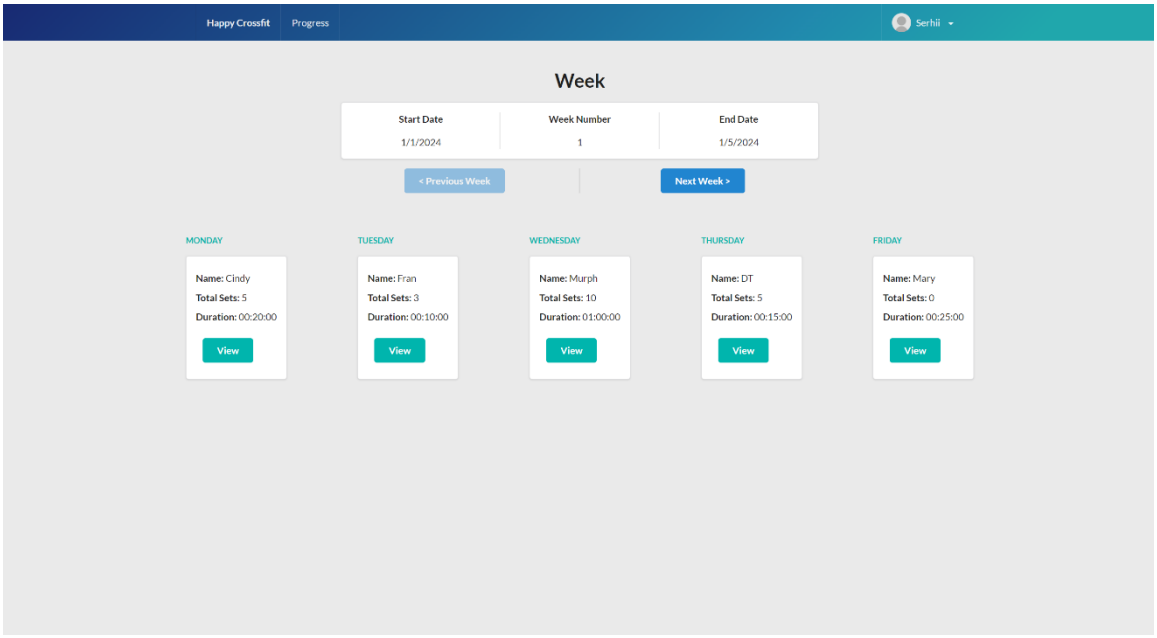

Рисунок 13 — Екран з комплексами

Перейшовши на екран «Exercise»(Див. Рисунок 14), користувач має можливість продивитись детальну інформацію про комплекс, а саме:

1. Кількість повторень, яке потрібно зробити.
2. Час виконання.
3. Список вправ

Біля кожної вправи є можливість записати кількість повторень та вагу, яку було зроблено (Див. Рисунок 15) та подивитись рекомендації з приводу поточного навантаження (Див. Рисунок 16).

