Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для вебсистеми велоклубу*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент групи КН-20-2* _____ *Максим ГОЛЯЖ*
Група виконавця | Підпис | Ім'я, ПРІЗВИЩЕ

Керівник: *к.т.н., доц. каф. КН* _____ *Олександр ПАСІЧНИК*
Науковий ступінь, посада | Підпис | Ім'я, ПРІЗВИЩЕ

Нормоконтроль: *к.т.н., доц. каф. КН* _____ *Руслан БАГРІЙ*
Науковий ступінь, посада | Підпис | Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
зав. кафедри КН, д.т.н., професор _____ *Олександр БАРМАК*
Підпис | Ім'я, ПРІЗВИЩЕ
21 06 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*

(підпис)
д.т.н., професор Олександр БАРМАК
« 16 » лютого 2024 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб-системи велоклубу»*

2. Завдання видано студенту *Максиму ГОЛЯЖУ*
(Ім'я, прізвище)

3. Керівник роботи *доцент кафедри КН Олександр ПАСІЧНИК*
(посада, ім'я, прізвище)

4. Затверджено наказом університету від «15» 02 2024 р. № 8

5. Дата видачі завдання студенту: «16» 02 2024 р.

6. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Метою кваліфікаційної роботи бакалавра є покращення зручності планування розвитку фізичних навичок учасників велоклубу за рахунок автоматизації процесу формування плану тренувань та врахування індивідуальних особливостей. Для досягнення поставленої мети визначені такі задачі дослідження: провести аналіз методів формування та покращення розвитку фізичних навиків учасників велоклубу; провести аналіз інтелектуальних методів для визначення індивідуальних планів тренувань для учасників велоклубу; реалізувати метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб системи велоклуб; виконати програмну реалізацію методу та провести її експериментальне тестування.

7. Календарний план виконання кваліфікаційної роботи бакалавра:

№	Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Примітка
1	*Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником, складання календарного графіка виконання роботи*	*січень 2024*	*виконано*
2	*Ознайомлення з характеристокою предметної області та постановка задачі*	*лютий 2024*	*виконано*
3	*Проектування та розробка методу плану розвидку фізичних навичок учасників за допомогою генетичного алгоритму для веб системи велоклуб*	*березень 2024*	*виконано*
4	*Програмна реалізація методу плану розвидку фізичних навичок учасників за допомогою генетичного алгоритму для веб системи велоклуб*	*квітень 2024*	*виконано*
5	*Написання пояснювальної записки, урахування зауважень керівника, оформлення згідно вимог*	*травень 2024*	*виконано*
6	*Розробка презентаційних матеріалів та попередній захист кваліфікаційної роботи*	*травень 2024*	*виконано*
7	*Отримання відгуку керівника, рецензії, перевірка на плагіат, нормоконтроль*	*червень 2024*	*виконано*
8	*Підготовка до захисту та захист кваліфікаційної роботи бакалавра*	*червень 2024*	*виконано*

Виконавець: *студент групи КН-20-2* _____ *Максим ГОЛЯЖ*
Група виконавця — Підпис — Ім'я, ПРІЗВИЩЕ

Керівник: *к.т.н., доц. каф. КН* _____ *Олександр ПАСІЧНИК*
Науковий ступінь, посада — Підпис — Ім'я, ПРІЗВИЩЕ

Анотація

Тема кваліфікаційної роботи бакалавра: *«Метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб-системи велоклубу»*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-20-2 Максим ГОЛЯЖ.*

Керівник кваліфікаційної роботи бакалавра: *к.т.н., доцент кафедри КН Олександр ПАСІЧНИК.*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
80	32	5	49	4

Метою кваліфікаційної роботи бакалавра є покращення зручності планування розвитку фізичних навичок учасників велоклубу за рахунок автоматизації процесу формування плану тренувань та врахування індивідуальних особливостей. Для досягнення поставленої мети визначені такі задачі дослідження: провести аналіз методів формування та покращення розвитку фізичних навиків учасників велоклубу; провести аналіз інтелектуальних методів для визначення індивідуальних планів тренувань для учасників велоклубу; реалізувати метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб системи велоклуб; виконати програмну реалізацію методу та провести її експериментальне тестування; виконати дослідження покращення планування розвитку фізичних навичок учасників велоклубу за рахунок автоматизації процесу формування плану тренувань та врахування індивідуальних особливостей.

Виконавець: *студент групи КН-20-2* _____ *Максим ГОЛЯЖ*

Група виконавця | Підпис | Ім'я, ПРІЗВИЩЕ

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
СКБД	Система керування базами даних
SQL	Structured query language –мова структурованих запитів
T-SQL	Transact-SQL – процедурне розширення мови SQL
ШІ	Штучний інтелект
ГА	Генетичний алгоритм

Вступ

Кваліфікаційна робота ставить перед собою мету покращити зручності планування розвитку фізичних навичок учасників велоклубу за рахунок автоматизації процесу формування плану тренувань та врахування індивідуальних особливостей.

Актуальність роботи.

У сучасному світі, де технології невпинно розвиваються, комп'ютерні науки стали невід'ємною частиною нашого повсякденного життя. Їхній вплив поширюється на різноманітні сфери, від комунікацій та освіти до охорони здоров'я та спорту. Завдяки інноваційним рішенням у галузі комп'ютерних наук, ми маємо можливість оптимізувати процеси, автоматизувати рутинні завдання та персоналізувати досвід користувачів у багатьох аспектах нашого життя.

Однією з областей, де комп'ютерні науки демонструють свій значний потенціал, є спорт. Сьогодні спортсмени та тренери використовують передові технології для аналізу даних, моніторингу продуктивності та вдосконалення тренувальних програм. Наприклад ШІ та машинне навчання допомагають аналізувати великі обсяги даних про фізичний стан спортсменів, їхню техніку та тактику, щоб виявити приховані закономірності та оптимізувати стратегії тренувань.

Одним з найперспективніших напрямків у комп'ютерних науках є застосування генетичних алгоритмів - методу оптимізації, натхненного процесами природного відбору та генетики. Генетичні алгоритми особливо ефективні при вирішенні складних проблем з багатьма змінними, де традиційні методи можуть бути неефективними. У спорті ці алгоритми можуть використовуватися для оптимізації тренувальних планів, враховуючи індивідуальні особливості кожного спортсмена: генетичні схильності, історію травм, рівень підготовки, цілі та навіть психологічні аспекти.

Велоспорт - це вид спорту, який особливо виграє від інтеграції комп'ютерних наук. Участь у велосипедних змаганнях вимагає високого рівня

витривалості, сили та тактичного мислення. Крім того, велоспорт характеризується різноманітністю дисциплін (шосе, трек, крос-кантрі тощо) та різними типами трас, кожна з яких вимагає специфічного набору навичок. Тому розробка оптимального плану тренувань для велосипедиста - це складне завдання, яке вимагає врахування багатьох факторів.

Саме тут на допомогу приходять велоклуби - спільноти ентузіастів велоспорту, які об'єднують професіоналів та аматорів. Велоклуби не лише організовують групові поїздки та змагання, але й надають своїм учасникам цінні ресурси для вдосконалення навичок: консультації тренерів, майстер-класи, а також індивідуальні плани тренувань.

Веб-система велоклубу, яка використовує генетичний алгоритм для планування розвитку фізичних навичок учасників, стає потужним інструментом. Така система може автоматично генерувати оптимальні тренувальні плани, адаптовані до індивідуальних характеристик кожного велосипедиста.

Розробка такої системи не лише вирішує практичну проблему оптимізації тренувальних планів для велосипедистів, але й демонструє потужний потенціал міждисциплінарного підходу. Автоматизуючи та персоналізуючи процес планування тренувань, ця система робить професійні методики доступними для ширшої аудиторії, сприяючи здоровому та активному способу життя.

Тема кваліфікаційної роботи бакалавра – метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб системи велоклубу.

Об’єкт дослідження – процес планування розвитку фізичних навичок учасників велоклубу за допомогою генетичного алгоритму для веб системи велоклубу.

Предмет дослідження – методи оптимізації за критерієм вагомості спортивних досягнень, технології створення веб систем.

Мета кваліфікаційної роботи бакалавра – покращення планування розвитку фізичних навичок учасників велоклубу за рахунок автоматизації процесу формування плану тренувань та врахування індивідуальних

особливостей.

Завдання кваліфікаційної роботи бакалавра:

Для досягнення поставленої мети визначені такі задачі дослідження:

– провести аналіз методів формування та покращення розвитку фізичних навиків учасників велоклубу;

– провести аналіз інтелектуальних методів для визначення індивідуальних планів тренувань для учасників велоклубу;

– реалізувати метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб системи велоклуб;

– виконати програмну реалізацію методу планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб системи велоклуб;

– провести експериментальне тестування програмної реалізації методу;

– виконати дослідження покращення планування розвитку фізичних навичок учасників велоклубу за рахунок автоматизації процесу формування плану тренувань та врахування індивідуальних особливостей.

Структура та обсяг кваліфікаційної роботи бакалавра.

Кваліфікаційна робота бакалавра складається з завдання, змісту, переліку скорочень, вступу, 3 розділів, висновків, переліку посилань із 49 кількостями джерел та 4 додатків. Загальний обсяг дипломної роботи бакалавра становить 123 сторінок, з них 80 сторінок основного тексту, 32 рисунків та 5 таблиць.

Ключові слова: метод, генетичний алгоритм, база даних, розвиток навичків, покращення навичків, програмна реалізація.

Розділ 1 Характеристика предметної області та постановка задачі

1.1 Огляд теоретичних підходів

Сучасні комп'ютерні науки надають потужні методи та алгоритми вирішення різноманітних завдань у різних сферах людської діяльності [1]. Одним із прикладів є оптимізаційні задачі, для розв'язання яких успішно застосовуються евристичні алгоритми, засновані на біологічних принципах еволюції та природного добору [2]. Спортивна галузь не є винятком - застосування новітніх комп'ютерних методів та систем здатне значно підвищити ефективність тренувальних процесів та розвитку фізичних навичок спортсменів [3].

У галузі комп'ютерних наук існує широкий спектр теоретичних підходів та алгоритмічних методів для вирішення оптимізаційних задач різної складності та специфіки [4]. Одними з найбільш потужних є евристичні методи, засновані на біологічних принципах еволюції та природного добору [5]. Вони дозволяють знаходити наближені до оптимального рішення для NP-складних проблем у випадках, коли точні методи виявляються непрактичними або надто ресурсо затратними.

Одним з провідних евристичних підходів є генетичні алгоритми [6], що моделюють процес природної еволюції для пошуку оптимальних рішень. Вони оперують "популяцією" можливих варіантів рішення задачі, які піддаються "мутаціям" та "схрещуванню" згідно певних правил відбору [7]. Найбільш пристосовані "особини" (рішення) виживають та дають початок новому поколінню варіантів, наближаючись до оптимального результату [8].

Генетичні алгоритми виявилися ефективними для розв'язання широкого кола оптимізаційних задач у різноманітних галузях [9]:

– у машинобудуванні для оптимізації конструкцій та вибору параметрів систем;

– в економіці та фінансах для складання ефективних інвестиційних стратегій;

– у біоінформатиці для аналізу й обробки генетичних послідовностей;

– у сфері штучного інтелекту для тренування та налаштування нейронних мереж.

І це лише деякі приклади застосування генетичних алгоритмів, що підтверджують їх універсальність та потужність як інструменту пошуку оптимальних рішень.

Використання генетичного алгоритму дозволяє ефективно опрацьовувати величезний обсяг можливих комбінацій вправ, навантажень та режимів для пошуку оптимального варіанту, максимально персоналізованого до індивідуальних потреб кожного учасника клубу [10].

Успішні приклади впровадження генетичних алгоритмів спостерігаються в різних спортивних дисциплінах - від професійної підготовки атлетів для Олімпійських ігор до розробки оптимальних стратегій командних видів спорту [11]. Отже, застосування цього підходу для оптимізації тренувань учасників велоклубу має значний потенціал для покращення ефективності фізичної підготовки.

Використання методів штучного інтелекту, машинного навчання та експертних систем дозволяє автоматизувати та оптимізувати процес планування розвитку фізичних навичок членів клубу [12]. Такі системи можуть генерувати персоналізовані тренувальні програми високої якості з урахуванням індивідуальних параметрів кожного спортсмена [13].

Поряд з плануванням тренувань, комп'ютерні технології покращують інші аспекти діяльності спортивних клубів. Так, веб-системи забезпечують зручний доступ до інформації про клуб, тренувальні заходи, досягнення тощо [14]. Вони також надають можливість дистанційного консультування з тренерами, обміну досвідом між учасниками. Інтерактивні онлайн-інструменти допомагають відстежувати власні показники, ставити цілі та мотивувати себе до їх досягнення [15].

Важливим застосуванням комп'ютерних технологій є автоматизація організаційних процесів у спортивних клубах. Спеціалізовані програми

дозволяють керувати розкладом тренувань, реєстрацією учасників, формуванням команд для змагань тощо [16]. Це значно спрощує адміністративну роботу та підвищує ефективність управління клубом.

Використання генетичного алгоритму має значну актуальність [17]. Ця технологія може допомогти в організації тренувань та розвитку фізичних здібностей учасників, аналізувати їхні досягнення й сприяти тренерам у складанні індивідуальних програм для кожного спортсмена. Крім того, штучний інтелект може забезпечити автоматизацію адміністративних процесів у клубі, що дозволить зосередитися на більш важливих аспектах діяльності.

1.2 Аналіз предметної області

Зазвичай планування діяльності велоклубу, включно з розробкою планів тренувань для учасників, здійснюється вручну тренерами або досвідченими учасниками [18]. Цей процес часто базується на їх власному досвіді, інтуїції та загальних рекомендаціях щодо тренувальних програм.

Типовий процес планування включає наступні етапи [19]:

– збір інформації про учасників: вік, стать, рівень підготовки, цілі, обмеження тощо. Ця інформація зазвичай збирається за допомогою анкет або особистих інтерв'ю [20];

– аналіз зібраної інформації та визначення загальних тренувальних потреб групи учасників;

– розробка базових планів тренувань для різних рівнів підготовки (початківці, середній рівень, просунутий рівень) з урахуванням загальних рекомендацій та досвіду тренера;

– індивідуальна адаптація базових планів для кожного учасника з урахуванням його конкретних цілей, обмежень та поточного рівня підготовки;

– періодичний перегляд та коригування планів тренувань на основі зворотного зв'язку від учасників та спостережень за їх прогресом.

Цей процес покладається на суб'єктивний досвід та судження тренерів

або досвідчених учасників [21]. Він може бути трудомістким та схильним до помилок, особливо коли кількість учасників є великою, а їх індивідуальні потреби різноманітні [22].

Застосування генетичного алгоритму для автоматизації процесу планування діяльності велоклубу та формування індивідуальних планів тренувань для учасників є актуальним з кількох причин [23]:

– об'єктивність та персоналізація - генетичний алгоритм може враховувати багато індивідуальних факторів для кожного учасника (вік, стать, рівень підготовки, цілі, обмеження тощо) та генерувати оптимальні плани тренувань без суб'єктивності людського фактору;

– ефективність та продуктивність. Автоматизований процес формування планів за допомогою генетичного алгоритму є набагато швидшим та менш трудомістким, ніж ручний процес, особливо для великої кількості учасників;

– адаптивність та гнучкість. Генетичний алгоритм може динамічно адаптувати плани тренувань до змін у потребах або обмеженнях учасників, забезпечуючи постійну актуальність планів [24];

– аналіз та оптимізація. Алгоритм може аналізувати дані про прогрес учасників та постійно вдосконалювати процес формування планів для досягнення кращих результатів.

Таким чином, використання генетичного алгоритму в процесі планування діяльності велоклубу [25] може значно підвищити ефективність, персоналізацію та об'єктивність формування планів тренувань для учасників, забезпечуючи кращі результати у розвитку їх фізичних навичок [26].

1.3 Аналіз інформаційного забезпечення предметної області

На сьогоднішній день ринок програмного забезпечення для спортивної індустрії пропонує досить широкий вибір різноманітних інструментів та додатків [27]. Проте, рішення для автоматизованого та персоналізованого планування фізичної підготовки на базі передових алгоритмічних методів, таких

як генетичні алгоритми, залишаються рідкісним явищем. Більшість наявних програм орієнтовані на стандартні функції фітнес-трекерів, відстеження активності чи базового складання тренувальних планів без глибокої персоналізації [28].

Розглянемо кілька популярних програмних продуктів, які певним чином стосуються проблематики планування розвитку фізичних навичок учасників спортивних клубів.

Strava [29] - одна з найпопулярніших мобільних та веб-платформ для відстеження фізичної активності (рисунок 1.1).

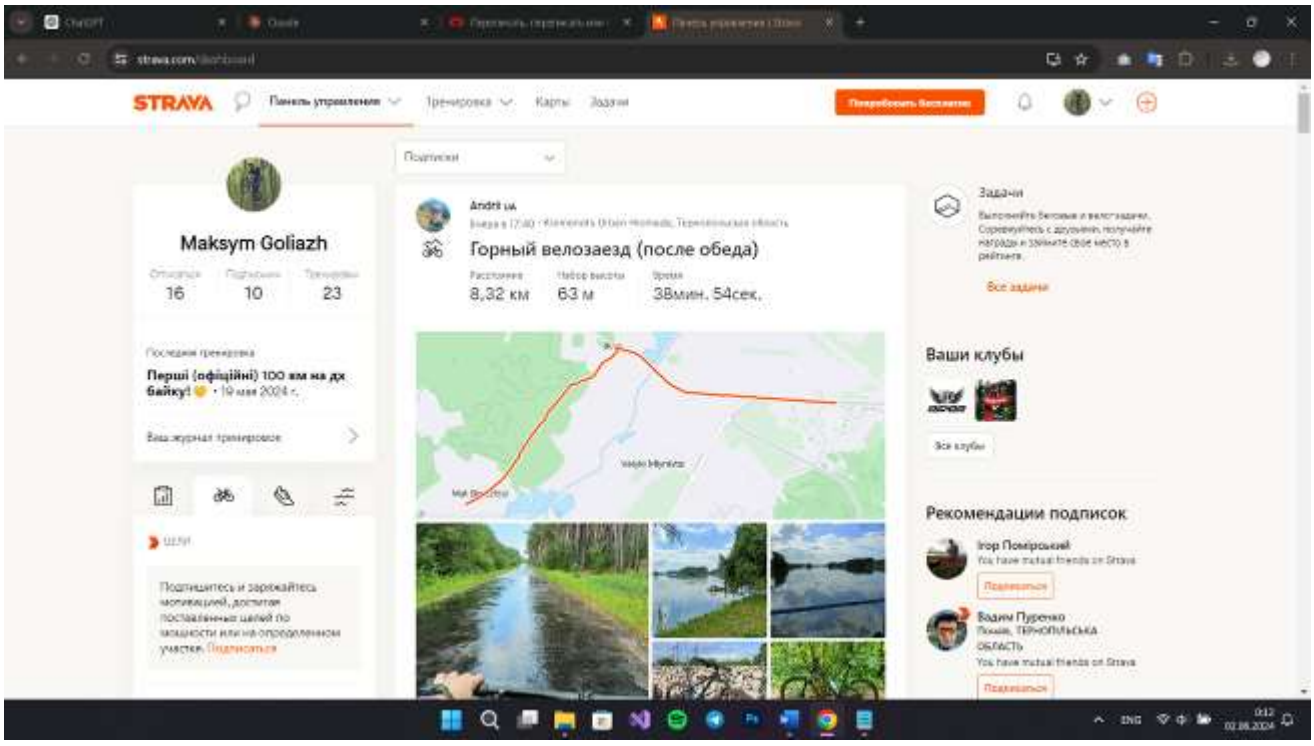


Рисунок 1.1 – Головне меню Strava [30]

Вона дозволяє записувати та аналізувати (рисунок 1.2) деталі велосипедних поїздок, забігів, піших прогулянок тощо [31].

Можна оглянути на карті загальний записаний маршрут. Що для сучасної веб системи є досить зручно та корисно. Кожен учасник має змогу це зробити у персональному кабінеті. Також записаний маршрут можуть переглядати інші учасники.

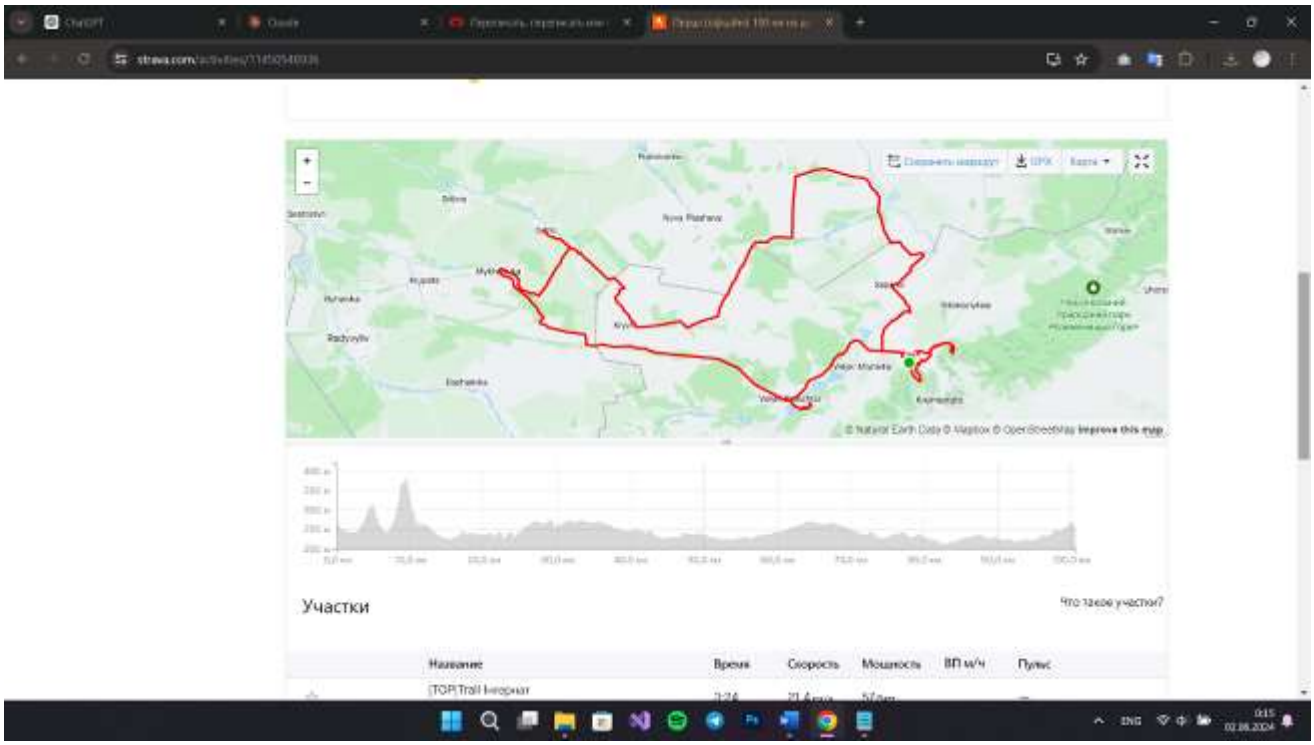


Рисунок 1.2 – Детальний аналіз поїздок [32]

Однією з ключових переваг платформи є статистика, яку вона надає користувачам після завершення тренування [33]. Окрім базових показників пройденої дистанції та тривалості, Strava розраховує середню та максимальну швидкість, загальний набір висоти, інтенсивність зусиль та багато іншої корисної інформації (рисунок 1.3), що дає змогу користувачам ретельно відстежувати свої досягнення та прогрес. Ці детальні статистичні дані не лише допомагають спортсменам відстежувати власний прогрес, ставити нові цілі та мотивувати себе до покращення результатів [34], а й дозволяють порівнювати свої показники з іншими атлетами, робити більш точний аналіз своїх тренувань та коригувати підготовку відповідно до потреб.