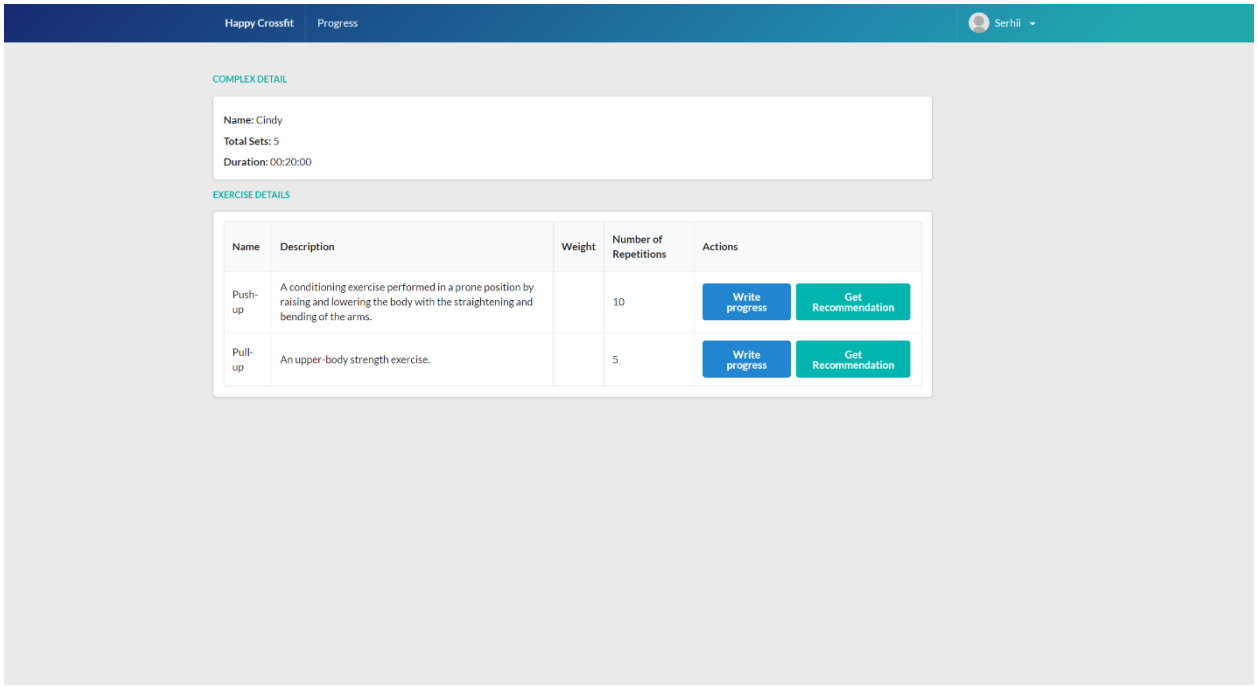

Рисунок 14 — Екран «Детальна інформація про комплекси»

Натиснувши на кнопку «Write progress», користувач може вписати дані та зберегти їх, які згодом зможе побачити на графіку на сторінці «Progress»(Див. Рисунок 17).

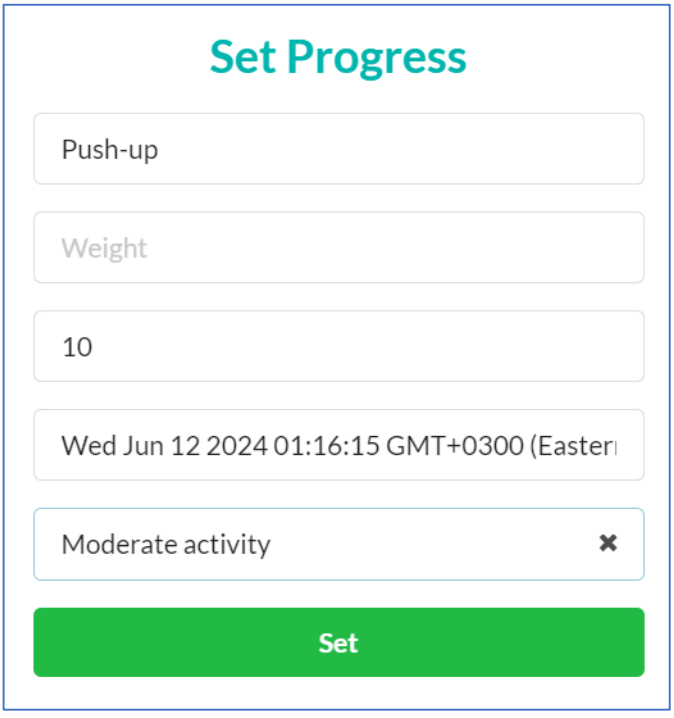

Рисунок 15 — Форма запису прогресу

Натиснувши на кнопку «Get Recommendation», користувач має можливість ввести свій поточний стан та отримати рекомендації з приводу поточного навантаження.