Крім того, Strava пропонує користувачам низку корисних інструментів для більш глибокого аналізу їхньої активності. Наприклад, функція "Стрілки" візуалізує темп користувача протягом тренування, допомагаючи виявляти періоди різкої зміни швидкості чи інтенсивності зусиль. Інший інструмент під назвою "Відсотки серцевого ритму" дозволяє оцінити, скільки часу під час тренування користувач перебував у різних зонах інтенсивності, базуючись на показниках серцевого ритму.

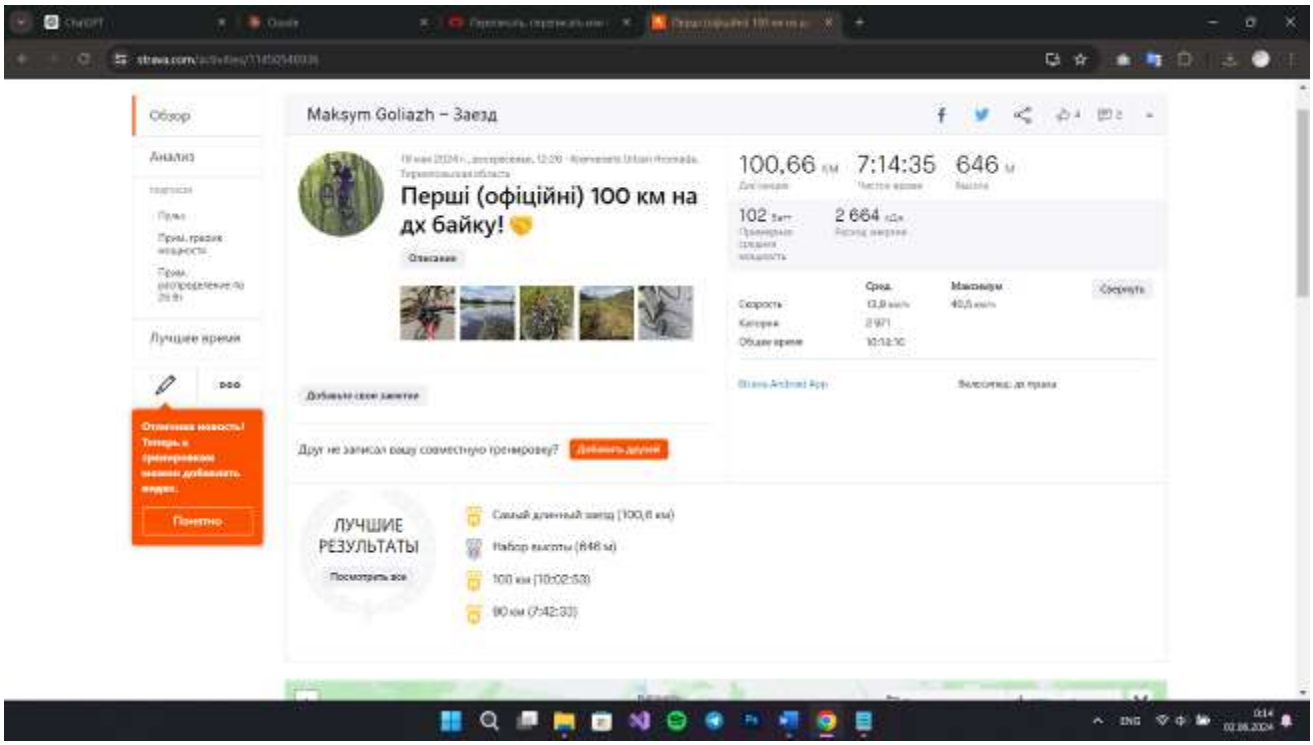


Рисунок 1.3 – Детальний аналіз тренування [35]

Крім того, Strava має потужні інструменти для візуалізації маршрутів активності на інтерактивних картах [36], з можливістю аналізу сегментів траси. Користувачі можуть створювати виклики, змагатися за результати на конкретних сегментах та долучатися до онлайн-спільнот однодумців (рисунок 1.4).

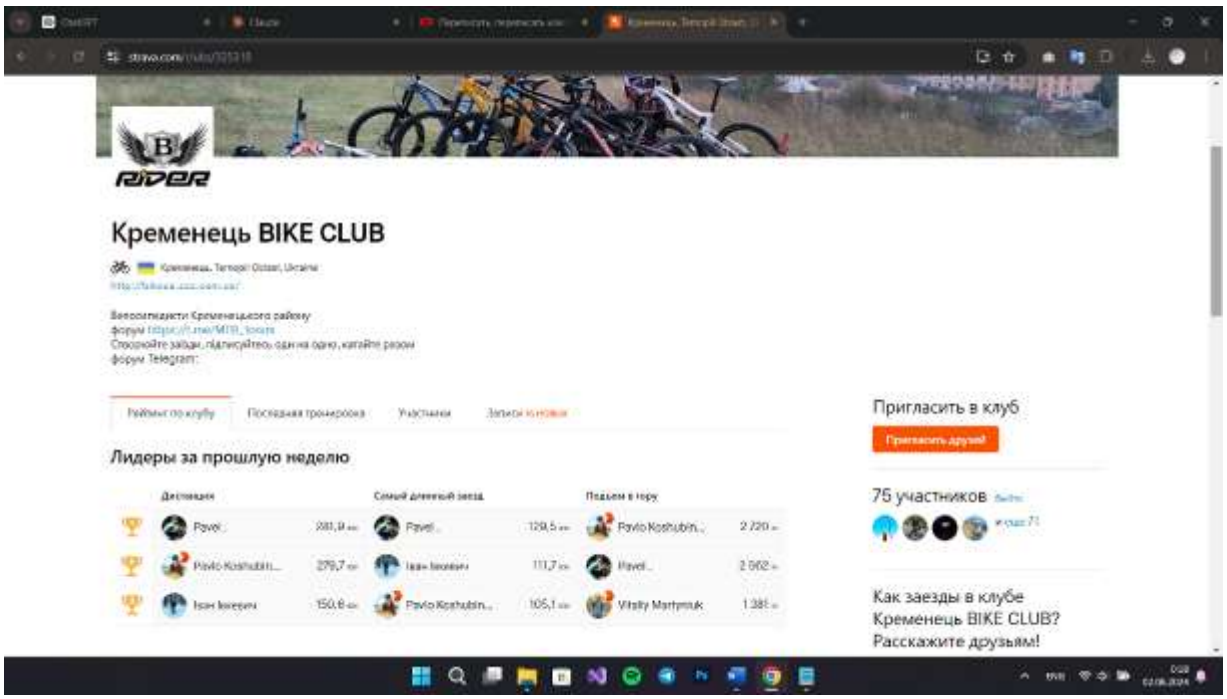


Рисунок 1.4 – Велоклуби в Strava [37]

Проте, незважаючи на багатий функціонал трекінгу та аналітики, Strava не пропонує інструментів для формування повноцінних структурованих програм фізичної підготовки на базі зібраних даних. Планування тренувальних циклів залишається на розсуд самих користувачів або їхніх тренерів. Також відсутні можливості врахування персональних фізичних особливостей, цілей розвитку конкретних навичок, генетичних факторів тощо при розробці програм підготовки.

Strava є однією з популярних програм для відстеження фізичної активності, зокрема для велосипедистів. Вона дозволяє записувати маршрути, відстань, швидкість, темп та інші дані під час тренувань. Проте, незважаючи на корисні функції, Strava не використовує генетичний алгоритм для автоматичного планування та генерування індивідуальних тренувальних програм для користувачів [38].

Strava надає деякі рекомендації щодо тренувань, але вони засновані на загальних принципах та не враховують специфічних індивідуальних факторів кожного користувача, таких як вік, стать, рівень підготовки, цілі та обмеження. Тренувальні плани в Strava зазвичай формуються вручну користувачами або тренерами на основі їхнього досвіду та інтуїції.

TrainingPeaks [39] - професійний інструмент для планування, відстеження та аналізу тренувального процесу спортсменів різних дисциплін (рисунок 1.5).

Це всеохоплююча платформа для планування, відстеження та аналізу тренувань, призначена для атлетів та тренерів різних видів спорту. Ось загальний опис цього інструменту:

TrainingPeaks дозволяє створювати персоналізовані тренувальні плани з урахуванням цілей, рівня підготовки та розкладу користувача. Тренери можуть легко розробляти та коригувати плани, а спортсмени можуть отримувати доступ до своїх тренувальних програм в зручному форматі.

Ця комплексність робить веб платформу цінним ресурсом як для професійних спортсменів, так і для любителів, які прагнуть покращити свої результати.

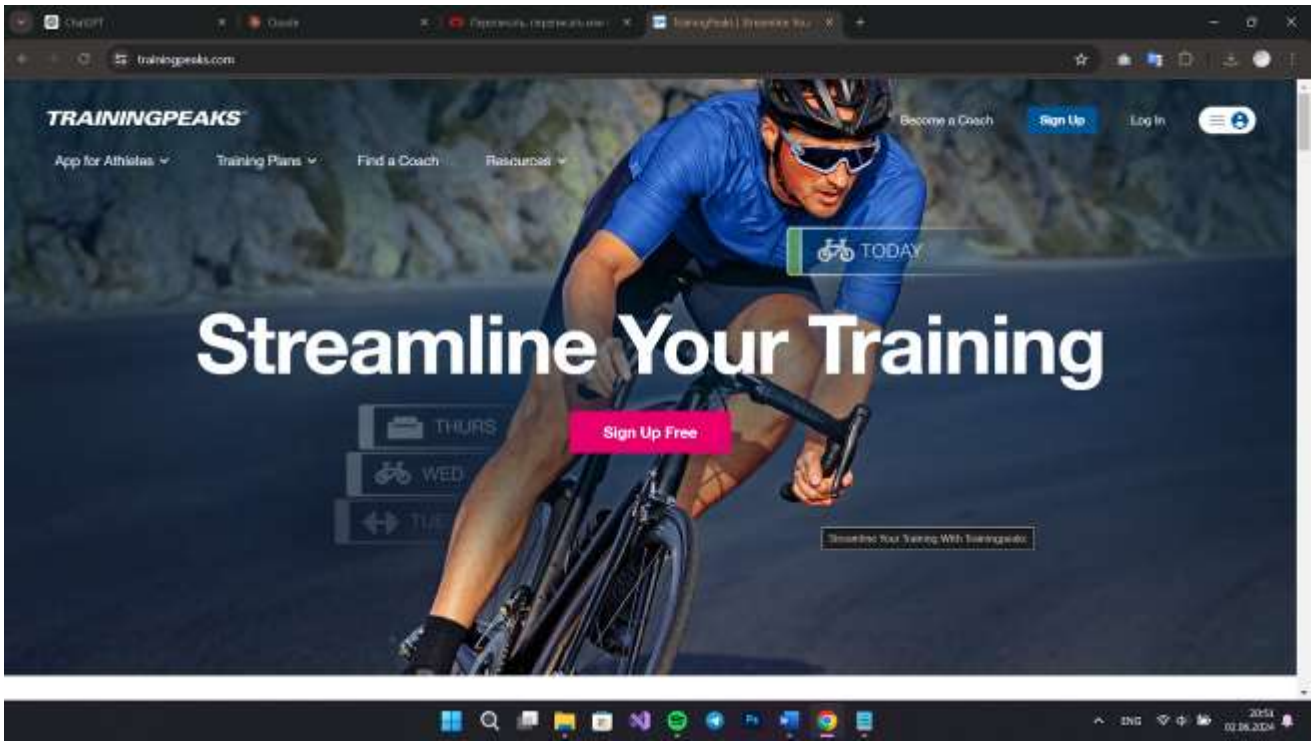


Рисунок 1.5 – Головне меню TrainingPeaks [40]

На відміну від Strava, яка більше орієнтована на широкі маси аматорів, TrainingPeaks [41] спеціалізується на підтримці продуктивної взаємодії між атлетами та їхніми тренерами (рисунок 1.6).

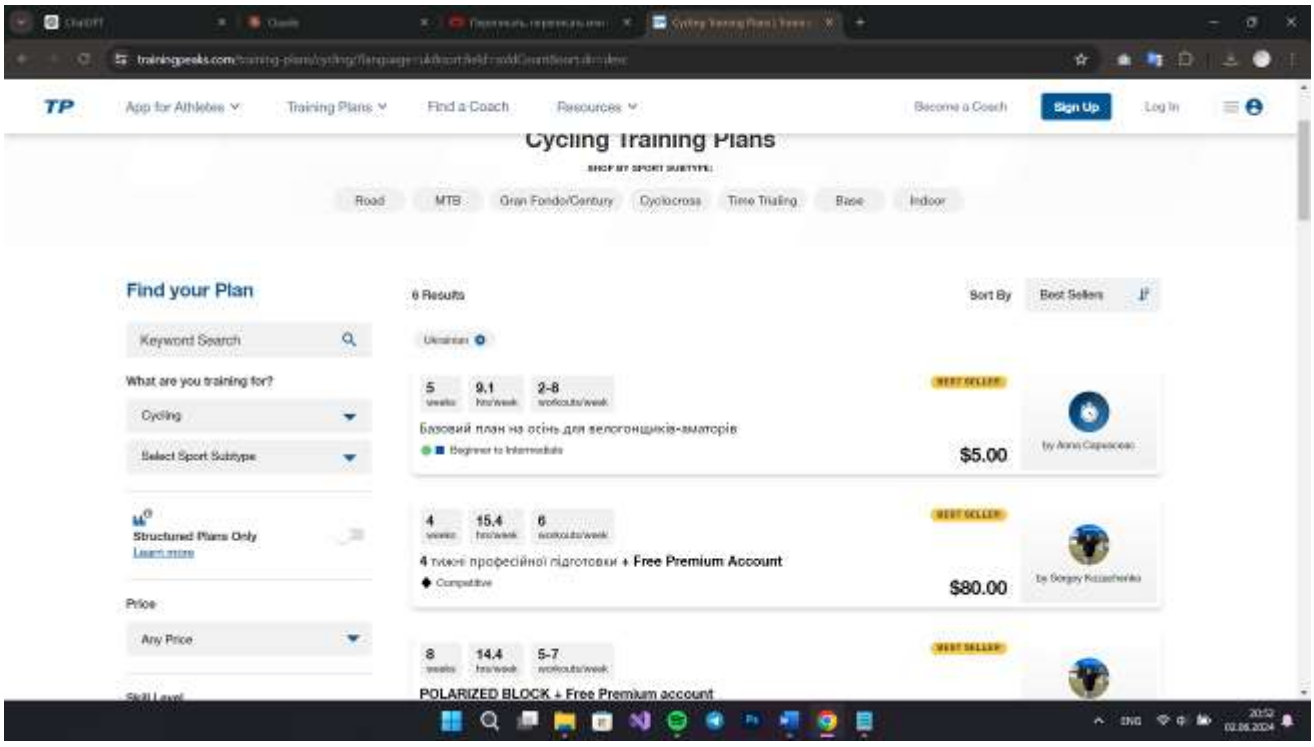


Рисунок 1.6 – Взаємодія між атлетами та тренерами [42]

Основний функціонал платформи зосереджений на створенні детальних щорічних циклів підготовки з розбивкою на блоки і мікро цикли [43]. Тренери можуть планувати навантаження для кожного тренувального дня, встановлювати зони інтенсивності, задавати специфічні вправи (рисунок 1.7). При цьому система дозволяє гнучко коригувати плани залежно від поточного стану спортсмена [44].

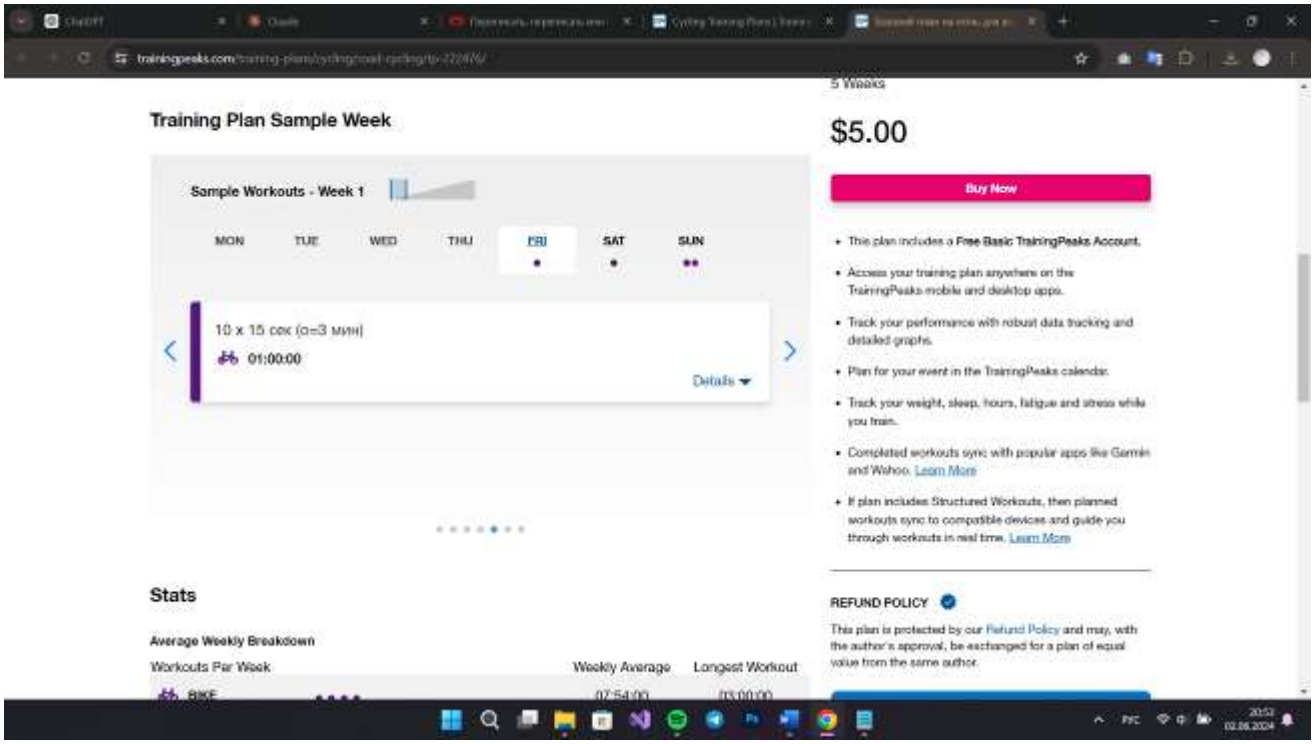


Рисунок 1.7 – Планування тренувань [45]

Платформа також має корисні інструменти для візуалізації та аналізу тренувальних навантажень, надаючи зрозумілі графіки та діаграми (рисунок 1.8). Передбачено можливості інтеграції з календарями, створення груп та команд, налаштування системи сповіщень тощо [46].

Після завершення тренувань TrainingPeaks надає докладну аналітику, включаючи показники потужності, серцевого ритму, швидкості, набору висоти та багато іншого. Ця інформація візуалізується у вигляді графіків, діаграм та звітів, що полегшує відстеження прогресу та виявлення сфер для покращення. Крім того, платформа пропонує можливість порівняння результатів з попередніми тренуваннями або визначеними цілями.

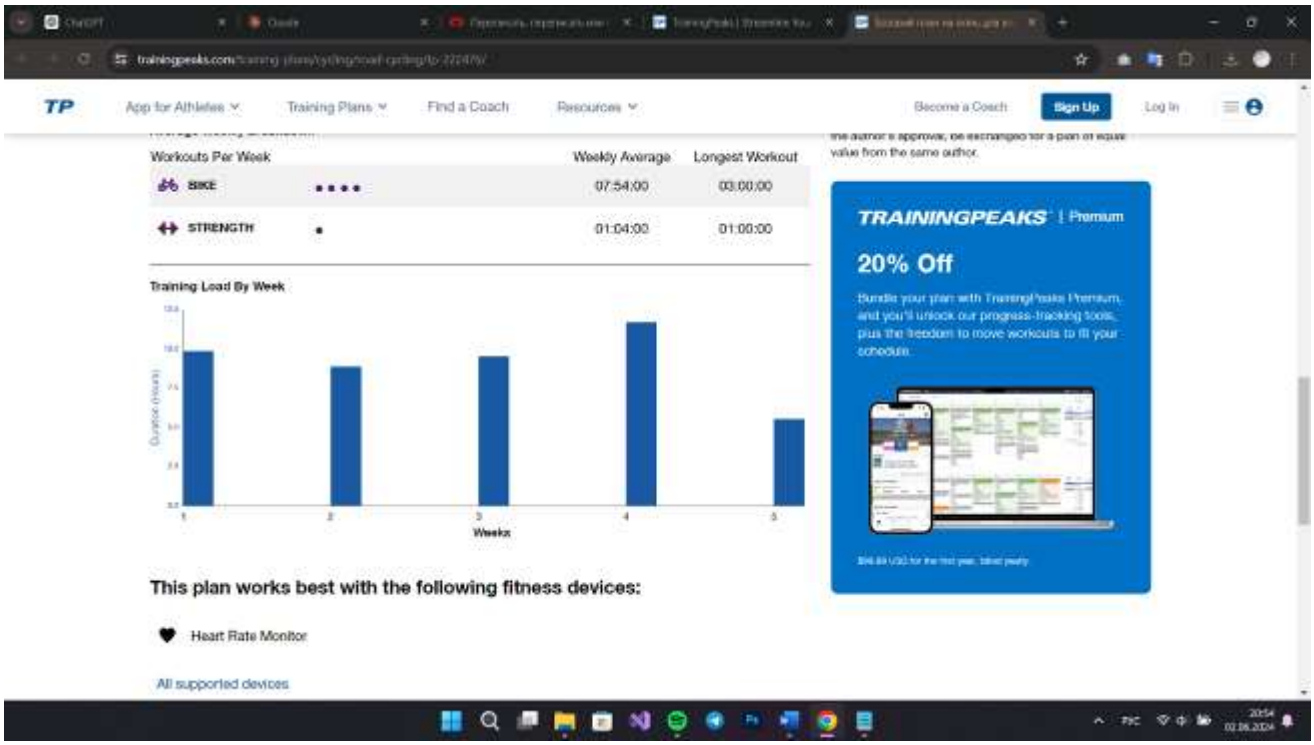


Рисунок 1.8 – Візуалізація та аналіз тренувань [47]

Проте, незважаючи на широкі можливості TrainingPeaks, формування самих тренувальних планів залишається напівавтоматизованим процесом. Тренери мають самостійно обирати вправи, тривалість, інтенсивність та періодичність кожного заняття згідно власного досвіду та інтуїції. Глибока персоналізація програми відповідно до індивідуальних фізичних особливостей, генетичного профілю, точних цілей розвитку конкретних навичок атлета не передбачена [48].