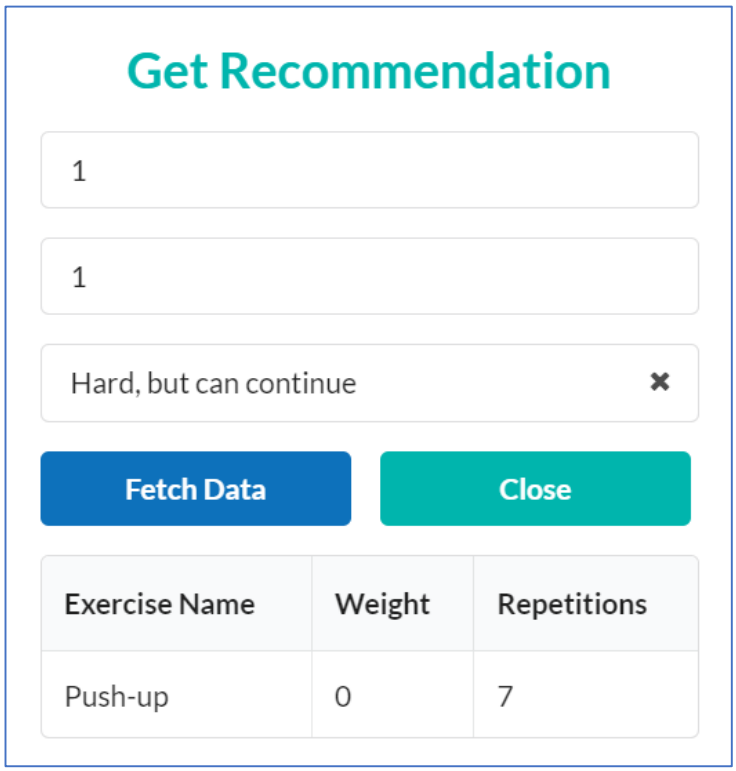

Рисунок 16 — Форма отримання рекомендацій про навантаження

Перейшовши на сторінку «Progress»(Див. Рисунок 17), користувач має можливість уважно продивитись про досягнення в кожній вправі та побачити графік його самопочуття на тренуваннях.