Таким чином, TrainingPeaks є потужним інструментарієм для професійних тренерів, проте не вирішує проблему повної автоматизації та параметризації процесу планування фізичної підготовки. Інтеграція сучасних алгоритмічних методів, таких як генетичні алгоритми, створює можливість вивести якість персоналізованого планування на новий рівень [49].

TrainingPeaks - це потужна платформа для планування та аналізу тренувань, яка широко використовується професійними спортсменами та тренерами. Вона пропонує низку корисних функцій, таких як створення тренувальних планів, відстеження показників виконання, аналіз даних та керування навантаженням.

Однак, як і Strava, TrainingPeaks не використовує генетичний алгоритм для автоматизованого формування персоналізованих тренувальних програм. Натомість, плани тренувань створюються вручну тренерами або самими користувачами на основі їхнього досвіду, інтуїції та загальних рекомендацій.

Наведені приклади демонструють, що більшість наявних рішень фокусується на відстеженні активності та простому плануванні циклів тренувань згідно стандартних схем. Водночас, проблема автоматизованої генерації оптимальних персоналізованих програм підготовки для розвитку фізичних навичок членів спортивного клубу із застосуванням сучасних алгоритмічних підходів, зокрема генетичних алгоритмів, залишається невирішеною.

1.4 Мета та завдання кваліфікаційної роботи

Аналіз предметної області демонструє важливість вирішення поставленого завдання, а використання цифрових технологій є доцільним і результативним підходом.

Метою кваліфікаційної роботи є покращення планування розвитку фізичних навичок учасників велоклубу за рахунок автоматизації процесу формування плану тренувань та врахування індивідуальних особливостей.

Для досягнення поставленої мети визначені такі задачі дослідження:

– провести аналіз методів формування та покращення розвитку фізичних навичків учасників велоклубу;

– провести аналіз інтелектуальних методів для визначення індивідуальних планів тренувань для учасників велоклубу;

– реалізувати метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб системи велоклуб;

– виконати програмну реалізацію методу планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб системи велоклуб;

– провести експериментальне тестування програмної реалізації методу;

– виконати дослідження покращення планування розвитку фізичних навичок учасників велоклубу за рахунок автоматизації процесу формування плану тренувань та врахування індивідуальних особливостей.

Розділ 2 Метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб системи велоклуб

2.1 Загальна структура методу

Представлено схему методу розвитку фізичних навичок за генетичним алгоритмом на рисунку 2.1.

Рисунок 2.1 – Схема загальної структури методу

Ефективне планування розвитку фізичних навичок є ключовим фактором для досягнення бажаних результатів у будь-якому виді спорту, зокрема у велоспорті. Однак, традиційні підходи до формування тренувальних програм часто не враховують індивідуальні особливості учасників, такі як вихідний рівень підготовки, схильність до певних типів навантажень, наявність травм або обмежень. Це може призвести до неоптимальних планів тренувань, що знижує ефективність та мотивацію учасників.

У даній веб системі пропонується інноваційний метод планування розвитку фізичних навичок учасників велоклубу за допомогою генетичного алгоритму для вебсистеми. Цей метод передбачає використання генетичного алгоритму в поєднанні з іншими методами машинного навчання, такими як нейронні мережі та кластеризація, для створення персоналізованих тренувальних програм.

За допомогою цієї схеми реалізовано представлення структури та логіки дослідження. Викладено послідовні етапи методу, досягнуто послідовне виконання кожного етапу роботи.

Запропонована структура методу являє собою комплексний, багатомодульний підхід, який органічно інтегрує передові концепції генетичних алгоритмів з практичними аспектами спортивного тренування та зручністю використання веб-систем. Центральним елементом цієї структури виступає генетичний алгоритм - потужний оптимізаційний механізм, натхненний природними процесами еволюції та спадковості.

Структура починається з збору та обробки даних, який виконує критично важливу функцію агрегації та систематизації різноманітної інформації про учасника велоклубу. Цей модуль не просто накопичує дані, але й структурує їх у чотири ключові категорії: профіль учасника, фізіологічні дані, історія тренувань та обмеження. Історія тренувань та відгуки про них формують контекст прогресу, а категорія обмежень враховує індивідуальні застереження, як-от травми чи графік роботи. Цей етап закладає фундамент для глибоко персоналізованого підходу.

Серцем методу є генетичний алгоритм - передова техніка вибору кращого плану, яка імітує процеси біологічної еволюції. Цей алгоритм розглядає кожен потенційний план тренувань як "індивіда" в популяції, де "гени" представляють різні аспекти плану: типи вправ, їхню інтенсивність, послідовність. Починаючи з ініціалізації різноманітної популяції планів, алгоритм проходить через цикли оцінки, селекції та генетичних операцій.

На етапі оцінки фітнесу кожен план аналізується за допомогою багатоцільової функції, яка враховує різні аспекти фізичної форми - витривалість, силу, гнучкість - з вагами, що відповідають цілям учасника. Плани, які порушують обмеження (наприклад, включають протипоказані вправи), отримують штрафи. Селекція, натхненна природним відбором, надає перевагу кращим планам, використовуючи методи на кшталт турнірного відбору.

Ключовим є етап генетичних операторів. Кросинговер дозволяє двом успішним планам "обмінятися генетичним матеріалом", наприклад, один план може передати ефективний блок силових вправ іншому. Мутації спонтанно змінюють окремі аспекти, збільшуючи інтенсивність конкретного тренування. Інверсія може переставити порядок мікро циклів для кращого відновлення. Через ці процеси алгоритм "вирощує" покоління за поколінням, постійно покращуючи плани.

Після збіжності алгоритму - коли поліпшення сповільнюються або досягається часовий ліміт - виходить оптимізований план тренувань. Але метод не зупиняється тут.

Модуль адаптації та моніторингу забезпечує, щоб план залишався оптимальним з плином часу. На короткостроковому рівні він щодня коригує план на основі суб'єктивних відгуків (наприклад, "відчуваю втому") та об'єктивних даних з носимих пристроїв. У середньостроковій перспективі, кожні 2-4 тижні, аналізується загальний прогрес, що може призвести до перегляду цілей. Довгострокова адаптація охоплює сезонні зміни та використовує накопичені дані для навчання самого алгоритму.

Унікальною особливістю цього методу є модуль візуалізації та пояснення,

який робить складні рішення алгоритму зрозумілими для користувачів. Інтерактивні графіки використовують кольорове кодування для різних типів вправ і дозволяють масштабування від денного до місячного перегляду. Ще важливіше, система пояснює свої рішення: чому обрано певні вправи, як обмеження вплинули на план. Візуалізації прогресу, що порівнюють поточні досягнення з минулими, служать потужним мотиваційним інструментом.

Нарешті, всі ці складні процеси акуратно упаковані в інтуїтивно зрозумілий веб-інтерфейс користувача. У своєму персональному кабінеті учасник має доступ до поточного плану та історії тренувань. Функція щоденного звіту дозволяє швидко ввести самопочуття і отримати скориговані завдання на день. Соціальні функції, як-от можливість поділитися прогресом або участь у групових викликах, додають шар спільноти та здорової конкуренції.

Ця структура методу являє собою гармонійний синтез генетичного алгоритму, спортивної науки та орієнтованого на користувача веб-дизайну. Генетичний алгоритм забезпечує потужність і гнучкість для вирішення складного завдання персоналізації тренувань. Модулі адаптації та моніторингу гарантують, що план еволюціонує разом з учасником. А компоненти візуалізації та веб-інтерфейсу трансформують ці технічні досягнення в досвід, який є не лише ефективним, але й зрозумілим, залучаючим і глибоко персоналізованим. Як результат, члени велоклубу отримують плани розвитку фізичних навичок, які є науково обґрунтованими, динамічно адаптивними та, найголовніше, зручними у використанні.

2.2 Адаптація генетичного алгоритму

Загальна схема роботи генетичного алгоритму зображена на рисунку 2.2. Ця схема демонструє послідовність основних етапів генетичного алгоритму, таких таких як ініціалізація початкової популяції, оцінка придатності, селекція, схрещування.

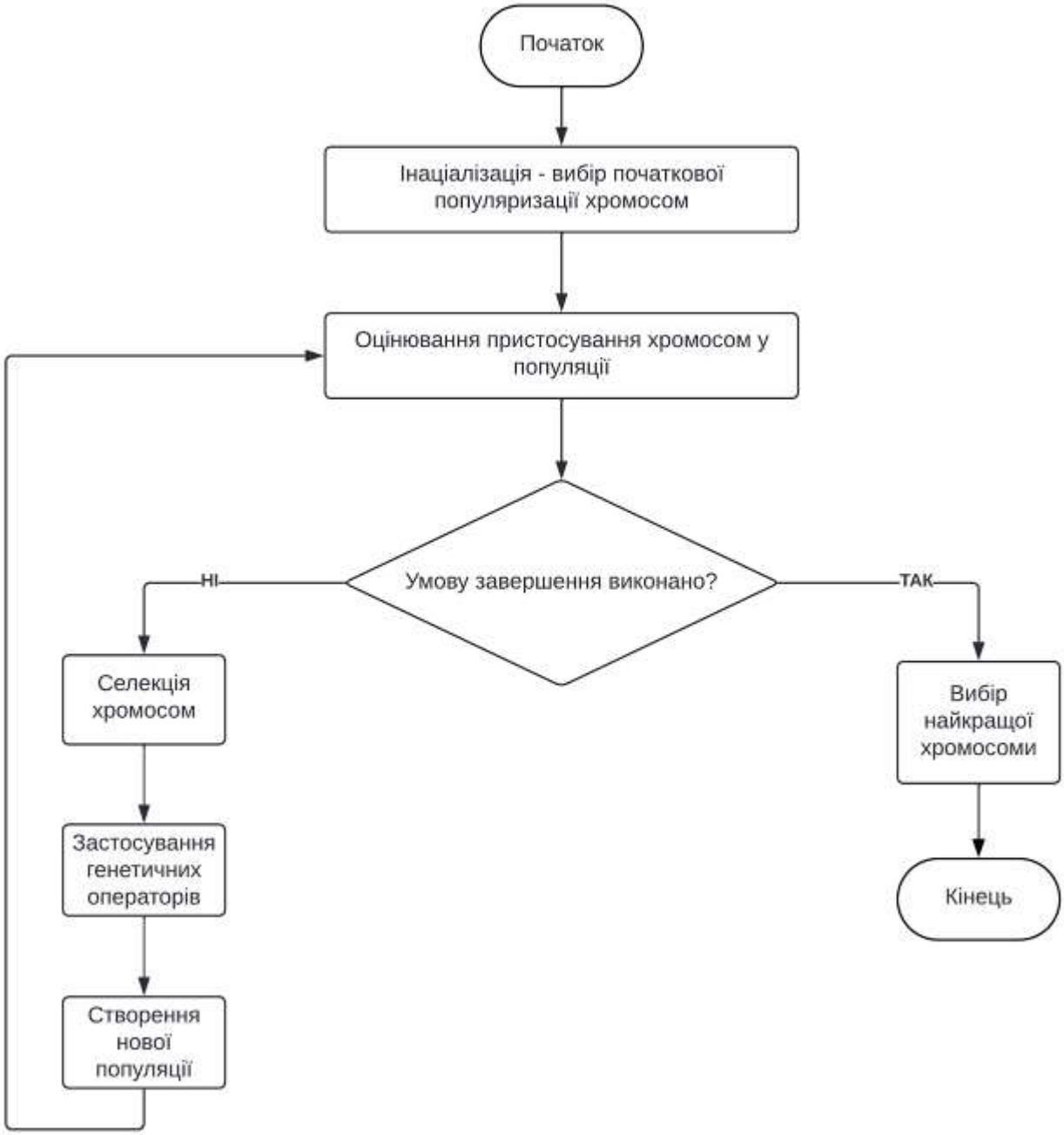


Рисунок 2.2 – Загальна схема генетичного алгоритму

Все починається з ініціалізації - створення початкової популяції можливих рішень, представлених у вигляді хромосом. Кожна хромосома складається з послідовності генів, які кодують різні параметри або властивості рішення.

Далі відбувається оцінювання пристосованості кожної хромосоми в популяції за допомогою спеціальної функції пристосованості (фітнес-функції). Ця функція визначає, наскільки добре дане рішення задовольняє критерії

оптимізаційної задачі.

Після оцінювання алгоритм перевіряє умови завершення, такі як досягнення заданого рівня пристосованості або максимальної кількості ітерацій. Якщо умови не виконано, алгоритм переходить до наступного кроку - селекції хромосом.

Під час селекції відбираються найкращі хромосоми з поточної популяції для створення нового покоління. Ймовірність відбору хромосоми залежить від її пристосованості - хромосоми з вищою пристосованістю мають більші шанси на відбір.

Потім до відібраних хромосом застосовуються генетичні оператори, такі як схрещування (кросовер) і мутація. Схрещування - це обмін генетичним матеріалом між двома хромосомами для створення нових рішень. Мутація вносить невеликі випадкові зміни в гени хромосом, забезпечуючи різноманітність популяції і можливість уникнути локальних оптимумів.

Результатом застосування генетичних операторів є створення нової популяції хромосом - нового покоління рішень. Цей цикл (оцінка пристосованості, селекція, застосування генетичних операторів і створення нової популяції) повторюється доки не будуть виконані умови завершення.

Коли умови завершення задовольняються, алгоритм вибирає найкращу хромосому з поточної популяції в якості остаточного оптимального або достатньо хорошого рішення задачі.

Схема адаптації генетичного алгоритму для запропонованого методу представлена на рисунку 2.3. Зображення ілюструє основні компоненти та процеси, які задіяні в роботі алгоритму. Цей метод має на меті знайти оптимальний баланс між інтенсивністю навантаження, різноманітністю вправ та відповідністю індивідуальним потребам, забезпечуючи максимальну ефективність тренувального процесу та мінімізуючи ризики травм чи перенавантаження.

Рисунок 2.3 – Схема адаптація генетичного алгоритму

Хромосома представляє цілісний план тренувань для одного учасника. Вона складається з окремих генів, які кодують різні аспекти тренувального процесу: вид вправи, її інтенсивність та тривалість.

Популяція - це набір можливих планів тренувань для всіх учасників велоклубу. Кожна хромосома в популяції є унікальним планом.

Функція придатності (фітнес-функція) оцінює якість кожного плану тренувань відповідно до певних критеріїв, таких як рівень фізичного навантаження, відповідність індивідуальним потребам учасника та різноманітність вправ.

Селекція - це процес вибору найкращих планів тренувань з поточної популяції для подальшого "розмноження". Плани з вищою придатністю матимуть більшу ймовірність відбору.

Під час роботи генетичного алгоритму, відібрані плани тренувань піддаються генетичним операторам - схрещуванню (обмін частинами планів між хромосомами) та мутації (невеликі випадкові зміни в планах). Це дозволяє

створювати нові, потенційно кращі плани на наступну ітерацію.

Таким чином, шляхом еволюційного підходу - селекції, схрещування та мутації - генетичний алгоритм поступово покращує популяцію планів тренувань для учасників велоклубу з метою знайти найкращі комбінації вправ, їх інтенсивності та тривалості для розвитку фізичних навичок.

Адаптована схема генетичного алгоритму методу зображена на рисуноку 2.4.

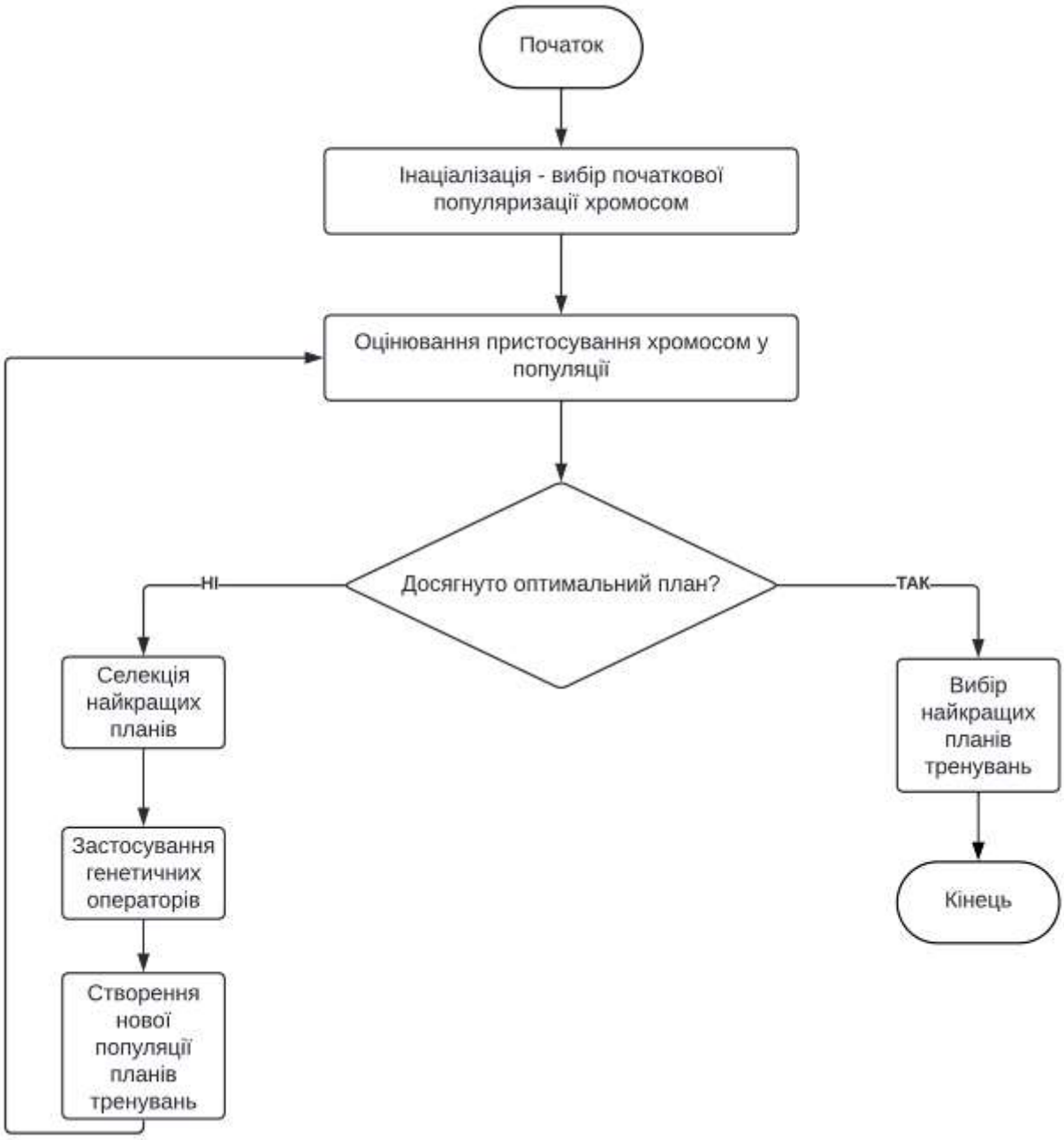


Рисунок 2.4 – Адаптована схема методу

Все починається з початку - старт процесу генетичного алгоритму.

Ініціалізація - вибір початкової популяції хромосом (планів тренувань). Це здійснюється шляхом генерації випадкових планів тренувань, які кодуються у вигляді хромосом. Кожна хромосома представляє унікальний план тренувань з різними типами вправ, інтенсивністю, тривалістю тощо.

Оцінювання пристосування хромосом у популяції - це крок, на якому визначається, наскільки кожен план тренувань відповідає поставленим цілям та обмеженням. Пристосованість плану оцінюється за допомогою спеціальної функції придатності, яка враховує різні фактори, такі як очікуваний приріст фізичних навичок, відповідність індивідуальним особливостям учасника тощо.

Досягнуто оптимальний план? - перевірка критеріїв зупинки. Якщо знайдено оптимальний або достатньо хороший план тренувань, процес завершується. В іншому випадку відбувається перехід до наступних кроків.

Селекція найкращих планів - відбір певної кількості найбільш пристосованих хромосом (планів) за допомогою методу селекції.

Застосування генетичних операторів - до відібраних планів застосовуються генетичні оператори, такі як схрещування та мутація, для створення нової популяції планів. Схрещування дозволяє комбінувати частини різних планів, а мутація вносить невеликі зміни у плани для забезпечення різноманітності.

Створення нової популяції планів тренувань - на основі відібраних, схрещених та мутованих планів формується нова популяція для наступної ітерації.

Цей процес повторюється ітераційно, поки не буде знайдено задовільний план тренувань для кожного учасника велоклубу, враховуючи їхні індивідуальні цілі та обмеження.

Таким чином, ця схема демонструє ітераційний процес еволюційного пошуку, де в кожній новій ітерації генеруються покращені плани тренувань на основі попередніх рішень за допомогою генетичних операторів.

2.3 Імплементація генетичного алгоритму

Імплементований генетичний алгоритм відповідно до методу реалізації програми зображений на рисунку 2.5.

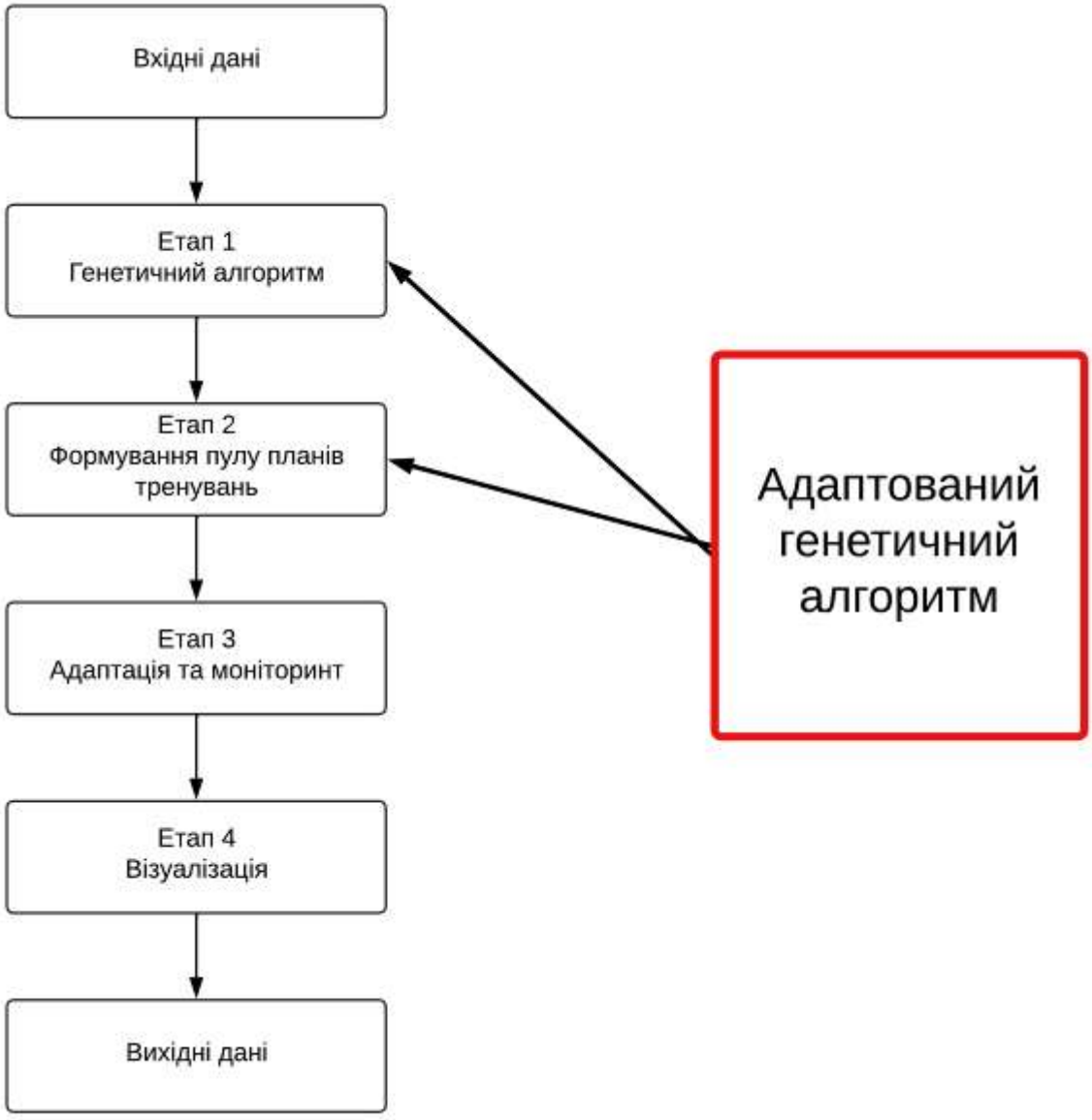


Рисунок 2.5 – Імплементація генетичного алгоритму для системи планування розвитку фізичних навичок учасників велоклубу

Ця схема ілюструє комплексний процес застосування адаптованого генетичного алгоритму для планування розвитку фізичних навичок учасників велоклубу. Процес починається зі збору вхідних даних про кожного учасника,

включаючи їхні профілі, фізіологічні показники, історію тренувань та індивідуальні обмеження.

Ядром системи є етап 1 - генетичний алгоритм. Тут відбувається ітераційний пошук оптимальних планів тренувань шляхом еволюційного моделювання. Спочатку генерується початкова популяція випадкових планів, закодованих у формі хромосом. Далі обчислюється пристосованість кожного плану за спеціальною функцією фітнесу, що враховує різноманітні чинники, такі як очікуваний прогрес у розвитку навичок, дотримання обмежень та індивідуальні особливості учасника. На основі оцінки фітнесу відбувається селекція найкращих планів для подальшого розмноження. Відібрані плани зазнають дії генетичних операторів схрещування, мутації та інверсії, що дозволяє створювати нові, потенційно кращі рішення. Сформоване таким чином нове покоління планів оцінюється знову, і цикл повторюється до досягнення визначених критеріїв збіжності.

На етапі 2 - формування пулу тренувань відбувається кінцеве формування персоналізованих планів для кожного учасника на основі найкращих розв'язків, отриманих генетичним алгоритмом з урахуванням усіх індивідуальних особливостей та вимог.

Під час реалізації планів тренувань, етап 3 - адаптація та моніторинг забезпечує гнучке коригування планів на основі щоденних відгуків учасників, аналізу їхнього прогресу та за потреби оновлення цілей розвитку фізичних навичок.

На етапі 4 - візуалізація згенеровані плани тренувань візуалізуються в інтерактивному інтерфейсі, що дозволяє учасникам наочно бачити свій прогрес, отримувати пояснення щодо прийнятих рішень і порівнювати себе з іншими.

Вихідними даними процесу є сформовані індивідуальні плани розвитку фізичних навичок для кожного учасника. Водночас ці вихідні дані, що містять оновлену інформацію про прогрес, повертаються на вхід як нові вхідні дані, завдяки чому адаптований генетичний алгоритм може ітераційно уточнювати й покращувати плани з урахуванням змін у стані учасників.

2.4 Проектна архітектура методу

Проектна архітектура веб-системи велоклубу, що реалізує метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму зображена на рисунку 2.6.

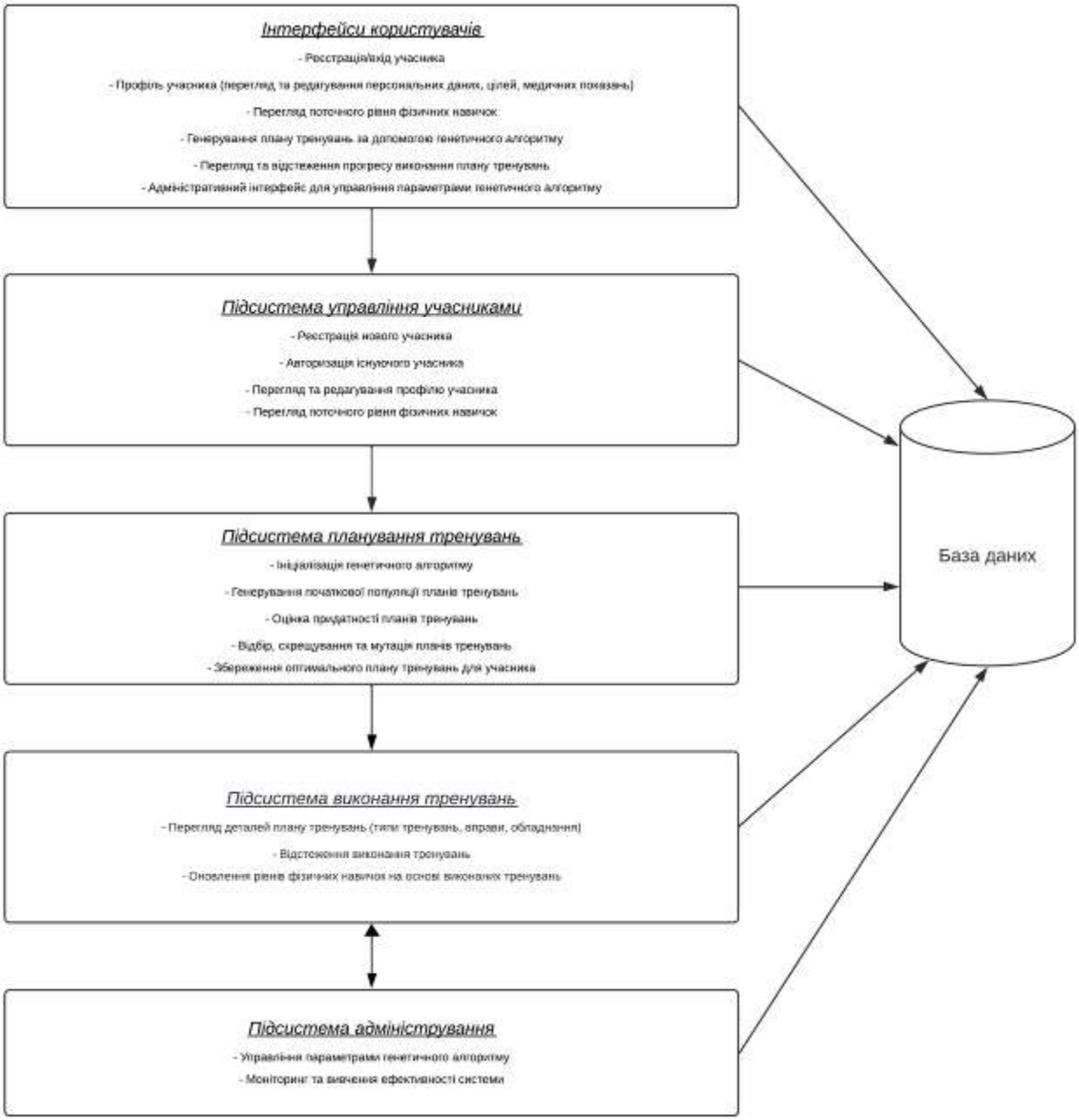


Рисунок 2.6 – Проєктна архітектура системи

Ця схема ілюструє загальну архітектуру програмної системи для автоматизованого планування персоналізованих тренувальних програм із застосуванням генетичного алгоритму. Система складається з кількох взаємопов'язаних підсистем, які забезпечують різні функціональні можливості.

Інтерфейси користувачів:

– реєстрація нового учасника: забезпечує зручний спосіб створення облікових записів для нових учасників велоклубу;

– профіль учасника: дозволяє учасникам переглядати та редагувати свої персональні дані, цілі тренувань, вподобання (наприклад, улюблені види вправ) та аналізувати результати тестів фізичних навичок;

– перегляд поточного рівня фізичних навичок: візуалізує поточний стан фізичних навичок учасника, таких як витривалість, сила, гнучкість, швидкість тощо, на основі історії тренувань та тестів;

– генерування нового тренувального плану: інтерфейс для запуску генетичного алгоритму, задання параметрів та цілей оптимізації, а також отримання згенерованого персоналізованого плану тренувань;

– перегляд та відстеження прогресу виконання плану тренувань: надає учасникам доступ до деталей їхнього поточного плану тренувань, дозволяє відмічати виконані тренування та візуально спостерігати свій прогрес;

– адміністративний інтерфейс: забезпечує адміністраторам можливість налаштовувати параметри генетичного алгоритму, такі як кількість поколінь, ймовірності мутацій, вагові коефіцієнти у фітнес-функції тощо.

Підсистема управління учасниками:

– реєстрація нового учасника: обробляє дані реєстрації, створює новий обліковий запис в базі даних та зберігає введену інформацію про учасника;

– авторизація існуючого учасника: перевіряє облікові дані (логін, пароль) для надання доступу до системи;

– перегляд та редагування профілю учасника: забезпечує функціонал для завантаження, оновлення та збереження змін в профілі учасника;

– перегляд поточного рівня фізичних навичок: обчислює поточні оцінки

фізичної форми учасника (витривалість, сила, гнучкість тощо) на основі даних у базі даних про виконані тренування та тести.

Підсистема планування тренувань:

– ініціалізація генетичного алгоритму: завантажує параметри та налаштування алгоритму з бази даних для підготовки до роботи;

– генерування початкової популяції планів тренувань: створює випадкові плани - "індивіди" для ініціалізації популяції, враховуючи профіль та потреби конкретного учасника;

– оцінка придатності планів тренувань: обчислює значення фітнес-функції для кожного плану в популяції, використовуючи дані про цілі учасника та обмеження для визначення "придатності";

– вибір, схрещування та мутація планів тренувань: застосовує генетичні оператори - селекцію (з використанням турнірного відбору або елітарної стратегії), схрещування (одноточкове, рівномірне) та мутацію для генерування нових поколінь планів;

– збереження оптимального плану тренувань: після досягнення критерію збіжності (максимальна кількість поколінь або стабілізація фітнес-функції) зберігає найкращий згенерований план тренувань для учасника в базі даних.

Підсистема виконання тренувань:

– перегляд деталей плану тренувань: візуалізує детальну інформацію про поточний план тренувань учасника, включаючи типи вправ, інтенсивність, періодизацію, кількість циклів тощо;

– відстеження виконання тренувань: дозволяє учасникам відмічати, які саме тренування з плану були виконані, отримуючи цю інформацію з інтерфейсу користувача;

– оновлення рівня фізичних навичок: після виконання певних тренувань або проведення тестів обраховує нові показники фізичної форми учасника (витривалість, сила, гнучкість тощо) та оновлює відповідні записи в базі даних.

База даних (БД): база даних виконує функцію центрального сховища інформації для всієї системи, забезпечуючи зберігання та доступ до таких даних:

– облікові дані учасників (логіни, паролі) для авторизації;

– профілі учасників з персональною інформацією, цілями, вподобаннями;

– історія виконаних тренувань та результатів тестів для відстеження прогресу кожного учасника;

– поточний стан фізичних навичок (витривалість, сила, гнучкість тощо) для всіх учасників;

– налаштування та параметри генетичного алгоритму для ініціалізації та його належного функціонування;

– усі згенеровані плани тренувань із деталями для кожного учасника.

База даних взаємодіє з усіма підсистемами, отримуючи та надаючи необхідні дані для належної роботи системи: зберігаючи облікові дані та профілі учасників, отримуючи налаштування алгоритму, зберігаючи згенеровані плани та оновлюючи показники фізичної форми.

Таким чином, діаграма представляє цілісну архітектуру, де кожен компонент відіграє важливу роль у забезпеченні належного функціонування веб-системи для планування розвитку фізичних навичок учасників велоклубу за допомогою генетичного алгоритму.

2.5 Інформаційна структура програмної реалізації методу

Ефективне функціонування системи планування розвитку фізичних навичок учасників велоклубу значною мірою залежить від правильного проектування її інформаційної структури та бази даних. Оскільки система має оперувати великими обсягами даних про учасників, їхні профілі, тренувальні програми, результати тощо, необхідно забезпечити надійне та масштабоване сховище даних.

ER-діаграма інформаційного методу для даної веб системи зображена на рисунку 2.7.

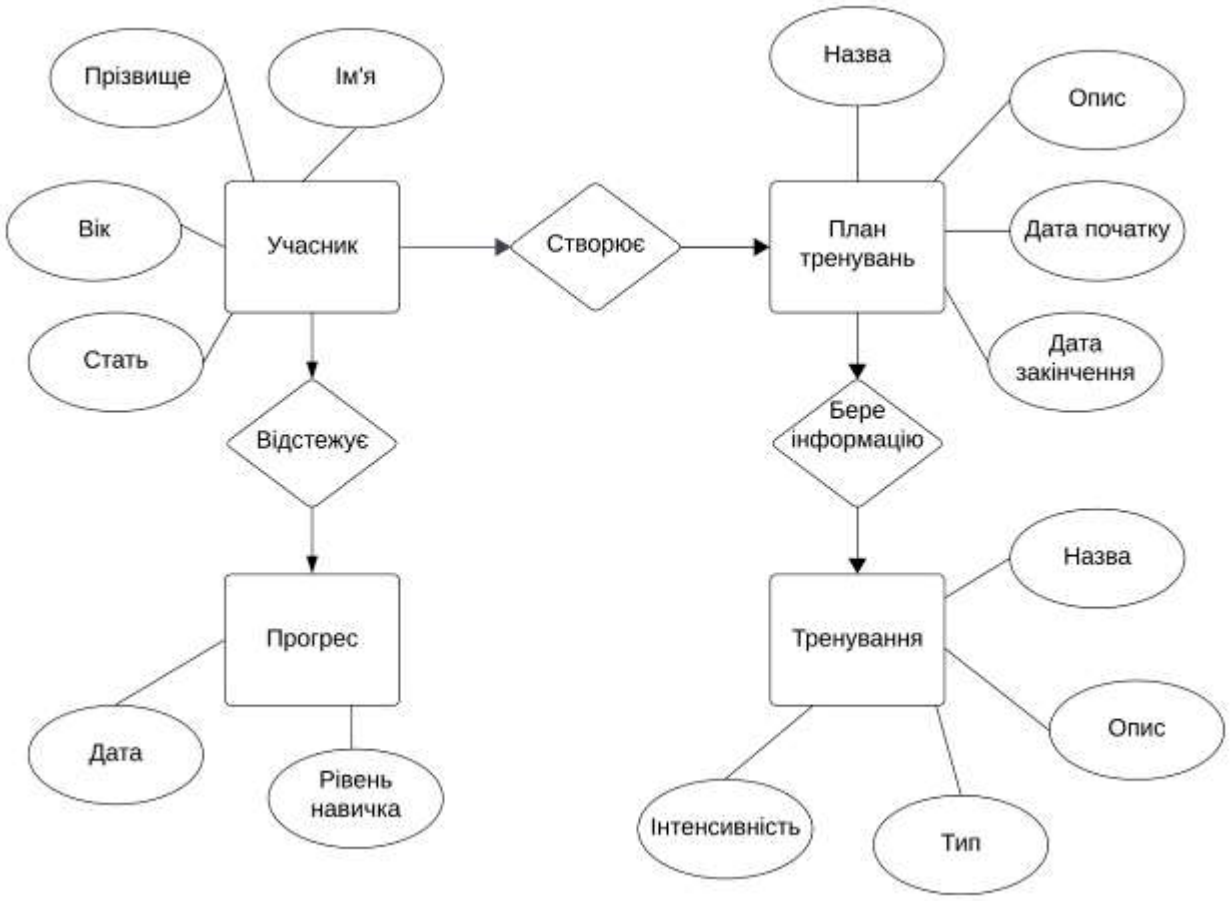


Рисунок 2.7 – Діаграма програмної реалізації методу

Центральною сутністю є "Учасник" з атрибутами "Ім'я", "Вік", "Стать", "Прізвище". Учасник пов'язаний за допомогою зв'язку "створює". Це означає, що кожен учасник створює певний стан, який відображає його/її поточний фізичний стан, важливий для планування тренувань.

Сутність "Учасник" також пов'язана з сутністю "Прогрес" через зв'язок "відстежує". Прогрес фіксує рівень навичок учасника на певну дату, відображаючи динаміку розвитку його фізичних здібностей.

Сутність "План тренувань" містить інформацію про майбутні тренувальні заняття, включаючи опис плану, дати початку та завершення. Зв'язок "Бере інформацію" вказує, що план тренувань складається з окремих тренувальних занять, описаних у сутності "Тренування".

"Тренування" деталізує окреме тренувальне заняття, визначаючи його опис, інтенсивність та тип. Кожне тренування також має коротку назву, представлену сутністю "Назва".

Таким чином, ця er-діаграма відображає зв'язки між учасниками, їхнім поточним станом, прогресом у розвитку навичок, а також планами тренувань та окремими тренувальними заняттями, що їх складають. Зв'язки між сутностями забезпечують належну організацію та цілісність даних системи планування тренувань.

Схематична структура бази даних для системи зображена на рисунку 2.8.

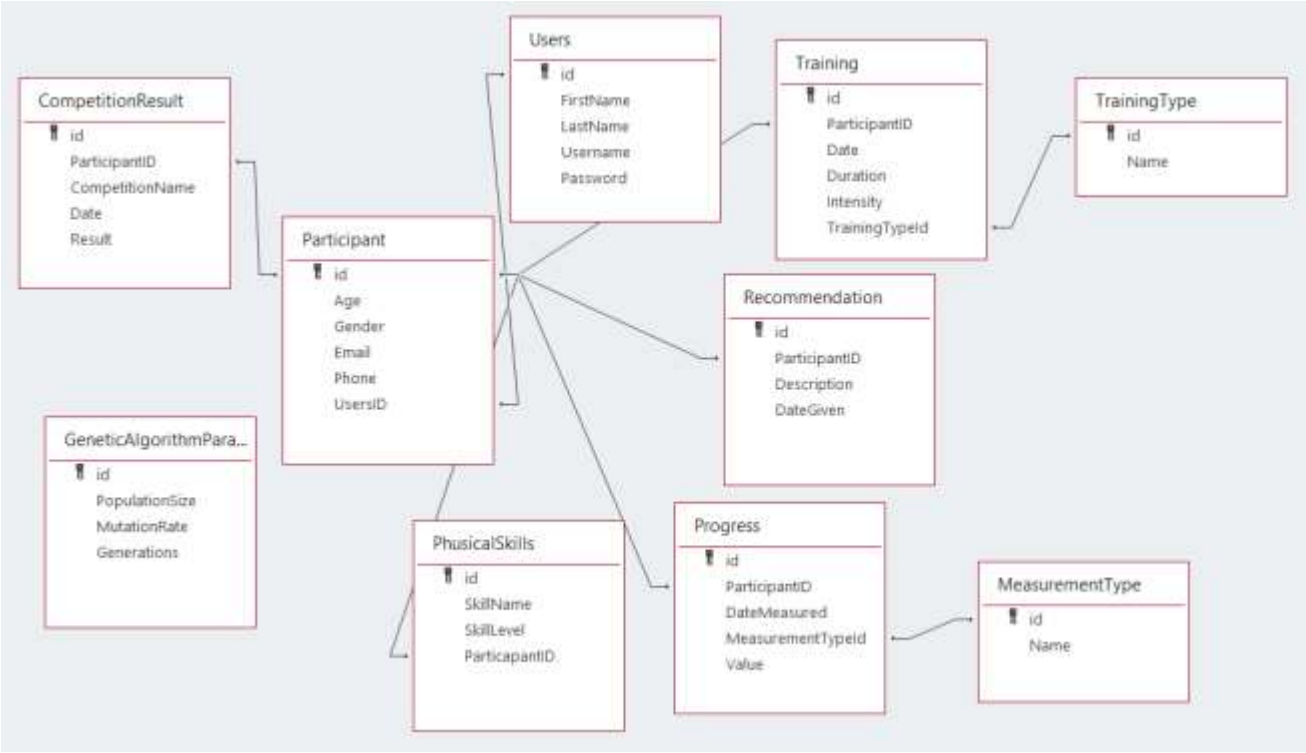


Рисунок 2.8 – Схематична структура бази даних

Таблиця users зберігає інформацію про користувачів при реєстрації. Вона має наступні атрибути:

– id - первинний ключ, унікальний ідентифікатор;

– firstname - ім’я користувача;

– lastname - прізвище користувача;

– username - унікальне ім’я користувача у веб системі;

– password - пароль користувача.

У таблиці participant зберігається детальна інформація про користувачів у веб системі. Має наступні атрибути:

– id - первинний ключ, унікальний ідентифікатор;

– age - кількість років користувачеві;

– gender - стать користувача (чоловіча/жіноча);

– email - електрона пошта користувача;

– phone - мобільний номер телефону користувача;

– usersid - інформація з таблиці Users про дані для входу коритувачів.

Таблиця training зберігає у собі інформацію про тренерування. Вона має наступні атрибути:

– id - первинний ключ, унікальний ідентифікатор;

– firstname - ім’я користувача;

– participantId - інформація з таблиці Participant про розширені дані користувачів;

– date - дата тренерування;

– duration - тривалість тренерування;

– trainingtypeid - інформація з таблиці TrainingType що містить інформація про види тренерувань;

– itensity - інтенсивність тренерувань.

У таблиці trainingtype зберігаються види тренерувань. Має наступні атрибути:

– id - первинний ключ, унікальний ідентифікатор;

– name - назва тренерування("кардіо", "силове", "інтервальне").

Таблиця recommendation буде містити інформацію про рекомендації для учасника клубу. Вона має атрибути:

– id - первинний ключ, унікальний ідентифікатор;

– participantId - інформація з таблиці Participant про розширені дані користувачів;

– description - детальний опис рекомендації;

– datagiven - дата видачі рекомендації.

У таблиці progress зберігається інформація про прогрес тренувань учасників клубу. Вона має наступні атрибути:

– id - первинний ключ, унікальний ідентифікатор;

– participantid - інформація з таблиці Participant про розширені дані користувачів;

– datemeasured - вимірювані дані;

– measurementtypeid - інформація з таблиці MeasurementType, про тип прогресу;

– value - значення прогресу.