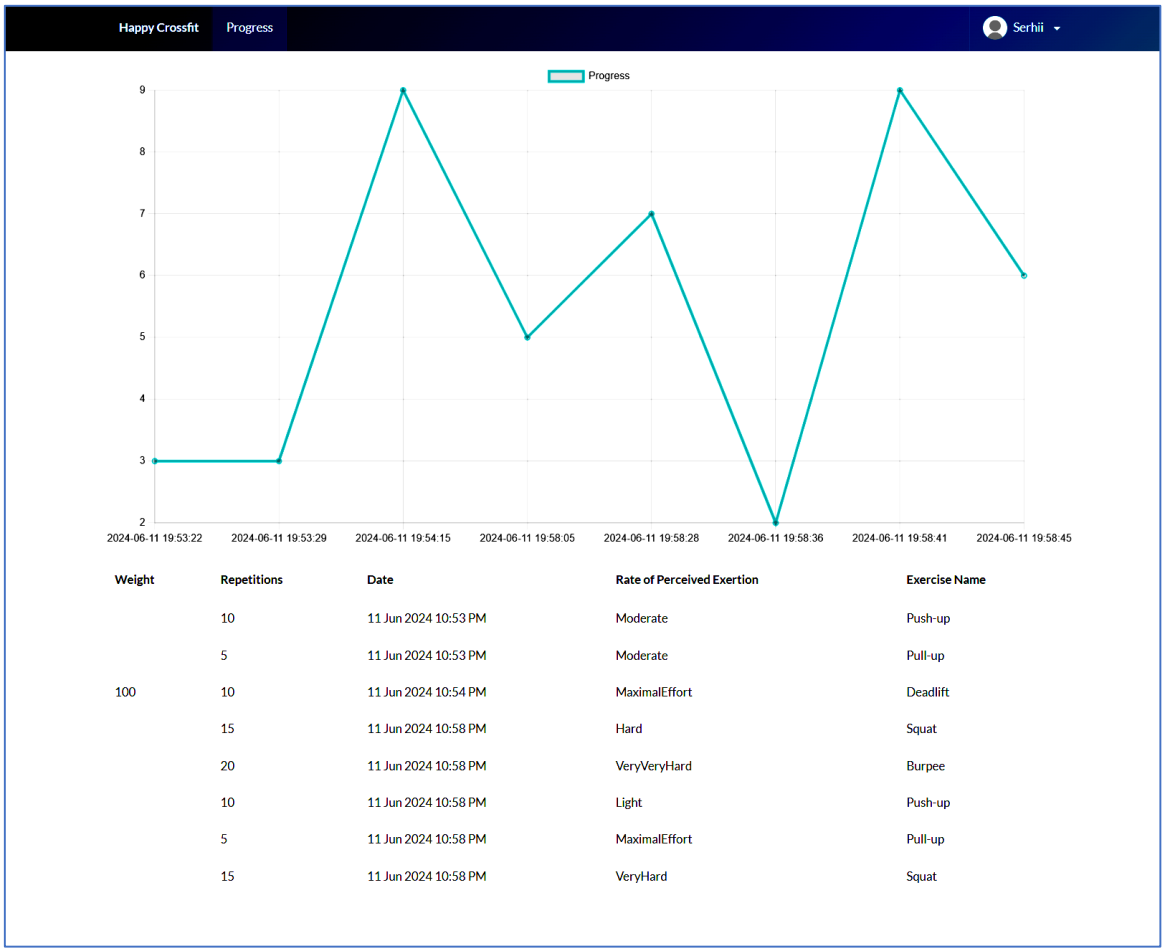

Рисунок 17 — Екран Progress

Вийшовши з акаунту звичайного користувача, переходимо на екран аутентифікації (Див. Рисунок 18), за допомогою якого можна увійти, або зареєструватися в додатку. Увійшовши під акаунтом адміністратора, буде доступний додатковий функціонал додатку (Див. Рисунок 19-21).

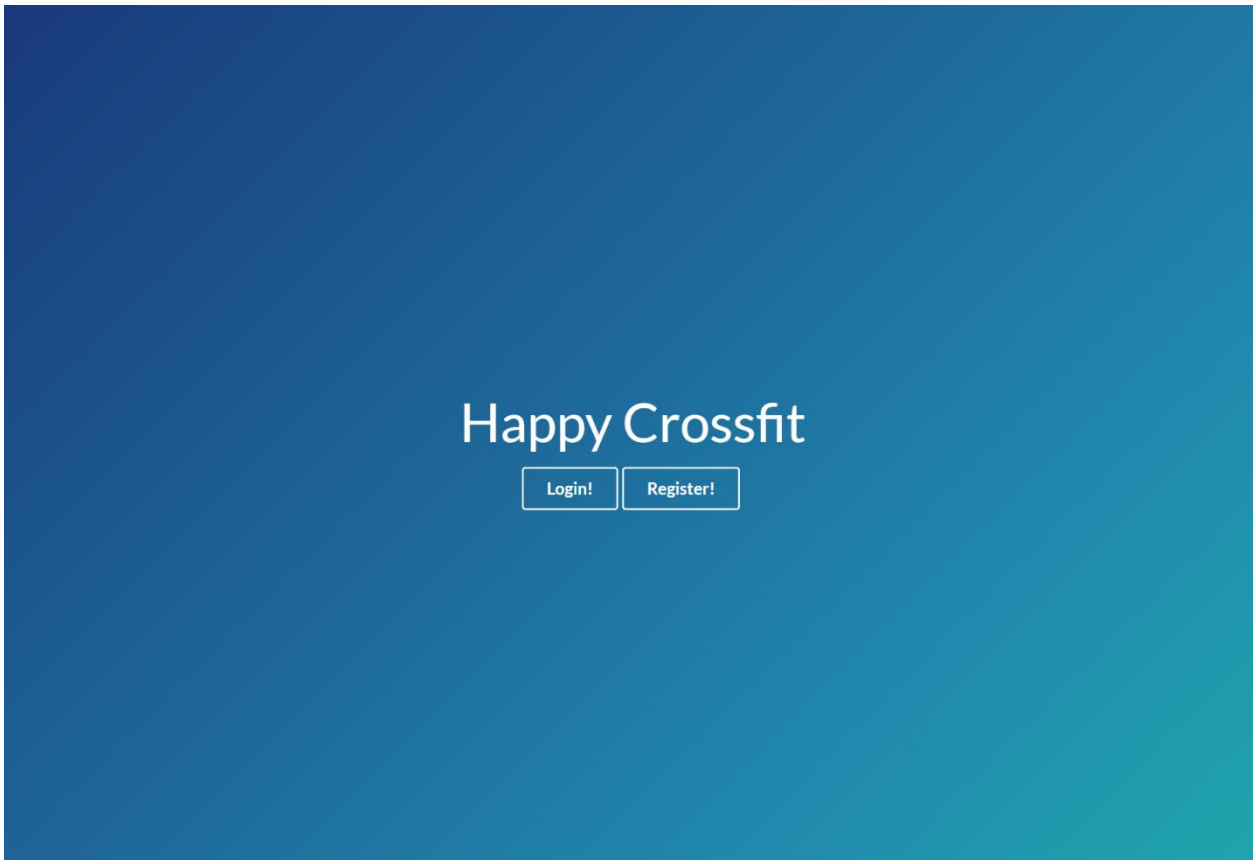

Рисунок 18 — Екран аутентифікації

На екрані «Complexes», тренер має можливість створити комплекс, або видалити його.