Таблиця measurementtype буде зберігати інформацію про типи прогресу. Вона має наступні атрибути:

– id - первинний ключ, унікальний ідентифікатор;

– name - назва прогресу, наприклад: "час на дистанції", "вага".

Таблиця physicalskill містить інформацію про фізичні навички учасників вело клубу. Буде мати наступні атрибути:

– id - первинний ключ, унікальний ідентифікатор;

– skillname - назва навичка;

– skilllevel - рівень навичка.

У таблиці competitionresult знаходиться інформація про результати змагань. Вона містить атрибути:

– id - первинний ключ, унікальний ідентифікатор;

– participantid - інформація з таблиці Participant про розширені дані користувачів;

– competitionname - назва результату змагань;

– date - дата результату;

– result - результат змагань.

Таблиця geneticalgorithmparameters буде містити інформацію необхідну для роботи генетичного алгоритму. Вона має наступні атрибути:

– id - первинний ключ, унікальний ідентифікатор;

– populationsize - розімір популяції генетичного алгоритму;

– mutationrate - швидкість мутації;

– generation - кількість генерацій в алгоритмі.

Описана схема бази даних є дуже важливою для функціонування веб системи.

2.6 Підготовка робочих вхідних даних для системи

Ефективність роботи системи планування розвитку фізичних навичок учасників велоклубу залежить від якості та повноти вхідних даних, які будуть використовуватися для генерації персоналізованих тренувальних програм. Підготовка належних робочих вхідних даних є критичним етапом, який забезпечує точність та релевантність результатів системи.

Для коректної роботи системи необхідно визначити індивідуальні характеристики кожного учасника. Ці характеристики будуть представлені у вигляді генів у генетичному алгоритмі:

– вік: вік учасника впливає на його фізичні можливості та темпи розвитку. Цей параметр може бути представлений у вигляді цілого числа (років);

– стать: стать учасника також може впливати на фізичні здібності та особливості тренувального процесу. Цей параметр може бути закодований як "0" для чоловіків та "1" для жінок;

– поточний рівень фізичної підготовки: необхідно оцінити поточний рівень фізичної підготовки кожного учасника за певними критеріями (наприклад, витривалість, сила, гнучкість). Ці показники можуть бути представлені як числа в межах від 0 до 1, де 0 відповідає найнижчому рівню, а 1 – найвищому;

– цілі та бажаний рівень фізичної підготовки: учасники можуть ставити різні цілі щодо покращення своїх фізичних здібностей. Ці цілі також можуть бути представлені як числа в межах від 0 до 1 для різних критеріїв;

– обмеження за часом та іншими ресурсами: деякі учасники можуть мати

обмеження щодо кількості часу, який вони можуть приділяти тренуванням, або доступності певних ресурсів (наприклад, спортивного інвентарю). Ці обмеження можуть бути закодовані як додаткові гени.

Для планування тренувань необхідно визначити характеристики різних видів тренувальних вправ. Ці характеристики також будуть представлені як гени в генетичному алгоритмі:

– типи тренувань: система повинна враховувати різні типи тренувань, такі як кардіо (біг, велосипед), силові вправи (з вагою), розтяжка та інші. Кожен тип тренувань може бути закодований як окремий ген;

– інтенсивність тренувань: інтенсивність тренувань (низька, середня, висока) впливає на ефективність розвитку фізичних здібностей. Цей параметр може бути представлений як ціле число або діапазон значень;

– тривалість тренувань: тривалість кожного тренування (у хвилинах або годинах) також є важливим параметром. Цей параметр може бути представлений як ціле число.

Після визначення параметрів учасників та характеристик тренувань, необхідно закодувати їх у вигляді генів для генетичного алгоритму:

– представлення індивідуальних характеристик учасників як генів: кожен параметр учасника (вік, стать, поточний рівень фізичної підготовки тощо) може бути представлений як окремий ген у хромосомі, яка представляє цього учасника;

– представлення характеристик тренувань як генів: аналогічно, кожна характеристика тренувань (тип, інтенсивність, тривалість тощо) може бути закодована як окремий ген у хромосомі, яка представляє певний тренувальний план;

– визначення функції пристосованості (fitness function): необхідно розробити функцію пристосованості, яка оцінюватиме якість кожного плану тренувань для конкретного учасника. Ця функція може враховувати відповідність плану цілям учасника, його обмеженням та іншим критеріям.

Для запуску генетичного алгоритму необхідно сформувати початкову

популяцію планів тренувань:

– генерація випадкових планів тренувань як початкової популяції: початкова популяція може бути згенерована випадковим чином, шляхом комбінування різних характеристик тренувань;

– забезпечення різноманітності початкової популяції: для покращення ефективності генетичного алгоритму важливо забезпечити різноманітність початкової популяції. Це можна зробити, використовуючи різні стратегії генерації випадкових планів.

Після підготовки вхідних даних та формування початкової популяції, необхідно налаштувати параметри генетичного алгоритму для досягнення оптимальних результатів:

– визначення параметрів генетичного алгоритму: необхідно визначити розмір популяції, ймовірності мутації та схрещування, а також інші параметри алгоритму;

– проведення тестових запусків для оцінки ефективності алгоритму: для оцінки ефективності алгоритму необхідно провести серію тестових запусків з різними параметрами та вхідними даними;

– налаштування параметрів для покращення результатів: на основі результатів тестових запусків необхідно налаштувати параметри генетичного алгоритму для досягнення кращих результатів планування тренувань.

Цей детальний опис допоможе зрозуміти процес підготовки вхідних даних та налаштування генетичного алгоритму для системи планування розвитку фізичних навичок учасників велоклубу.

2.7 Критерії оцінювання покращення зручності планування розвитку фізичних навичок учасників велоклубу

Основними критеріями є реальне покращення фізичних навичок учасників за об'єктивними показниками після виконання автоматично

сформованих планів, а також здатність методу формувати ефективні та збалансовані плани з урахуванням індивідуальних особливостей учасників.

Головною метою є покращення результативності процесу формування персоналізованих планів тренувань для кожного учасника з урахуванням його індивідуальних особливостей та поточного рівня підготовки.

Критерії оцінювання покращення планування розвитку фізичних навичок учасників велоклубу:

1. Покращення показників фізичних навичок учасників після виконання автоматично сформованого плану тренувань в порівнянні з вихідним рівнем цих навичок до початку тренувань. Показниками є об'єктивні виміри, такі як:

– збільшення часу витривалості на визначеній дистанції;

– збільшення кількості повторень силових вправ;

– покращення максимальної потужності (для оцінки швидкісних навичок);

– інші відповідні показники для конкретних фізичних навичок.

2. Здатність автоматизованого методу формувати збалансовані плани тренувань, які рівномірно розвивають різні фізичні навички учасника (витривалість, силу, швидкість тощо) відповідно до його індивідуальних потреб.

3. Можливість автоматизованого методу враховувати поточний рівень фізичних навичок учасника та формувати персоналізований план з поступовим нарощуванням навантажень для їх розвитку.

4. Здатність методу динамічно коригувати план тренувань в процесі його виконання учасником відповідно до проміжних результатів розвитку фізичних навичок.

5. Оптимальне врахування індивідуальних особливостей учасника (вік, стать, рівень підготовки, цілі тренувань тощо) при формуванні персоналізованого плану розвитку фізичних навичок.

Схема критерії оцінювання покращення зручності планування розвитку фізичних навичок учасників велоклубу зображена на рисунку 2.9.

Рисунок 2.9 – Схема критерів оцінювання

Спочатку необхідно визначити фізичні навички, які потребують розвитку для даного учасника, наприклад, витривалість, силу або швидкість. Потім проводиться вхідне тестування з метою вимірювання поточного рівня цих навичок учасника за допомогою об'єктивних показників, таких як час витривалості або кількість повторень силових вправ.

Після цього за допомогою методу формується індивідуальний план тренувань для цього учасника. Учасник виконує сформований план протягом визначеного періоду часу.

По завершенні тренувального циклу проводиться вихідне тестування учасника з використанням тих самих об'єктивних показників, що й на етапі

вхідного тестування. Це дозволяє порівняти вхідні та вихідні дані фізичних навичок учасника та визначити різницю, тобто покращення або погіршення результатів.

Якщо після виконання індивідуального плану тренувань спостерігається статистично значуще покращення відповідних фізичних навичок для даного учасника, це свідчитиме про правильну роботу методу. Покращення показників учасника є підтвердженням ефективності застосованого алгоритму для формування його персоналізованої програми тренувань.

2.8 Висновки до розділу 2

Розроблено метод планування розвитку фізичних навичок учасників велоклубу з використанням генетичного алгоритму. Представлено загальну структуру методу, яка включає етапи збору вхідних даних про учасників, генерацію початкової популяції планів тренувань, оцінку їх придатності за визначеними критеріями, застосування генетичних операторів для створення нових, потенційно кращих планів, та вибір оптимального рішення.

Описано процес адаптації класичного генетичного алгоритму до задачі планування розвитку фізичних навичок, визначення відповідних генетичних операторів, функцій придатності та параметрів алгоритму. Наведено приклади імплементації основних компонентів генетичного алгоритму в програмному коді.

Спроектовано архітектуру методу, що включає модулі генерації тренувальних планів, оцінки їх придатності, взаємодії з базою даних, а також користувацький інтерфейс веб-системи. Описано інформаційну структуру програмної реалізації методу, моделі даних для зберігання інформації про учасників, їх фізичні навички та згенеровані плани тренувань.

Розглянуто підготовку вхідних даних для системи, а саме збір та структурування інформації про учасників, їх індивідуальні особливості та цілі розвитку фізичних навичок.

Визначено критерії оцінювання покращення зручності планування розвитку фізичних навичок учасників велоклубу за рахунок використання розробленого методу. Ключовими критеріями є реальне покращення показників фізичних навичок учасників після виконання автоматично сформованих планів тренувань, а також здатність методу генерувати ефективні та збалансовані плани з урахуванням індивідуальних особливостей.

Розділ 3 Програмна реалізація методу планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб-системи велоклубу

3.1 Визначення шляхів дослідження та засобів програмної реалізації методу

Під час роботи над методом покращення планування розвитку фізичних навичок учасників велоклубу за допомогою генетичного алгоритму визначено підходи дослідження та розробки програмного забезпечення. Для успішної розробки веб-системи вибрано відповідну методологію дослідження, яка дозволить ретельно проаналізувати функціональність та ефективність автоматизованого планування розвитку фізичних навичок. Основний акцент зроблено на використанні генетичного алгоритму.

Щодо вибору програмної системи та її функціоналу, основним критерієм є забезпечення необхідного рівня функціональності та зручності використання для учасників велоклубу. Для цього реалізовано веб-застосунок, в якому основні функції включають реєстрацію учасника, введення його персональних даних та цілей розвитку фізичних навичок, формування автоматичного плану тренувань за допомогою генетичного алгоритму з урахуванням індивідуальних особливостей учасника, а також можливість коригування та оновлення плану. Такий вибір технологій дозволив забезпечити необхідний рівень продуктивності та зручність використання веб-системи.

Щодо стратегії тестування та перевірки функцій програмного забезпечення використані ручні методи тестування. Ручні тести проводилися для оцінки інтерфейсу користувача веб-системи та загальної зручності використання програми учасниками велоклубу.

У контексті дослідження ефективності автоматизованого планування розвитку фізичних навичок вимірювалися показники покращення цих навичок учасників на основі виконання автоматично сформованих планів тренувань. Для цього були використані відповідні об'єктивні метрики, такі як час витривалості,

кількість повторень силових вправ тощо. Це дозволило об'єктивно оцінити результати дослідження та внести необхідні корективи для подальшого вдосконалення методу автоматичного планування тренувань.

3.2 Вибір засобів розробки програмної реалізації методу

Вибір відповідних засобів розробки є надзвичайно важливим етапом у процесі створення веб системи для планування розвитку фізичних навичок учасників велоклубу. Правильний підбір технологій, інструментів та середовищ розробки може значно вплинути на ефективність, масштабованість, підтримку та загальну якість кінцевого програмного забезпечення.

Для вирішення задачі розробки веб-системи велоклубу було обрано наступні інструменти: ASP.Net Web Forms як платформу, Microsoft Visual Studio як середовище програмування, C# як мову програмування, SQL Server як систему керування базами даних (СКБД), T-SQL як мову запитів, а також IIS як веб-сервер. Кожен з цих компонентів був обраний з огляду на їх специфічні переваги та можливості, що відповідають потребам проекту.

ASP.Net Web Forms було обрано для створення інтерактивних і динамічних веб-додатків завдяки його можливості швидкої розробки з використанням подійно-орієнтованої моделі програмування. Web Forms забезпечує багатий набір вбудованих серверних контролів, які дозволяють легко створювати складні веб-інтерфейси без необхідності глибокого знання HTML, CSS та JavaScript. Крім того, ця платформа підтримує повну інтеграцію з .NET Framework, що спрощує розробку і тестування додатків.

Microsoft Visual Studio обрано як середовище розробки завдяки його потужним інструментам для кодування, налагодження та тестування. Цей інтегрований середовище розробки надає розробникам можливість використовувати розширені функції, такі як інтегрований контроль версій, підтримка різних мов програмування та потужний редактор коду, що значно прискорює процес розробки та полегшує підтримку коду.

C# обрано як основну мову програмування завдяки його тісній інтеграції з платформою .NET, підтримці об'єктно-орієнтованого програмування та широкій екосистемі бібліотек і фреймворків. C# також відомий своєю простотою у використанні та високою продуктивністю, що робить його ідеальним вибором для розробки веб-додатків.

SQL Server був обраний як СКБД для цього проекту через його надійність, масштабованість та високу продуктивність. SQL Server забезпечує ефективне управління великими обсягами даних, підтримує складні запити та аналітику, а також пропонує інтегровані засоби безпеки та резервного копіювання, що є критично важливим для захисту даних користувачів велоклубу.

Для роботи з базами даних використовується T-SQL, мова запитів, яка є розширенням SQL і забезпечує додаткові можливості для управління транзакціями, створення складних запитів і процедур. T-SQL дозволяє оптимізувати роботу з даними та забезпечити високу продуктивність додатка.

IIS був обраний як веб-сервер для розгортання веб-додатку завдяки його високій продуктивності, безпеці та можливості гнучкого налаштування. IIS легко інтегрується з ASP.Net додатками, забезпечуючи стабільне та швидке виконання веб-сервісів та додатків.

Комбінація ASP.Net Web Forms, Microsoft Visual Studio, C#, SQL Server, T-SQL та IIS забезпечує потужну та ефективну платформу для розробки веб-системи велоклубу. Цей набір інструментів дозволяє швидко і якісно реалізувати всі необхідні функції, забезпечуючи високу продуктивність, безпеку та масштабованість системи.

3.3 Функціональне призначення та структура програмної реалізації методу

Для кращого розуміння роботи системи розроблено основні модулі, з яких вона складається. Розглянуто функції, які виконують ці модулі, їх

взаємозв'язок та передачу даних між ними. Представлена структура програми з називанням конкретних модулів та їх складових елементів. Наведені діаграми модулів та відповідних класів програмного продукту, а також детальний опис кожного з них.

Створено діаграму класів на основі структури програми. Ця діаграма була використана для розробки веб системи. Діаграму зображено на рисунку 3.1.

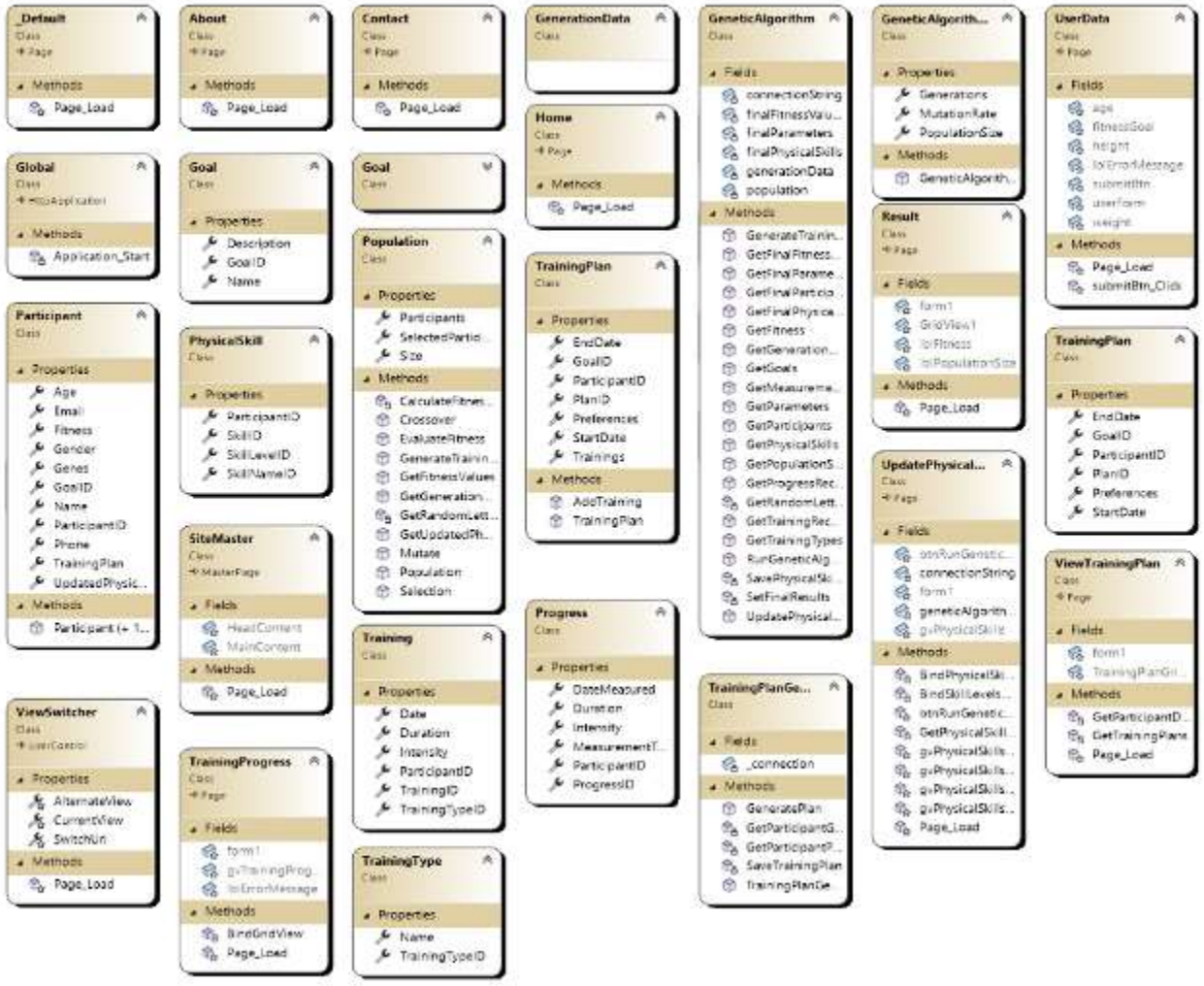


Рисунок 3.1 – Діаграма класів

GeneticAlgorithm. Головний клас, що реалізує генетичний алгоритм. Містить методи для підключення до бази даних, отримання даних, виконання генетичного алгоритму, збереження результатів тощо.

Participant. Клас, що представляє учасника програми. Містить властивості для зберігання інформації про учасника, такі як ім'я, вік, стать, електронна

пошта, номер телефону та фітнес-рівень. Також містить властивість Genes для зберігання генів учасника.

TrainingPlan. Клас, що представляє план тренувань для учасника. Містить властивості для зберігання інформації про план, такі як ідентифікатор, ідентифікатор учасника, дати початку та закінчення, ідентифікатор цілі, переваги та список тренувань.

PhysicalSkill. Клас, що представляє фізичні навички учасника. Містить властивості для зберігання ідентифікаторів навички, назви навички, рівня навички та ідентифікатора учасника.

Progress. Клас, що представляє запис про прогрес учасника. Містить властивості для зберігання ідентифікатора запису, дати вимірювання, тривалості, інтенсивності, ідентифікатора учасника та ідентифікатора типу вимірювання.

Goal. Клас, що представляє ціль для учасника. Містить властивості для зберігання ідентифікатора цілі, назви та опису.

Training. Клас, що представляє запис про тренування учасника. Містить властивості для зберігання ідентифікатора тренування, дати, тривалості, інтенсивності, ідентифікатора учасника та ідентифікатора типу тренування.

TrainingType. Клас, що представляє тип тренування. Містить властивості для зберігання ідентифікатора та назви типу тренування.

MeasurementType. Клас, що представляє тип вимірювання. Містить властивості для зберігання ідентифікатора та назви типу вимірювання.

GeneticAlgorithmParameter. Клас, що представляє параметри генетичного алгоритму. Містить властивості для зберігання розміру популяції, швидкості мутації та кількості поколінь.

Population є частиною реалізації генетичного алгоритму. Він містить методи для ініціалізації початкової популяції, обчислення придатності учасників, відбору, кросинговеру та мутації, а також для отримання оновлених даних після застосування генетичного алгоритму.

TrainingPlan в просторі імен BikeClub.Data. Клас TrainingPlan представляє план тренувань для учасника. Використовується для зберігання та

маніпулювання даними про план тренувань

TrainingPlanGenerator в просторі імен BikeClub.Data. Цей клас використовується для генерації планів тренувань для учасників на основі їхніх цілей та переваг. Він взаємодіє з базою даних для отримання необхідних даних та збереження згенерованих планів тренувань.

TrainingPlan (успадковується від System.Web.UI.Page). Частковий клас, що означає, що він може містити додатковий код.

UserData (успадковується від System.Web.UI.Page). Частковий клас, що означає, що він може містити додатковий код.

ViewTrainingPlan (успадковується від System.Web.UI.Page). Частковий клас, що означає, що він може містити додатковий код.

UpdatePhysicalskill (успадковується від System.Web.UI.Page). Частковий клас, що означає, що він може містити додатковий код.

Застосунок використовує об'єктно-орієнтований підхід і складається з декількох взаємопов'язаних класів, які відповідають за різні аспекти функціональності додатку.

Надана діаграма класів є частиною веб-застосунку, який спрямований на допомогу користувачам у плануванні та відстеженні їхньої фізичної активності та прогресу.

3.4 Особливості реалізації програмних складових методу

Після детального проектування структури та функціонального призначення програмних складових веб системи для планування розвитку фізичних навичок учасників велоклубу, наступним важливим етапом є безпосередня реалізація цих компонентів. Ефективна реалізація є критично важливою для забезпечення належної продуктивності, масштабованості, безпеки та загальної якості системи.

У процесі реалізації програмних складових веб системи використано низку підходів та рішень, які забезпечули ефективність, гнучкість та

масштабованість додатку.

Обчислення фітнесу для окремого учасника на основі його фізичних навичок виконує код (додаток В). Результат цього обчислення використовується для визначення ефективності тренувальної програми та внесення необхідних коригувань.

Цей фрагмент коду реалізує метод CalculateFitnessForParticipant, який обчислює значення придатності (фітнес) для окремого учасника на основі його фізичних навичок, прогресу та інших доступних даних. Розглянуто деталі цього методу:

Метод приймає об'єкт Participant та списки PhysicalSkill, Progress, Training, TrainingType та MeasurementType як вхідні параметри.

Змінна fitness ініціалізується значенням 0.0 і буде акумулювати значення придатності на основі різних факторів.

Перший цикл foreach ітерується по списку physicalSkills і перевіряє, чи відповідає ParticipantID навички поточному учаснику.

Якщо так, то код перевіряє рівень навички (skill.SkillLevelID) і призначає відповідну вагу до змінної fitness.

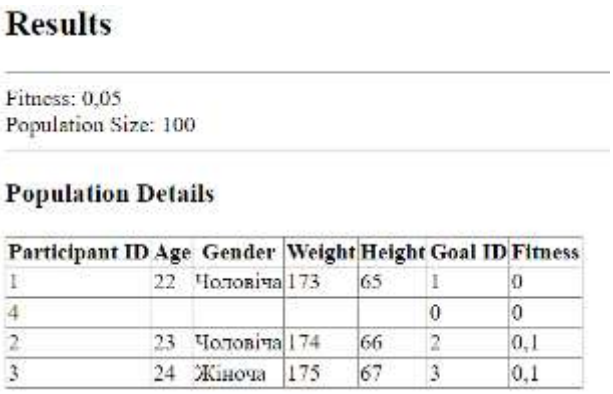
У наданому коді використовується проста схема, де початковому рівню ("Beginner") призначається вага 0.5, проміжному ("Intermediate") - 0.75, а просунутому ("Advanced") - 1.0.

Другий цикл foreach ітерується по списку progressRecords і перевіряє, чи відповідає ParticipantID запису прогресу поточному учаснику.

Якщо так, то до змінної fitness додається фіксоване значення 0.1 для кожного запису прогресу.

Після обчислення загальної оцінки придатності, метод повертає значення змінної fitness.

Роботу кода для програмної реалізації методу можна побачити на рисунку 3.2.



Results

Fitness: 0,05
Population Size: 100

Population Details

Participant ID	Age	Gender	Weight	Height	Goal ID	Fitness
1	22	Чоловіча	173	65	1	0
4					0	0
2	23	Чоловіча	174	66	2	0,1
3	24	Жіноча	175	67	3	0,1

Рисунок 3.2 – Обчислення фітнесу

Розроблено фрагмент коду який реалізує метод Crossover в класі Population (додаток В).

Метод виконує операцію кросинговеру на списку відібраних учасників (selectedParticipants) для генерації нових учасників (нащадків) для наступного покоління в рамках генетичного алгоритму.

Спочатку створюється новий список newParticipants для зберігання нових учасників, які будуть згенеровані в результаті кросинговеру.

Створюється об'єкт Random для генерації випадкових чисел.

Цикл for ітерується по списку selectedParticipants з кроком 2, щоб обробити кожну пару сусідніх учасників.

Для кожної пари учасників (батьків) вибирається випадкова точка кросинговеру (crossoverPoint) за допомогою методу random.Next(1, parent1.Genes.Length). Ця точка визначає позицію, де відбудеться обмін частинами генетичних рядків батьків.

Створюються два нових генетичних рядки (childGenes1 та childGenes2) для нащадків шляхом комбінування частин генетичних рядків батьків до та після точки кросинговеру.

За допомогою конструктора Participant(genes) створюються два нових об'єкти Participant (child1 та child2), які представляють нащадків з новими генетичними рядками.

Нові нащадки додаються до списку newParticipants.

Після завершення циклу, метод повертає список newParticipants, який

містить нових учасників, згенерованих в результаті кросинговеру.

Операція кросинговеру є ключовою складовою генетичного алгоритму, оскільки вона дозволяє комбінувати характеристики існуючих учасників (батьків) для створення нових, потенційно кращих рішень (нащадків). У цьому конкретному випадку кросинговер здійснюється шляхом обміну частинами генетичних рядків батьків, але в реальних додатках можуть використовуватися інші методи кросинговеру, залежно від специфіки задачі та способу представлення рішень.

Фрагмент коду який реалізує метод Mutate в класі Population для даної веб системи (додаток В).

Метод виконує операцію мутації на списку учасників (participants) згідно із заданою швидкістю мутації (mutationRate). Мутація є важливою частиною генетичного алгоритму, оскільки вона забезпечує введення випадкових змін у генетичні рядки учасників, що може призвести до появи нових, потенційно кращих рішень.

Спочатку створюється об'єкт Random для генерації випадкових чисел.

Цикл foreach ітерується по списку participants.

Для кожного учасника (participant) генерується випадкове число за допомогою random.NextDouble().

Якщо згенероване число менше, ніж задана швидкість мутації (mutationRate), то виконується мутація.

Для виконання мутації спочатку вибирається випадкова позиція (mutationIndex) у генетичному рядку учасника за допомогою random.Next(0, participant.Genes.Length).

Генетичний рядок учасника (participant.Genes) перетворюється на масив символів (genesArray) за допомогою ToCharArray().

Символ на вибраній позиції (mutationIndex) в масиві genesArray замінюється на новий випадковий символ, згенерований за допомогою методу GetRandomLetter().

Після заміни символу новий генетичний рядок створюється з масиву

genesArray за допомогою нового конструктора рядка string(genesArray), і присвоюється властивості participant.Genes.

Таким чином, метод Mutate вносить випадкові зміни в генетичні рядки учасників з певною ймовірністю, визначеною швидкістю мутації (mutationRate). Ця операція є важливою для запобігання збіжності генетичного алгоритму до локального оптимуму та сприяє більш ефективному пошуку кращих рішень.

Реалізація генетичного алгоритму у методі RunGeneticAlgorithm для даної веб системи (додаток В).

Цей фрагмент коду є центральною частиною реалізації генетичного алгоритму. Він виконує підключення до бази даних, отримує необхідні дані, ініціалізує початкову популяцію, а потім виконує цикл генетичного алгоритму протягом заданої кількості поколінь. У циклі відбувається оцінка придатності особин, відбір, кросинговер і мутація. Після завершення циклу встановлюються фінальні результати та зберігаються оновлені фізичні навички в базі даних.

Збереження оновлених фізичних навичок у базі даних (метод SavePhysicalSkills) (додаток В).

Цей фрагмент коду демонструє, як зберігати оновлені дані у базі даних після завершення роботи генетичного алгоритму. Він приймає список оновлених фізичних навичок (PhysicalSkill) і виконує SQL-запит UPDATE для кожної навички. Параметри запиту встановлюються за допомогою SqlCommand.Parameters.AddWithValue, а сам запит виконується методом ExecuteNonQuery.

Генерація планів тренувань для учасників (додаток В).

Цей фрагмент коду відповідає за генерацію планів тренувань для учасників після завершення роботи генетичного алгоритму. Він отримує оновлені дані про учасників з бази даних, а потім для кожного учасника створює екземпляр класу TrainingPlanGenerator і викликає метод GeneratePlan, передаючи йому об'єкт Participant.

Функція GenerateTrainingPlan є критично важливим елементом системи, що відповідає за автоматичне створення індивідуальних планів тренувань. Вона

представляє собою процес, який починається з отримання персональних даних учасника і завершується генерацією та збереженням унікального плану тренувань у базі даних.

Центральна ідея цієї функції полягає в тому, що ефективний план тренувань повинен бути побудований на основі всебічного розуміння фізичного стану та цілей людини. У світі, де більшість фітнес-планів є загальними шаблонами, система виділяється тим, що створює план, адаптований до кожного унікального профілю учасника.

Функція починається з запиту до бази даних для отримання повної інформації про учасника за його ідентифікатором користувача.

Витягнута інформація охоплює ключові фізичні характеристики: стать, вагу, зріст і вік. Ці дані не просто цифри - вони є фундаментальними факторами, що визначають метаболізм, силу, витривалість та схильність до травм. Наприклад, чоловіки і жінки мають різний рівень тестостерону, що впливає на набір м'язової маси. Вік впливає на час відновлення та щільність кісток. Зріст і вага разом визначають індекс маси тіла (ІМТ), який може вказувати на ризики для здоров'я.

Додатково є включення GoalID. Мета не просто впливає на поради, а стає основою для всього плану тренувань. Це відображає прогресивний підхід у сучасному фітнесі.

Після збору даних, функція делегує створення плану класу TrainingPlanGenerator. Замість того, щоб намагатися робити все самостійно, GenerateTrainingPlan фокусується на координації процесу, передаючи спеціалізовані завдання іншим класам.

Після створення плану, функція робить ще один важливий крок - збереження його в базі даних. План має чітко визначені дати початку та завершення (місячний цикл).

Функція GenerateTrainingPlan є важливою для даної системи, оскільки вона дає кожному користувачеві власні персональні плани тренувань на основі генетичного алгоритму, що значно спрощує тренування.

Створення рекомендацій для учасника на основі згенеровано плану тренувань виконує код (додаток В).

Функція GenerateRecommendation є ключовим компонентом системи персоналізованих фітнес-рекомендацій. Вона демонструє застосування принципів адаптивного навчання та персоналізації в контексті спортивних тренувань. Ця функція не просто видає загальні поради, а створює індивідуальний план, враховуючи унікальний профіль кожного учасника.

Основний принцип роботи функції - це аналіз розривів у навичках. Вона розділяє фізичні якості на ті, що потребують вдосконалення (оцінка 2 або нижче), та ті, що вже добре розвинені (оцінка 4 або вище). Це дозволяє визначити області, де учасник може найефективніше покращити свої результати. Наприклад, якщо витривалість оцінюється низько, система пропонує цілеспрямовано працювати над цією якістю, оскільки її покращення може дати значний приріст у загальній фізичній підготовці.

Водночас функція визнає і відзначає досягнення учасника. Якщо, скажімо, швидкість отримала високу оцінку, система не лише хвалить за прогрес, але й стратегічно спрямовує увагу на інші області, такі як силові тренування. Це забезпечує збалансований розвиток і запобігає "плато" в тренуваннях.

Ще одним важливим аспектом є інтеграція особистих цілей учасника. Функція взаємодіє з базою даних, щоб отримати інформацію про індивідуальну мету - будь то схуднення, нарощування м'язів чи підвищення витривалості. Ця інформація використовується для точного налаштування рекомендацій. Наприклад:

Для тих, хто прагне схуднути, акцент робиться на збільшенні кардіонавантажень, які ефективно спалюють калорії.

Учасникам, які хочуть наростити м'язову масу, рекомендують інтенсивні силові тренування з важкими вагами і більшою кількістю підходів, що стимулює гіпертрофію м'язів.

Для покращення витривалості пропонуються довгі велосипедні прогулянки на помірній інтенсивності, які розвивають аеробні можливості без надмірного навантаження на суглоби.

Важливо відзначити гнучкість та масштабованість цієї системи. Хоча зараз вона містить три основні цілі, код легко розширити для включення додаткових завдань. Це може бути підготовка до марафону, реабілітація після травми чи тренування для конкретного виду спорту. Кожна нова мета може бути додана з відповідними, науково обґрунтованими рекомендаціями.

GenerateRecommendation демонструє принципи модульності та роздільності обов'язків. Вона отримує дані про навички та учасника як параметри, а не витягує їх напряму, що полегшує тестування та інтеграцію. Крім того, функція делегує отримання цілі окремому методу GetParticipantGoal, що дозволяє змінювати джерело даних без впливу на основну логіку.

Загалом, ця функція є прикладом того, як сучасні технології можуть перетворити загальні фітнес-поради на персоналізований, науково обґрунтований план тренувань. Вона демонструє, як аналіз даних, адаптивні алгоритми та розуміння людських цілей можуть об'єднатися для створення більш ефективних і мотиваційних фітнес-рекомендацій.

Усі ці фрагменти коду демонструють різні аспекти реалізації генетичного алгоритму, роботи з базою даних, генерації планів тренувань, налаштування параметрів, отримання та збереження результатів, а також моделі даних. Без них робота веб-застосунку була б неможливою. Наведені діаграми модулів та класів програмного продукту ілюструють структуру системи та взаємозв'язки між її компонентами.

3.5 Тестування програмної реалізації методу та вимоги до розгортання

Тестування є важливим етапом у процесі розробки програмного забезпечення, що дозволяє виявити та усунути помилки, перевірити

відповідність функціональним вимогам та забезпечити належну якість продукту. Під час розробки даної інформаційної системи було проведено різні види тестування.

Ретельне тестування веб застосунку за допомогою добре продуманих тест-кейсів може значно підвищити надійність та зручність використання програмного продукту.

Перший тест кейс показує чи зберігається сесія користувача після входу на панель керування персональним аккаунтом (таблиця 3.1).

Таблиця 3.1 – Тест-кейс TS0001

<table>
<tr><td>Тест-кейс ID: TS0001</td><td>Пріоритет: 1</td><td>Створено:24.05.2024, Ґоляж М.М.</td></tr>
<tr><td colspan="3">Назва: Перевірка на збереження сесії користувача</td></tr>
<tr><td colspan="2">Кроки</td><td>Очікуваний результат</td></tr>
<tr><td colspan="2">1. Натиснути на панель користувача на головній сторінку
2. Ввести логін і пароль від аккаунта
3. Перевіряємо чи зберігається сессія аккаунта</td><td>Після натискання кнопки перекидає на сторінку де потрібно ввести свої дані від аккаунта
Переходимо на панель користувача під власним аккаунтом
На сторінці буде відобрадатися наш нік</td></tr>
<tr><td colspan="3">Результат виконання тест-кейсу: пройдено успішно</td></tr>
</table>

Натискаємо кнопку «Панель користувача» на головній сторінці веб застосунку зображено на рисунку 3.3.

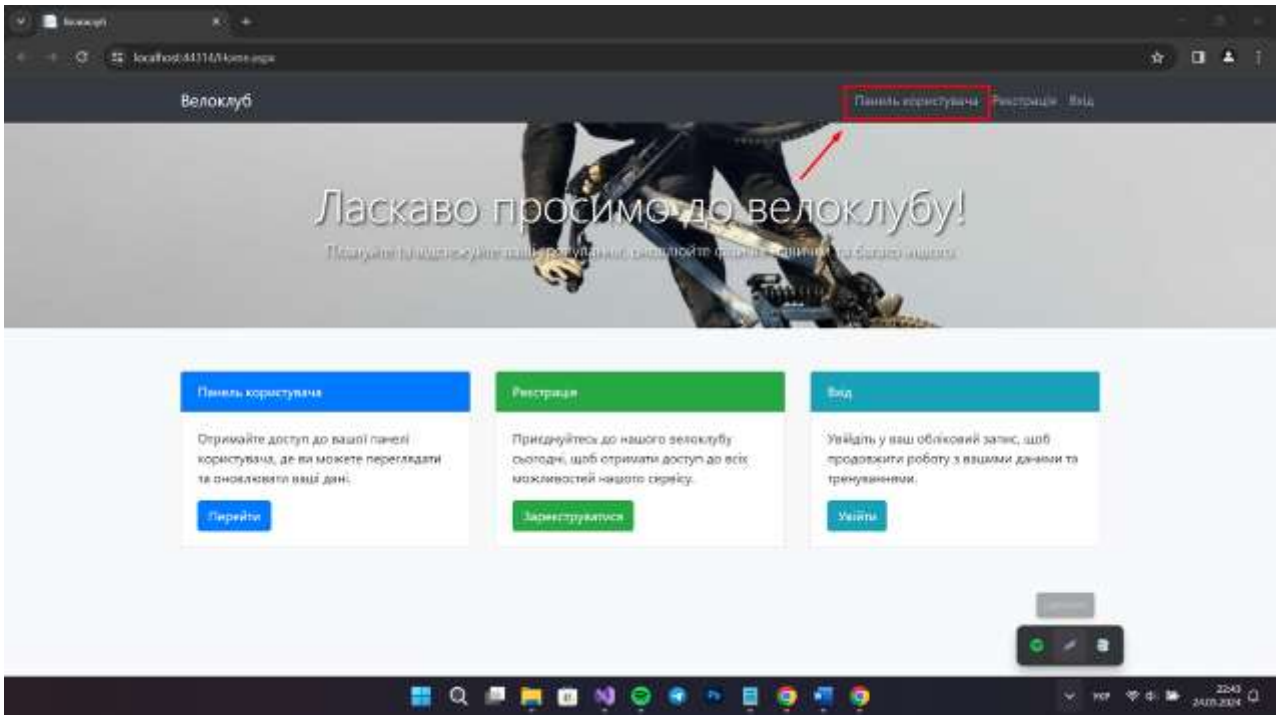


Рисунок 3.3 – Головна сторінка веб застосунка

Далі нас перекидає на сторінку авторизації, де вводимо наші дані від аккаунту, а саме логін і пароль, та натискаємо «Увійти». Виконано на рисунку 3.4.

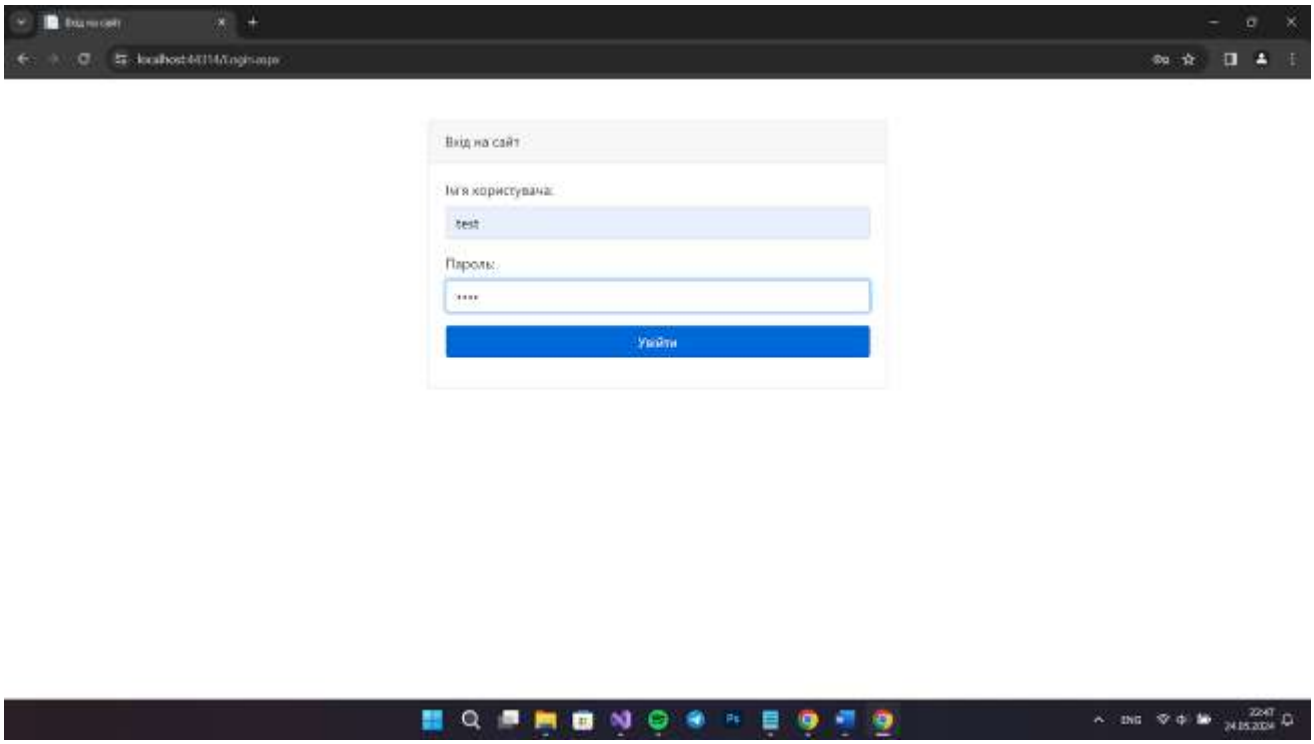


Рисунок 3.4 – Сторінка авторизації

Після цього бачимо на наступній сторінці, що наша сессія збережена, і ми можемо продовжувати роботу на цьому аккаунті. Зображено на рисунку 3.5.

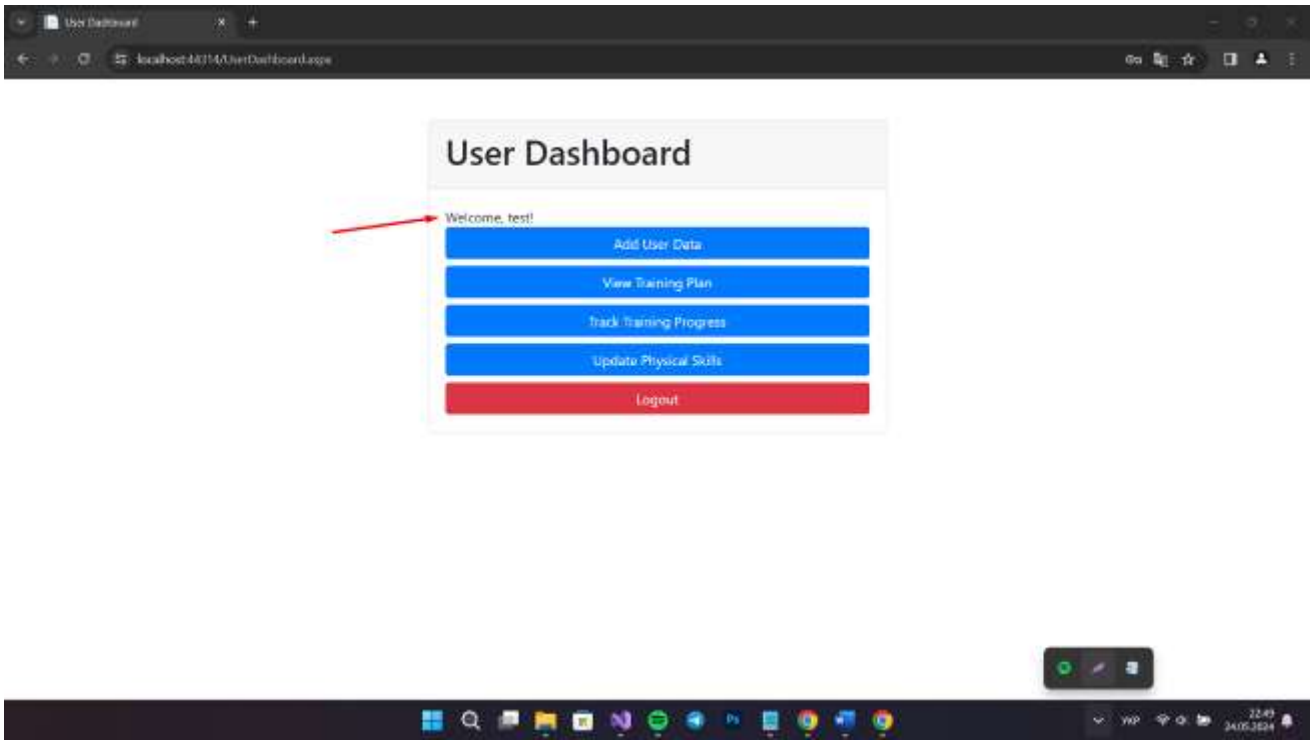


Рисунок 3.5 – Панель користувача

Як показано на вище наведеному рисунку, сесія користувача успішно збережена і можна користуватися системою. Отже, тестування було успішно пройдено.