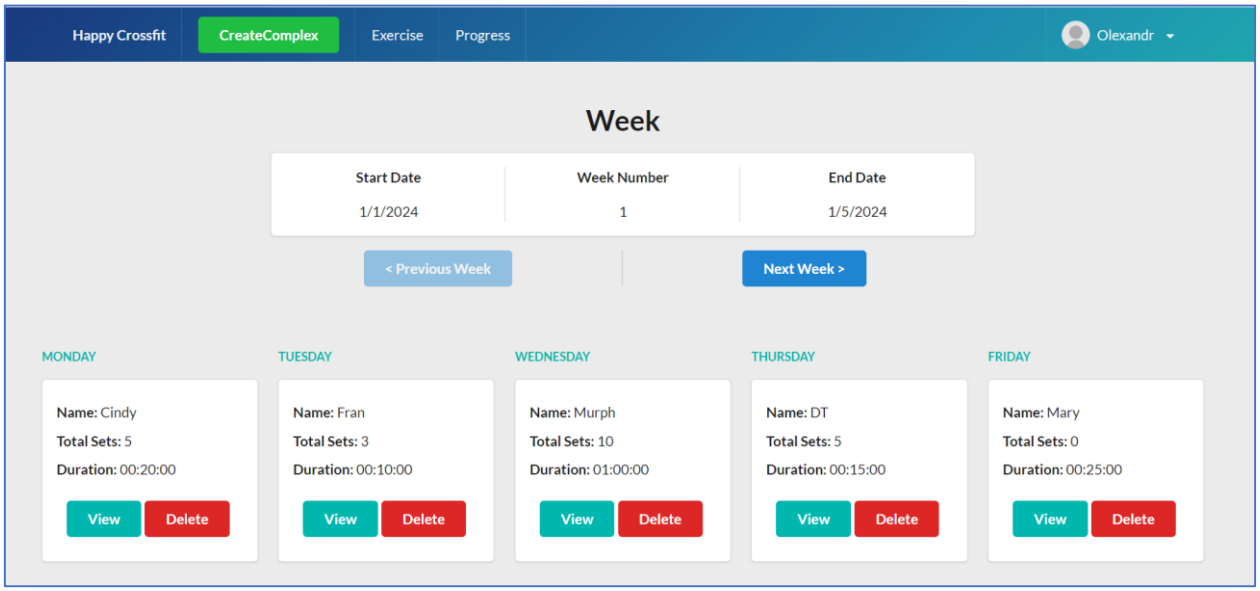

Рисунок 19 — Екран Complexes

Натиснувши кнопку «CreateComplex», адміністратор може заповнити поля та створити новий комплекс, який буде відображено на екрані «Complexes».

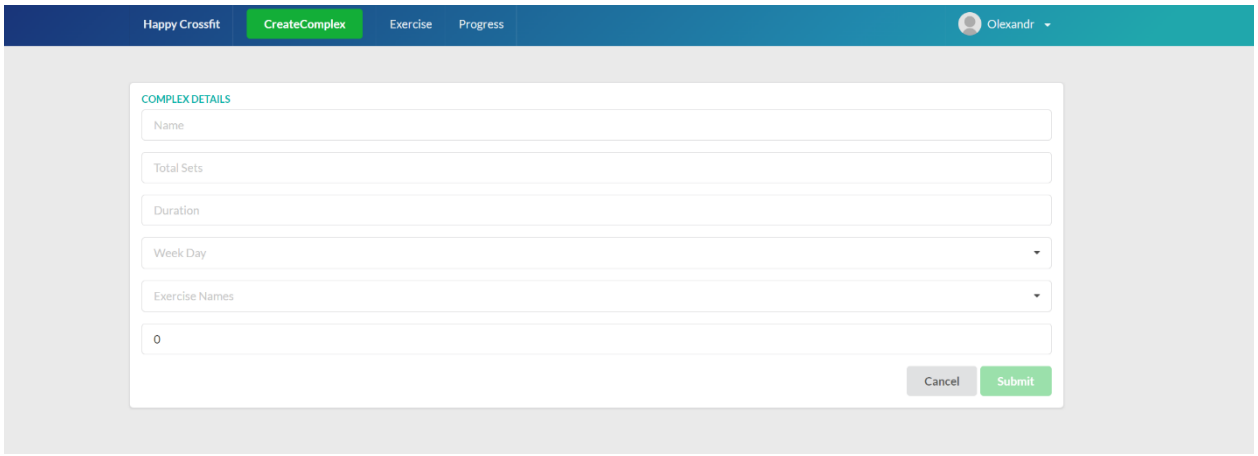

Рисунок 20 — Створення комплексу

Перейшовши на екран «Progress», тренер має можливість створити вправу, редагувати, або видалити.

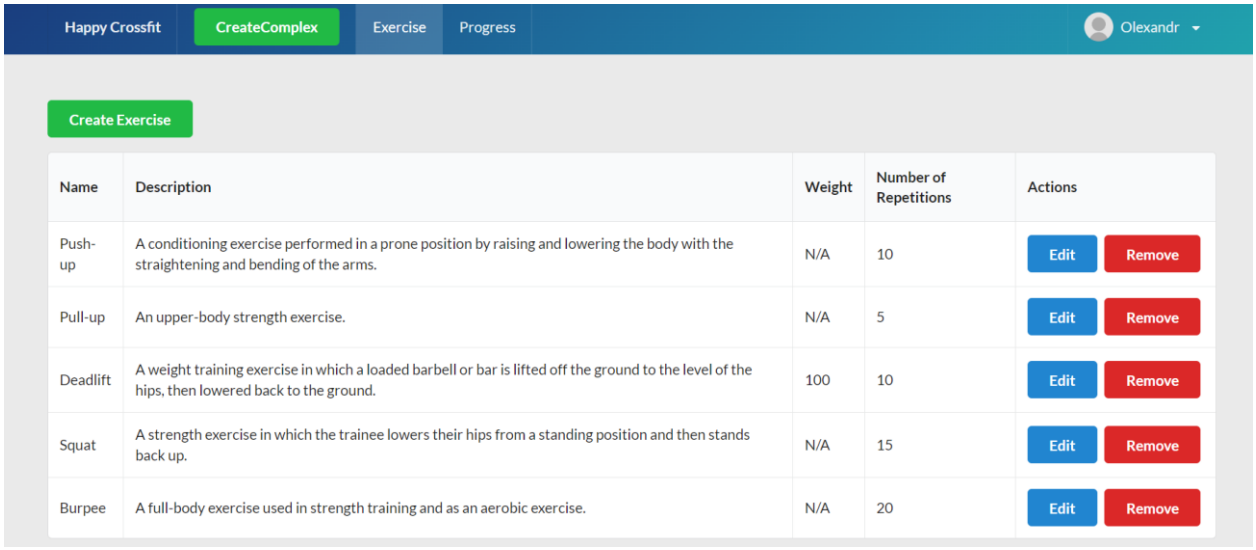

Рисунок 21 — Екран Exercise

3.7 Висновки з 3 розділу

У цьому розділі було здійснено комплексний підхід до розробки комп'ютерної системи оптимізації фізичних навантажень спортсменів. На кожному етапі розробки проведено детальний аналіз та проектування для забезпечення ефективності та надійності системи.

Спершу, проведено ґрунтовний аналіз предметної області. Визначено основні потреби та виклики, що стоять перед спортсменами та їхніми тренерами в контексті оптимізації фізичних навантажень. Це дозволило сформувати базу для подальшого проектування системи.

Далі, розроблено архітектуру системи. Було запропоновано багаторівневу структуру, що включає базу даних, серверну частину та клієнтський додаток. Такий підхід забезпечує гнучкість, масштабованість та злагоджену роботу всіх компонентів системи.

Визначено функціональні вимоги системи. На основі аналізу предметної області сформульовано ключові функції, які має виконувати система,

включаючи моніторинг фізичних показників, аналіз даних, генерацію рекомендацій та налаштування індивідуальних програм тренувань.

Розглянуто основні компоненти, їхні функції та взаємодію між собою. Це забезпечило цілісність та злагоджену роботу системи.

Описано програмну реалізацію системи. Використано сучасні технології та мови програмування для створення надійного та ефективного програмного забезпечення. Наведено приклади коду та пояснення до основних алгоритмів, що дозволило продемонструвати реалізацію функцій системи.

Загалом, проведено всебічний аналіз та розробку комп'ютерної системи оптимізації фізичних навантажень спортсменів. Від опису предметної області до програмної реалізації, кожен етап сприяв створенню ефективної та надійної системи, яка відповідає сучасним вимогам та потребам користувачів.