Успішне проходження тестів, зокрема збереження сесії користувача, свідчить про коректну роботу реалізованих функцій аутентифікації та авторизації в системі. Це дозволяє користувачам безпечно входити в обліковий запис, зберігати свої персональні дані та мати доступ до індивідуальних тренувальних планів.

Однією з ключових функцій розробленої веб-системи є можливість зберігати та отримувати дані користувачів та їхні персоналізовані плани тренувань з бази даних.

Наступний тест показує чи зберігаються наші дані у базі даних (таблиця 3.2).

Таблиця 3.2 – Тест-кейс TS0002

Тест-кейс ID: TS0002	**Пріоритет:** 1	**Створено:**24.05.2024, Ґоляж М.М.
Назва: Перевірка на збереження даних у базі даних		
Кроки		**Очікуваний результат**
1. У панелі користувача перейти на сторінку «Додати користувацькі дані» 2. Ввести потрібні дані		Після натискання кнопки «Додати користувацькі дані» перекидає на сторінку де потрібно ввести потрібні дані Вводимо і зберігаємо
Результат виконання тест-кейсу: пройдено успішно		

У панелі користувача переходимо на сторінку добавлення даних для аккаунта. Там заповнюємо усі дані, та обираємо мету. Продемонстровано на рисунку 3.6.

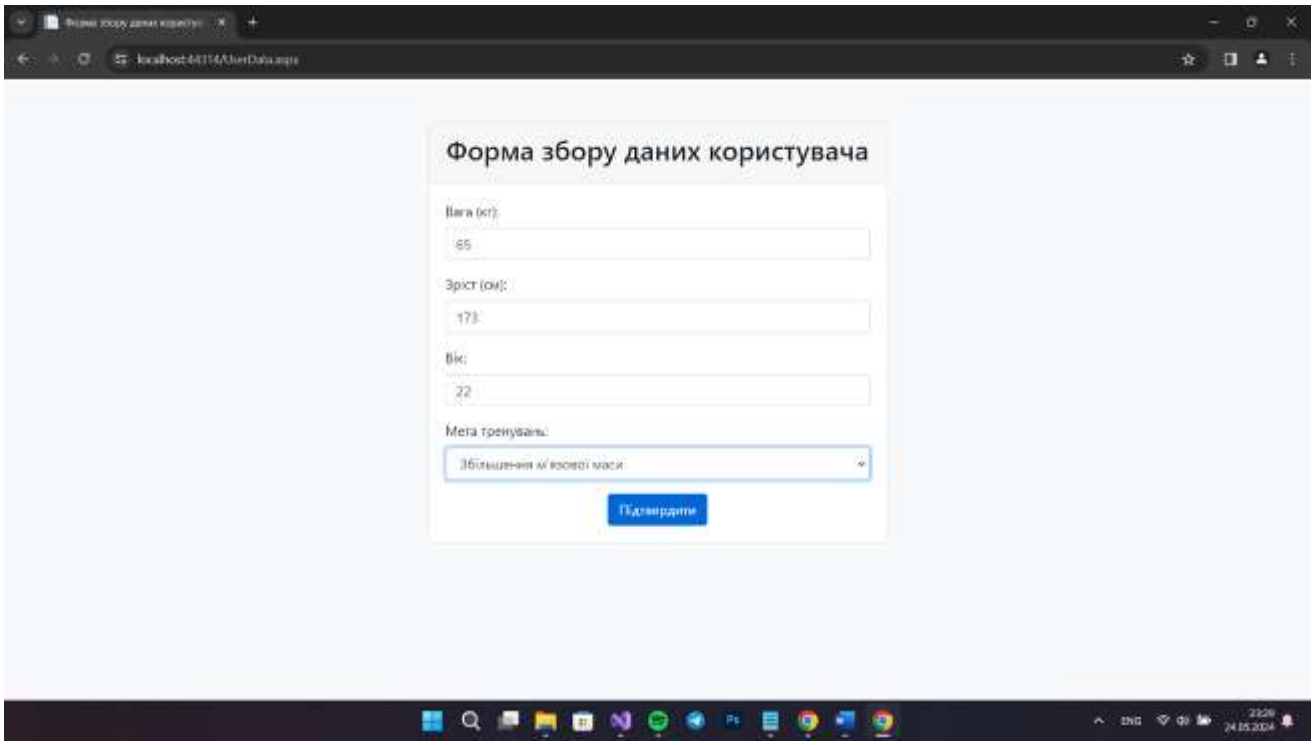
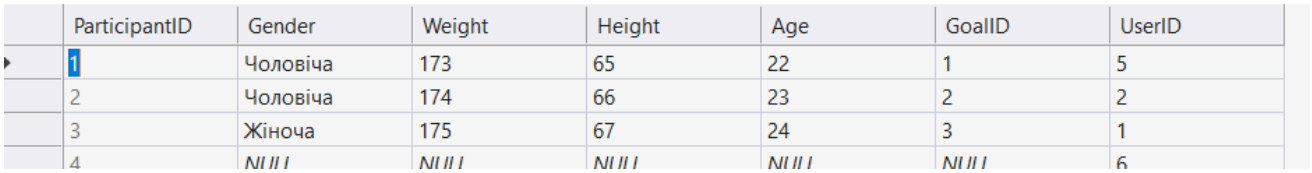


Рисунок 3.6 – Форма заповнення даних

Далі потрібно перевірити чи у базі даних чи наші дані збереглися. Показано на рисунку 3.7.



ParticipantID	Gender	Weight	Height	Age	GoalID	UserID
1	Чоловіча	173	65	22	1	5
2	Чоловіча	174	66	23	2	2
3	Жіноча	175	67	24	3	1
4	NULL	NULL	NULL	NULL	NULL	6

Рисунок 3.7 – Дані в базі даних

Як видно на вище наведеному рисунку, дані були успішно збережені в таблиці. Отже, тест успішно пройдено.

Далі потрібно протестувати як працює генетичний алгоритм у системі чи не має він збоїв чи інших проблем (таблиця 3.3).

Таблиця 3.3 – Тест-кейс TS0003

Тест-кейс ID: TS0004	**Пріоритет:** 1	**Створено:**26.05.2024, Ґоляж М.М.
Назва: Перевірка роботи генетичного алгоритму		
Кроки		**Очікуваний результат**
1. Запускаємо окрему сторінку на сервері Result.aspx 2. У базі даних в таблиці GeneticAlgorithmPatametr тестуємо рідні значення для алгоритму 3. Перевіряємо роботу на нашій сторінці		Після налаштування усіх параметрів у базі даних перевіряємо результати на сторінці Result.aspx та дивимося фітнес, кількість популяцій
Результат виконання тест-кейсу: пройдено успішно		

Спочатку потрібно налаштувати параметри для генетично алгоритму у базі даних, щ обуло зображено на рисунку 3.8.

	ParameterID	PopulationSize	MutationRate	Generations
▶	1	100	0,01	50
*	NULL	NULL	NULL	NULL

Рисунок 3.8 – Налаштування параметрів

Запускаємо сторінку Result.aspx на стороні сервера. Та переглядаємо на рисунку 3.9.

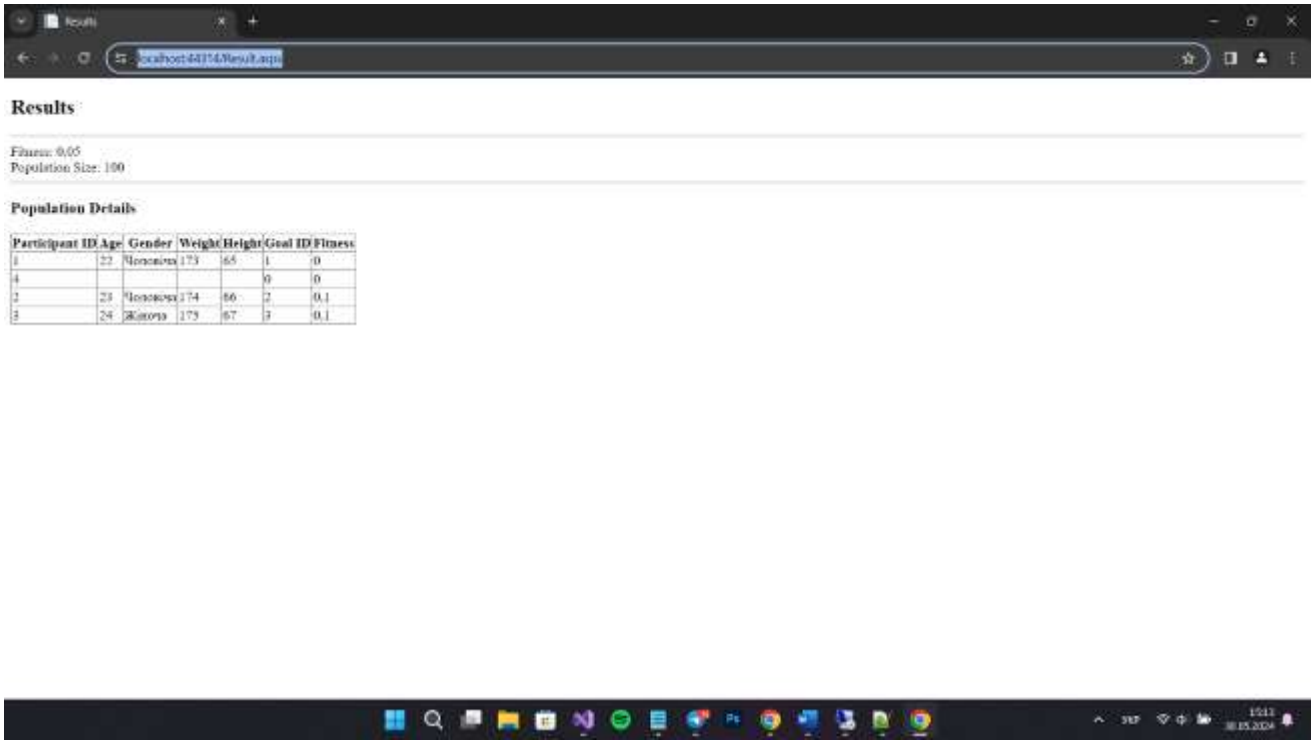


Рисунок 3.9 – Перегляд роботи алгоритму

Додатково можна глянути у консолі, чи немає ніяких помилок при роботі, переглядаэмо на рисунку 3.10. Перевірка консолі на наявність помилок є важливим кроком у процесі тестування та налагодження програмного забезпечення. Консоль надає цінну інформацію про стан виконання програми, виявлені проблеми та можливі причини їх виникнення.

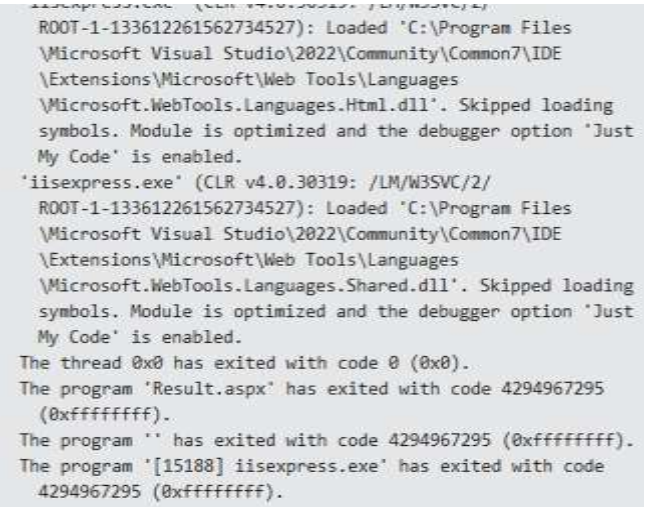


Рисунок – 3.10 – Перевірка помилок у консолі

На сторінці Result.aspx показано дані виконання генетичного алгоритму, такі як найкраще знайдене рішення, кількість ітерацій, що знадобилися для його знаходження, а також графік зміни значення цільової функції протягом роботи алгоритму. Ці дані дозволяють оцінити ефективність роботи генетичного алгоритму та проаналізувати його поведінку.

Відсутність помилок у консолі свідчить про коректну роботу програмного коду та відсутність критичних збоїв під час виконання. Це є важливим показником стабільності та надійності реалізованого алгоритму.

Таким чином, наявність очікуваних результатів на сторінці Result.aspx та відсутність помилок у консолі дозволяють зробити висновок, що тестування генетичного алгоритму пройдено успішно, і він працює коректно з наданими вхідними параметрами. Цей успішний результат вказує на те, що алгоритм був правильно реалізований та налаштований для вирішення поставленої задачі. Проте, важливо регулярно проводити додаткові тести з різноманітними вхідними даними, щоб переконатися в стабільності та надійності роботи алгоритму у різних сценаріях.

Потрібно протестувати створення індивідуальних планів тренувань за допомогою генетичного алгоритму, адже це один із основних функціоналів системи (таблиця 3.4).

Таблиця 3.4 – Тест-кейс TS0004

Тест-кейс ID: TS0005	**Пріоритет:** 1	**Створено:**26.05.2024, Ґоляж М.М.
Назва: Перевірка створення індивідуальних планів тренувань		
Кроки	**Очікуваний результат**	
1. Запускаємо сторінку створення планів тренувань 2. Перевіряємо роботу на нашій сторінці	Перевіряємо як згенерувало тренування і які рекомендації нам відобразило	
Результат виконання тест-кейсу: пройдено успішно		

Входимо у власний кабінет та переходимо на сторінку перегляду індивідуальних планів тренувань за допомогою генетичного алгоритму відповідно до наших даних, що зображені на рисунку 3.11.

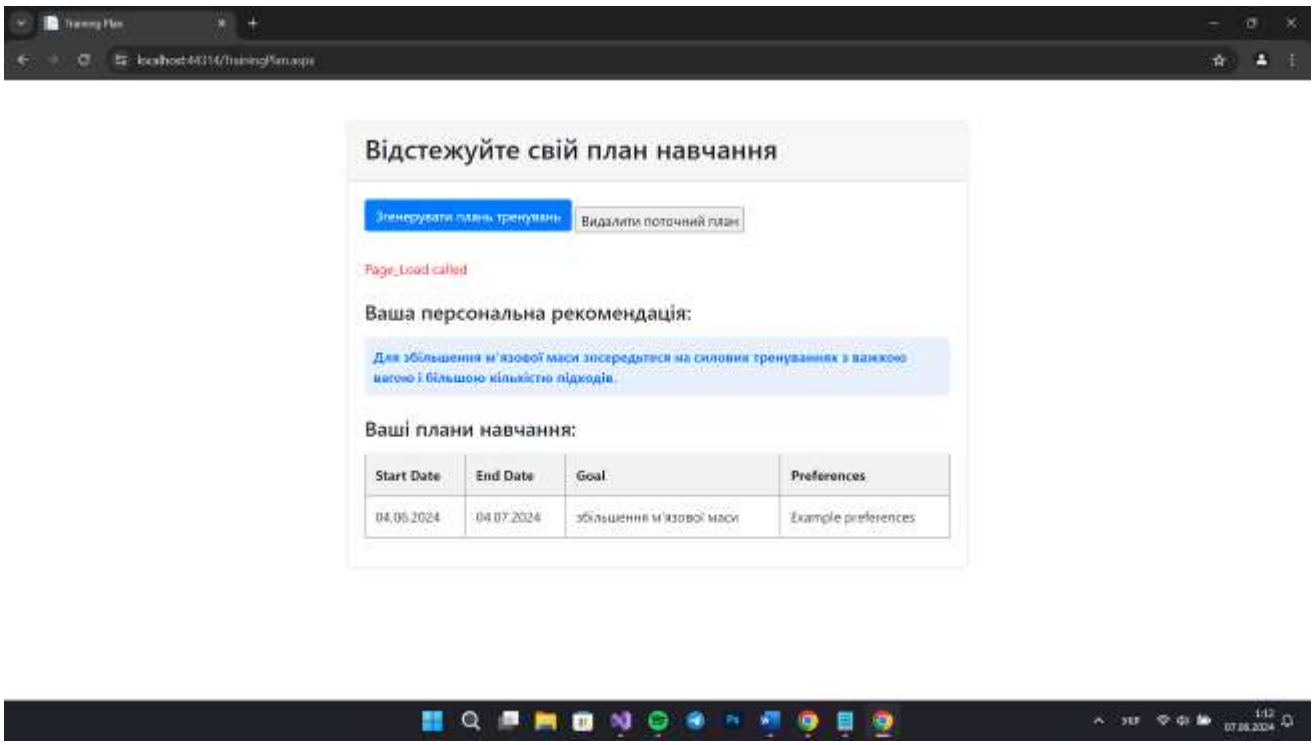


Рисунок 3.11 – Перегляд індивідуального плану тренувань

Після генерації плану з’явилося вікно з персональними рекомендаціями, де вказано на що можна більше зосередити увагу. Додатково можна видалити план тренувань, як показано на рисунку 3.12.

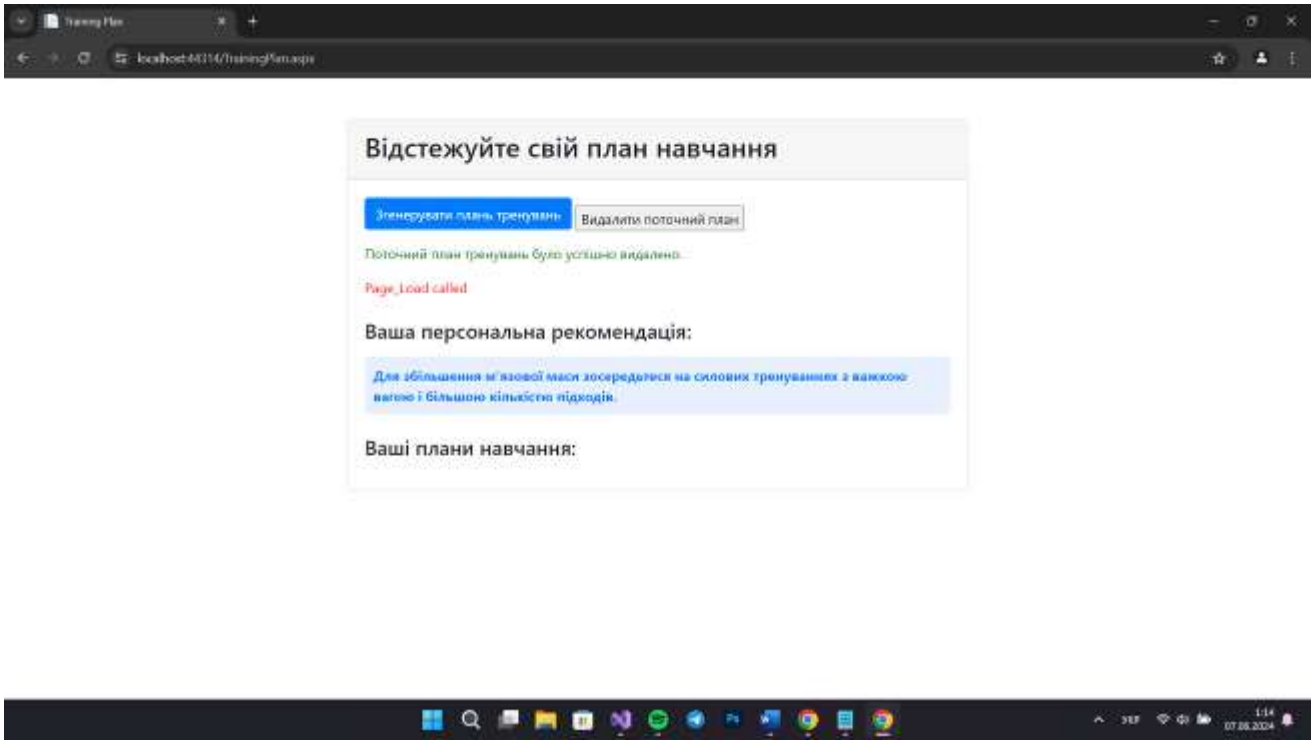


Рисунок 3.12 – Видалення плану тренувань

Як видно на зображенях вище, індивідуальні плани тренувань відповідно до наших даних успішно генеруються за допомогою генетичного алгоритму.

Належне тестування та дотримання вимог до розгортання є критично важливими для забезпечення якості, стабільності та безперебійної роботи інформаційної системи для планування розвитку фізичних навичок учасників велоклубу в продуктивному середовищі.

3.6 Оцінювання покращення зручності планування розвитку фізичних навичок учасників велоклубу

Проведення тестування для оцінки правильності роботи методу планування тренувань на основі генетичного алгоритму для учасників велоклубу є важливим. Воно забезпечує об'єктивну, кількісну та персоналізовану оцінку

ефективності методу планування тренувань. Для проведення тестування необхідно використати реальні дані учасників велоклубу з їхніми індивідуальними цілями та початковими фізичними показниками.

Олександр, 25 років, учасник велоклубу. Його метою є покращення витривалості та сили. Для початку обираємо потрібний нам навик та вводимо початкові фізичні дані Олександра, зображено на рисунку 3.11. Після цього генетичний алгоритм повинен згенерувати персоналізований план тренувань для Олександра, враховуючи його вік, поточний рівень фізичної підготовки та цілі щодо підвищення витривалості та сили.

Виконання запропонованого плану тренувань Олександром протягом певного періоду та порівняння його результатів з початковими показниками дозволить оцінити ефективність методу планування тренувань на основі генетичного алгоритму. Аналогічні тести слід провести для інших учасників велоклубу з різними цілями та початковими даними для комплексної перевірки роботи методу.

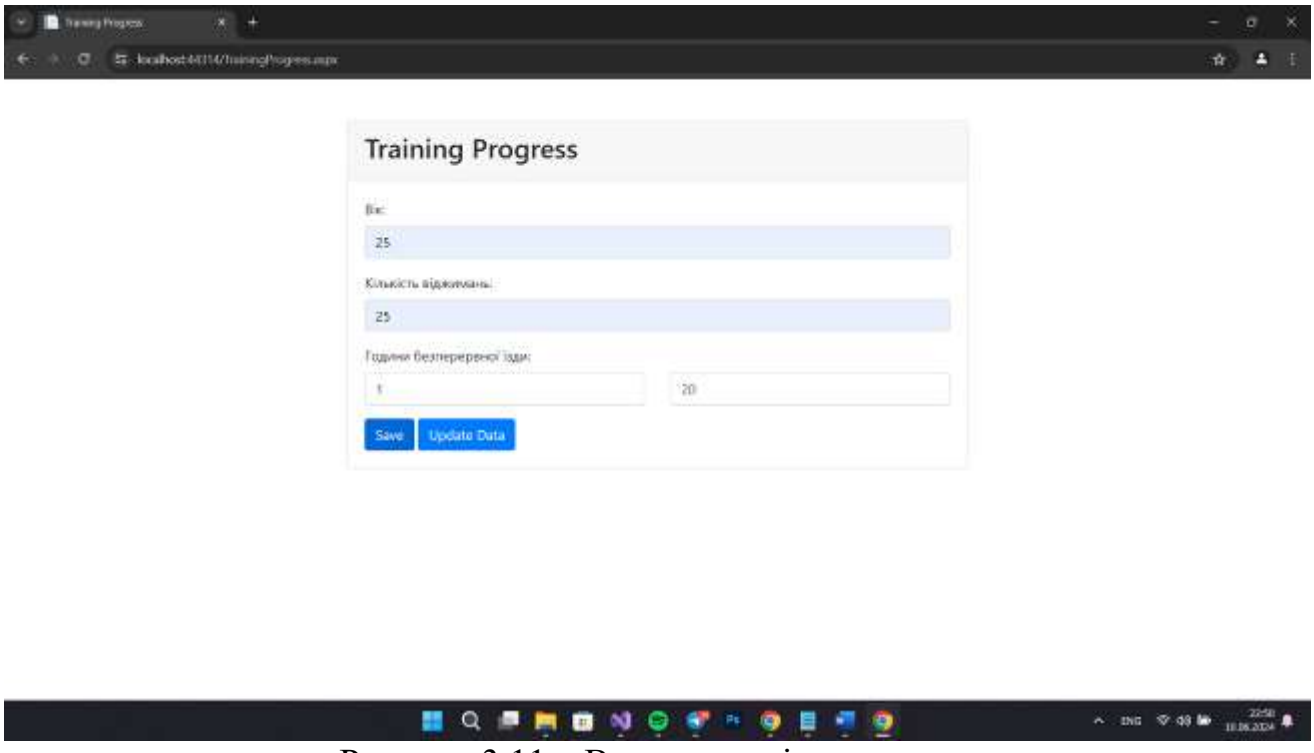


Рисунок 3.11 – Введення вхідних даних

На початку процесу було чітко визначено дві фізичні навички, які необхідно розвинути за допомогою тренувань - витривалість та силу. Витривалість вимірювалась часом безперервної їзди на велосипеді, а сила - кількістю віджимань, які може виконати людина. Проведене вхідне тестування:

– витривалість: 1 година 20 хвилин безперервної їзди;

– сила: 25 віджимань.