ВИСНОВКИ

1. Під час виконання дипломної роботи проведений комплексний аналіз предметної області, що дозволило точно сформулювати основну задачу роботи та узгодити її з науковим керівником. Проведений детальний аналіз існуючих методів рішення, а також вивчено та зібрано матеріали, що стосуються теми дипломної роботи. Оцінені наявні рішення та аналоги, що дозволило зрозуміти їх сильні та слабкі сторони.
2. Проведений аналіз та оцінку сучасних технологій для створення клієнтської частини веб-застосунків, що забезпечило вибір оптимальних інструментів для реалізації проекту. Окрему увагу було приділено аналізу PostgreSQL СУБД, що допомогло визначити її переваги для використання в проекті.
3. Проведений огляд алгоритму машинного навчання для прогнозування майбутніх навантажень, що стало основою для розробки системи оптимізації фізичних навантажень спортсменів. На основі цього було розроблено та уточнено функціональні вимоги до системи, що забезпечило чітке розуміння необхідних функцій та можливостей.
4. Розробка архітектури системи та її проектування були виконані з урахуванням всіх зібраних даних та проведеного аналізу, що дозволило створити ефективну та функціональну систему. В подальшому було розроблено програмне забезпечення.
5. Кінцевим результатом роботи над проєктом стало створення та впровадження комп’ютерної системи для оптимізації фізичних навантажень спортсменів. Усі заплановані функціональні вимоги були виконані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Підвищення ефективності індивідуального тренування в спорті за допомогою штучного інтелекту: комплексний огляд. URL: https://oliverbodemer.medium.com/enhancing-individual-sports-training-through-artificial-intelligence-a-comprehensive-review-786725d57ef3 (дата звернення 01.06. 2024 р.).
2. Технологічні досягнення в спорті. URL: https://cactusware.com/blog/sports-technology#:~=Ultimately%2C%20technological%20advancements%20in%20sports,line%20technology%2C%20and%20virtual%20reality. (дата звернення 01.06. 2024 р.).
3. Машинне навчання. Що це? URL: (дата звернення 01.06. 2024 р.).
4. Що таке ML .Net? URL: https://learn.microsoft.com/ru-ru/dotnet/machine-learning/how-does-mldotnet-work (дата звернення 01.06. 2024 р.).
5. PostgreSql Documentation. URL: https://www.postgresql.org/docs/ (дата звернення 01.06. 2024 р.).
6. Entity Framework Core. URL: https://learn.microsoft.com/en-us/ef/core/ (дата звернення 01.06. 2024 р.).
7. React Documentation. URL: https://ru.legacy.reactjs.org/docs/getting-started.html (дата звернення 01.06. 2024 р.).
8. Mobx-React. URL: https://mobx.js.org/react-integration.html (дата звернення 01.06. 2024 р.).
9. Semantic Ui React Documentation. URL: https://react.semantic-ui.com/ (дата звернення 01.06. 2024 р.).
10. How to Use Axios with React. URL: https://www.digitalocean.com/community/tutorials/react-axios-react (дата звернення 01.06. 2024 р.).
11. .Net Documentation. URL: https://dotnet.microsoft.com/ru-ru/ (дата звернення 01.06. 2024 р.).

12. Three Layered Architecture. URL: https://medium.com/@deanrubin/the-three-layered-architecture-fe30cb0e4a6 (дата звернення 01.06. 2024 р.).
13. Three Layered Architecture Implementation example. URL: https://medium.com/@asoldan1459/three-layered-architecture-with-example-b597a2161538 (дата звернення 01.06. 2024 р.).
14. AutoMapper Documentation. URL: https://docs.automapper.org/en/stable/Getting-started.html (дата звернення 01.06. 2024 р.).
15. Using repository and specificartion patterns in .Net. URL: https://medium.com/@rudyzio92/net-core-using-the-specification-pattern-alongside-a-generic-repository-318cd4eea4aa (дата звернення 01.06. 2024 р.).

Декларація
академічної доброчесності
здобувача ступеня вищої освіти ЗНУ

Я, Швець Олександр Сергійович, студент 4 курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти ipz20bd-216@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Комп'ютерна система оптимізації фізичних навантажень спортсменів**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

згоден/згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 17.06.2024 Підпис _____ Швець Олександр Сергійович
(студент)

Дата 18.06.2024 Підпис _____ Безверхий Анатолій Ігорович
(науковий керівник)