Сформований індивідуальний план тренувань з використанням методу на основі генетичного алгоритму.

Згідно з розробленим планом, Олександр ретельно виконував всі запропоновані тренування протягом 3 місяців. Програма тренувань була ретельно збалансована та містила вправи на розвиток як витривалості, так і сили. Проведене вихідне тестування після виконання плану:

– витривалість: 2 години безперервної їзди;

– сила: 35 віджимань.

Записуємо це у наші результати, як показано на рисунку 3.12.

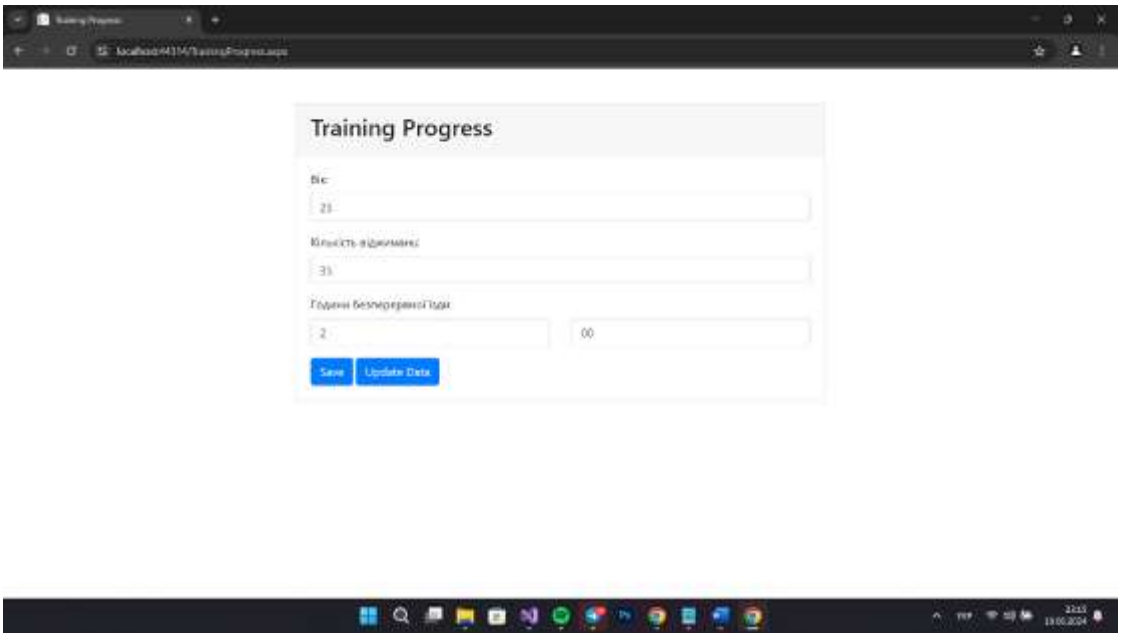


Рисунок 3.12– Введення вихідних даних

Порівняння вхідних та вихідних показників:

– витривалість покращилася на 40 хвилин (31% приросту);

– сила покращилася на 10 віджимань (40% приросту).

Спостерігається статистично значуще покращення обох фізичних навичок для Олександра, що свідчить про правильну роботу методу планування в його випадку, продемонстровано на рисунку 3.13.

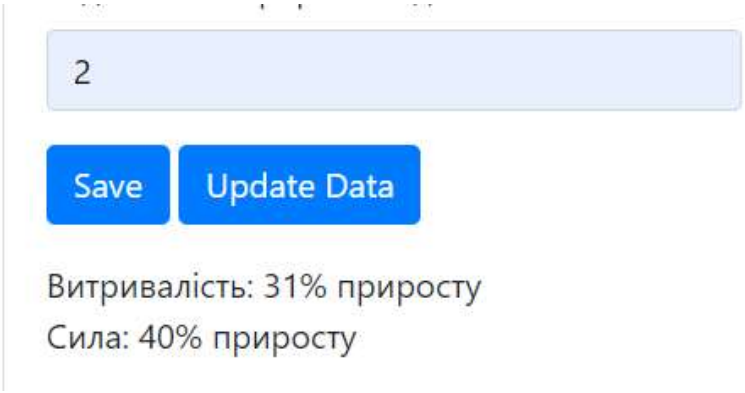


Рисунок 3.13 – Покращення результатів

Такий детальний опис кожного кроку дозволяє краще зрозуміти весь процес тестування методу планування тренувань та оцінки його результативності для конкретного учасника велоклубу.

Результати представлені у таблиці 3.5.

Таблиця 3.5 – Покращення фізичних навичок

Фізична навичка	Вхідний показник	Вихідний показник	Покращення
Витривалість	1 год 20 хв	2 год	40 хв (31%)
Сила	25 віджимання	35 віджимання	10 (40%)

Для учасника велоклубу Олександра, 25 років, метою якого було покращення витривалості та сили, було сформовано індивідуальний план тренувань з урахуванням його початкових показників та цілей. Після виконання запропонованого плану протягом 3 місяців було проведено повторне тестування, яке продемонстрував значне покращення обох фізичних навичок: витривалість збільшилася на 31% (з 1 години 20 хвилин до 2 годин безперервної їзди), а сила зросла на 40% (з 25 до 35 віджимань).

Цей приклад ілюструє ефективність розробленого методу та підтверджує досягнення поставленої мети - покращення зручності планування розвитку фізичних навичок учасників велоклубу. Завдяки автоматизації процесу формування плану тренувань та врахуванню індивідуальних особливостей кожного учасника, метод дозволяє отримувати персоналізовані та збалансовані програми тренувань, які забезпечують значний прогрес у розвитку потрібних фізичних навичок.

Таким чином, програмна реалізація запропонованого методу планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб-системи велоклубу є ефективним інструментом, який значно покращує зручність та результативність процесу планування тренувань для учасників.

3.7 Висновки до розділу 3

Виконао програмну реалізацію методу планування розвитку фізичних навичок учасників велоклубу з використанням платформи ASP.Net Web Forms та мови програмування C# для створення веб-додатку.

Представлено функціональне призначення та структуру програмної реалізації методу, а саме - архітектуру веб-додатку.

Особливості реалізації програмних складових методу, таких як реалізація генетичного алгоритму, обробка вхідних даних та генерація планів тренувань, були детально описані та проілюстровані відповідними фрагментами коду.

Проведено тестування програмної реалізації методу та визначено вимоги до розгортання системи в продуктивному середовищі.

Проведено оцінювання покращення зручності планування розвитку фізичних навичок учасників велоклубу за рахунок використання розробленого програмного забезпечення.

Наведено приклад його застосування для учасника велоклубу. Після виконання сформованого індивідуального плану тренувань протягом певного часу спостерігалося значне покращення його фізичних навичок. Цей приклад

ілюструє досягнення поставленої мети - покращення зручності планування розвитку фізичних навичок учасників велоклубу завдяки автоматизації процесу формування персоналізованих програм тренувань з урахуванням індивідуальних особливостей.

У рамках досягнення мети покращення зручності планування розвитку фізичних навичок учасників велоклубу за допомогою методу на основі генетичного алгоритму, було проведене тестування на прикладі учасника Олександра. Для Олександра із заданими цілями покращення витривалості та сили був сформований індивідуальний план тренувань. Після виконання запропонованого плану протягом 3 місяців спостерігалося значне покращення обох фізичних навичок із наступними чисельними результатами:

– витривалість покращилася на 40 хвилин (31% приросту) - з 1 години 20 хвилин до 2 годин безперервної їзди;

– сила покращилася на 10 віджимань (40% приросту) - з 25 до 35 віджимань.

Отримані результати демонструють ефективність розробленого методу планування тренувань на основі генетичного алгоритму для досягнення поставленої мети покращення зручності та результативності процесу планування розвитку фізичних навичок учасників велоклубу. Завдяки автоматизації процесу формування індивідуальних планів тренувань та врахуванню особистих характеристик учасників, метод забезпечує значний прогрес у розвитку необхідних фізичних навичок.

Загальні висновки

Було досліджено теоретичні підходи до автоматизованого планування розвитку фізичних навичок, зокрема застосування генетичних алгоритмів для генерації оптимальних тренувальних програм. Здійснено аналіз предметної області - специфіки тренувального процесу та потреб учасників велоклубу у індивідуальному плануванні з урахуванням їхніх особливостей та цілей. Досліджено інформаційне забезпечення предметної області, яке включає дані про учасників, фізичні навички, що потребують розвитку, та параметри планування тренувань.

На основі проведеного аналізу була сформульована мета роботи - покращення зручності планування розвитку фізичних навичок учасників велоклубу за рахунок автоматизації процесу формування плану тренувань та врахування індивідуальних особливостей учасників. Для досягнення поставленої мети було визначено основні завдання: розробка методу автоматизованого планування тренувань з використанням генетичного алгоритму, створення веб-системи для реалізації розробленого методу, забезпечення зручного інтерфейсу для введення персональних даних учасників та отримання згенерованих планів, а також вдосконалення методу на основі результатів тестування та зворотнього зв'язку від користувачів.

Розроблено метод планування розвитку фізичних навичок учасників велоклубу з використанням генетичного алгоритму. Представлено загальну структуру методу, яка включає етапи збору вхідних даних про учасників, генерацію початкової популяції планів тренувань, оцінку їх придатності за визначеними критеріями, застосування генетичних операторів для створення нових, потенційно кращих планів, та вибір оптимального рішення. Описано процес адаптації класичного генетичного алгоритму до задачі планування розвитку фізичних навичок, визначення відповідних генетичних операторів, функцій придатності та параметрів алгоритму. Наведено приклади імплементації основних компонентів генетичного алгоритму в програмному коді.

Спроектовано архітектуру методу, що включає модулі генерації тренувальних планів, оцінки їх придатності, взаємодії з базою даних, а також користувацький інтерфейс веб системи. Описано інформаційну структуру програмної реалізації методу, моделі даних для зберігання інформації про учасників, їх фізичні навички та згенеровані плани тренувань. Розглянуто підготовку вхідних даних для системи, а саме збір та структурування інформації про учасників, їх індивідуальні особливості та цілі розвитку фізичних навичок.

Визначено критерії оцінювання покращення зручності планування розвитку фізичних навичок учасників велоклубу за рахунок використання розробленого методу. Ключовими критеріями є реальне покращення показників фізичних навичок учасників після виконання автоматично сформованих планів тренувань, а також здатність методу генерувати ефективні та збалансовані плани з урахуванням індивідуальних особливостей.

Виконано програмну реалізацію методу планування розвитку фізичних навичок учасників велоклубу з використанням платформи ASP.Net Web Forms та мови програмування C# для створення веб-додатку. Представлено функціональне призначення та структуру програмної реалізації методу, а саме - архітектуру веб-додатку. Особливості реалізації програмних складових методу, таких як реалізація генетичного алгоритму, обробка вхідних даних та генерація планів тренувань, були детально описані та проілюстровані відповідними фрагментами коду.

Проведено тестування програмної реалізації методу та визначено вимоги до розгортання системи в продуктивному середовищі. Наведено приклад застосування розробленого програмного забезпечення для учасника велоклубу Олександра. Після виконання сформованого індивідуального плану тренувань протягом певного часу спостерігалося значне покращення його фізичних навичок - витривалості на 40 хвилин (31% приросту) та сили на 10 віджимань (40% приросту). Цей приклад ілюструє досягнення поставленої мети - покращення зручності планування розвитку фізичних навичок учасників велоклубу завдяки автоматизації процесу формування персоналізованих програм

тренувань з урахуванням індивідуальних особливостей.

Отримані результати демонструють ефективність розробленого методу планування тренувань на основі генетичного алгоритму для досягнення поставленої мети покращення зручності та результативності процесу планування розвитку фізичних навичок учасників велоклубу. Завдяки автоматизації процесу формування індивідуальних планів тренувань та врахуванню особистих характеристик учасників, метод забезпечує значний прогрес у розвитку необхідних фізичних навичок.

Перелік посилань

1. Csuglobal. What Does a Computer Scientist Do On a Daily Basis. URL: https://csuglobal.edu/blog/what-does-computer-scientist-do-daily-basis

2. Sciencedirect. Optimization Task. URL: https://www.sciencedirect.com/topics/computer-science/optimization-task

3. Institut-schmelz. Computer Science in Sport. URL: https://institut-schmelz.univie.ac.at/en/subunits/biomechanics-kinesiology-and-computer-science-in-sport/computer-science-in-sport/

4. Softjourn. Heuristics Unleashed: A Comprehensive Guide to Heuristics in Computer Science and Programming. URL: https://softjourn.com/insights/heuristic-programming

5. Khanacademy. Heuristics & approximate solutions. URL: https://www.khanacademy.org/computing/ap-computer-science-principles/algorithms-101/solving-hard-problems/a/using-heuristics.

6. Towardsdatascience. Instruction to Genetic Algorithms URL: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code

7. SpiceWorks. What Are Genetic Algorithms URL: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-are-genetic-algorithms/

8. Mathworks. What Is the Genetic Algorithm? URL: https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html

9. Baeldung. Real-World Uses for Genetic Algorithms. URL: https://www.baeldung.com/cs/genetic-algorithms-applications

10. Medium. Fitness Functions in Genetic Algorithms: Evaluating Solutions. URL: https://medium.com/@sowmy3010/fitness-functions-in-genetic-algorithms-evaluating-solutions

11. Inderscienceonline. User experience evaluation method of sports product based on genetic algorithm. URL:

https://www.inderscienceonline.com/doi/abs/10.1504/IJPD.2023.129312

12. Zfort. AI in Sports: How is artificial intelligence change sports industry? URL: https://www.zfort.com/blog/AI-in-Sports

13. Imaginovation. How Artificial Intelligence is Transforming the Sports Industry? URL: https://imaginovation.net/blog/ai-in-sports-industry/

14. Springly. Cycling Administration. URL: https://www.springly.org/en-us/nonprofit/cycling-club-software/

15. Siroko. Artificial intelligence in cycling. URL: https://www.siroko.com/blog/c/artificial-intelligence-in-cycling/

16. Cyclingweekly. The future is here: five ways AI is being used in cycling. URL: https://www.cyclingweekly.com/news/the-future-is-here-5-ways-ai-is-being-used-in-cycling

17. Whentotrade. Genetic Algorithm and Cycles. URL: https://whentotrade.com/genetic-algorithm-and-cycles-preview-page/

18. Wikipedia. Cycling Club. URL: https://en.wikipedia.org/wiki/Cycling_club

19. Surreycyclingclub. Joining a Cycling Club. URL: https://surreycyclingclub.co.uk/blog/10-benefits-of-joining-a-cycling-club/

20. Semanticscholar. Sports Analytics: Predicting Athletic Performance with a Genetic Algorithm. URL: https://www.semanticscholar.org/paper/Sports-Analytics%3A-Predicting-Athletic-Performance-a-Cordes-Olfman

21. Medium. Genetic algorithms in Sport. URL: https://medium.com/analytics-vidhya/genetic-algorithm-for-player-scouting

22. Sciencedirect. Using genetic algorithms to abbreviate the Mindfulness Inventory for Sport. https://www.sciencedirect.com/science/article/abs/pii/S1469029218306757

23. Mdpi. Genetic Hybrid Optimization of a Real Bike Sharing System. URL: https://www.mdpi.com/2227-7390/9/18/2227

24. Cactusware. Sports Technology That’s Revolutionizing Sports in 2024. URL: https://cactusware.com/blog/sports-technology

25. Digiteum. Impact of Technologies on Sports and Fitness. URL: https://www.digiteum.com/impact-of-technology-sports-fitness/

26. Tritrainingharder. AI Training Plans. URL: https://tritrainingharder.com/all-training-plans

27. Appinventiv. AI in Sport. URL: https://appinventiv.com/blog/ai-in-sports

28. Medium. A Fundamental Introduction to Genetic Algorithm. URL: https://medium.com/@h.chegini/a-fundamental-introduction-to-genetic-algorithm-part-two-4abfd0ce1eeb

29. Strava. Dashboadr. URL: https://www.strava.com/dashboard

30. Strava. URL: https://www.strava.com/dashboard

31. Runnersworld. How to Start Using Strava. URL: https://www.runnersworld.com/beginner/g25619156/what-is-strava/

32. Strava. URL: https://www.strava.com/activities/11670452757

33. Strava. Recording an Activity. URL: https://support.strava.com/hc/en-us/articles/216917397-Recording-an-Activity

34. Strava. My Segment Results. URL: https://support.strava.com/hc/en-us/articles/216917447-My-Segment-Results

35. Strava. URL: https://www.strava.com/activities/11665160370

36. Strava. Routes on Web. URL: https://support.strava.com/hc/en-us/articles/216918387-Routes-on-Web

37. Strava. URL: https://www.strava.com/clubs/Kremenets

38. Strava. Athlete Intelligence on Strava. URL: https://support.strava.com/hc/en-us/articles/26786795557005-Athlete-Intelligence-on-Strava-Beta

39. Trainingpeaks. Streamline your training. URL: https://www.trainingpeaks.com/

40. TrainingPeaks. URL: https://www.trainingpeaks.com/

41. Robertovukovic. Reasons Why I Use TrainingPeaks and You Should Too. URL: https://robertovukovic.com/why-use-training-peaks/

42. TrainingPeaks. URL: https://www.trainingpeaks.com/training-

plans/cycling/?language=uk&sort.field=soldCount&sort.dir=desc

43. Trainingpeaks. TrainingPeaks Athlete User Guide. URL: https://help.trainingpeaks.com/hc/en-us/articles/231472468-TrainingPeaks-Athlete-User-Guide

44. Trainingpeaks. An Introduction to TrainingPeaks Metrics. URL: https://www.trainingpeaks.com/learn/articles/an-introduction-to-trainingpeaks-metrics/

45. TrainingPeaks. URL: https://www.trainingpeaks.com/training-plans/cycling/road-cycling/tp-222476/

46. Trainingpeaks. Full distance triathlon guide. URL: https://www.trainingpeaks.com/guides/full-distance-triathlon/

47. TrainingPeaks. URL: https://www.trainingpeaks.com/training-plans/cycling/?language=uk&sort.field=soldCount&sort.dir=desc

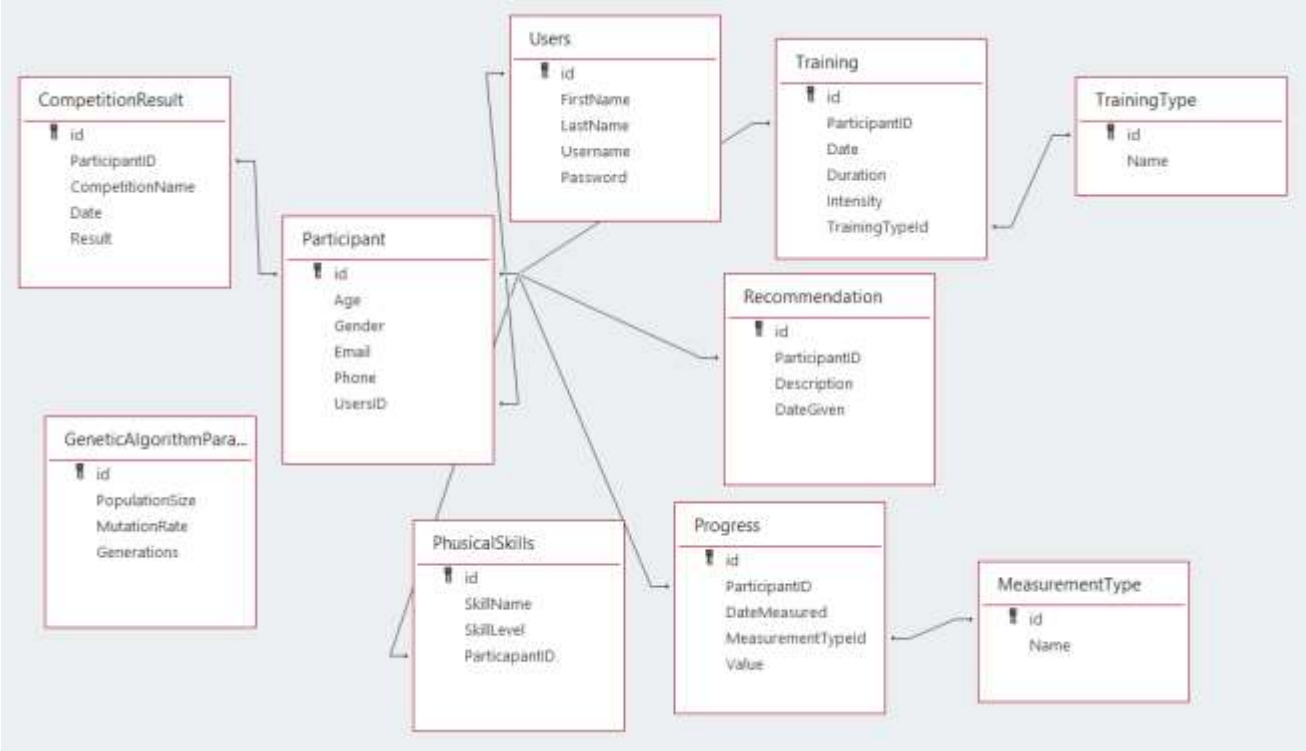
48. Trainingpeaks. The TrainingPeaks Quick Start Guide. URL: https://www.trainingpeaks.com/learn/

49. Bicycling. A Cyclist's Complete Guide to Understanding TrainingPeaks. URL: https://www.bicycling.com/training/a38795537/how-to-use-trainingpeaks/

ДОДАТКИ

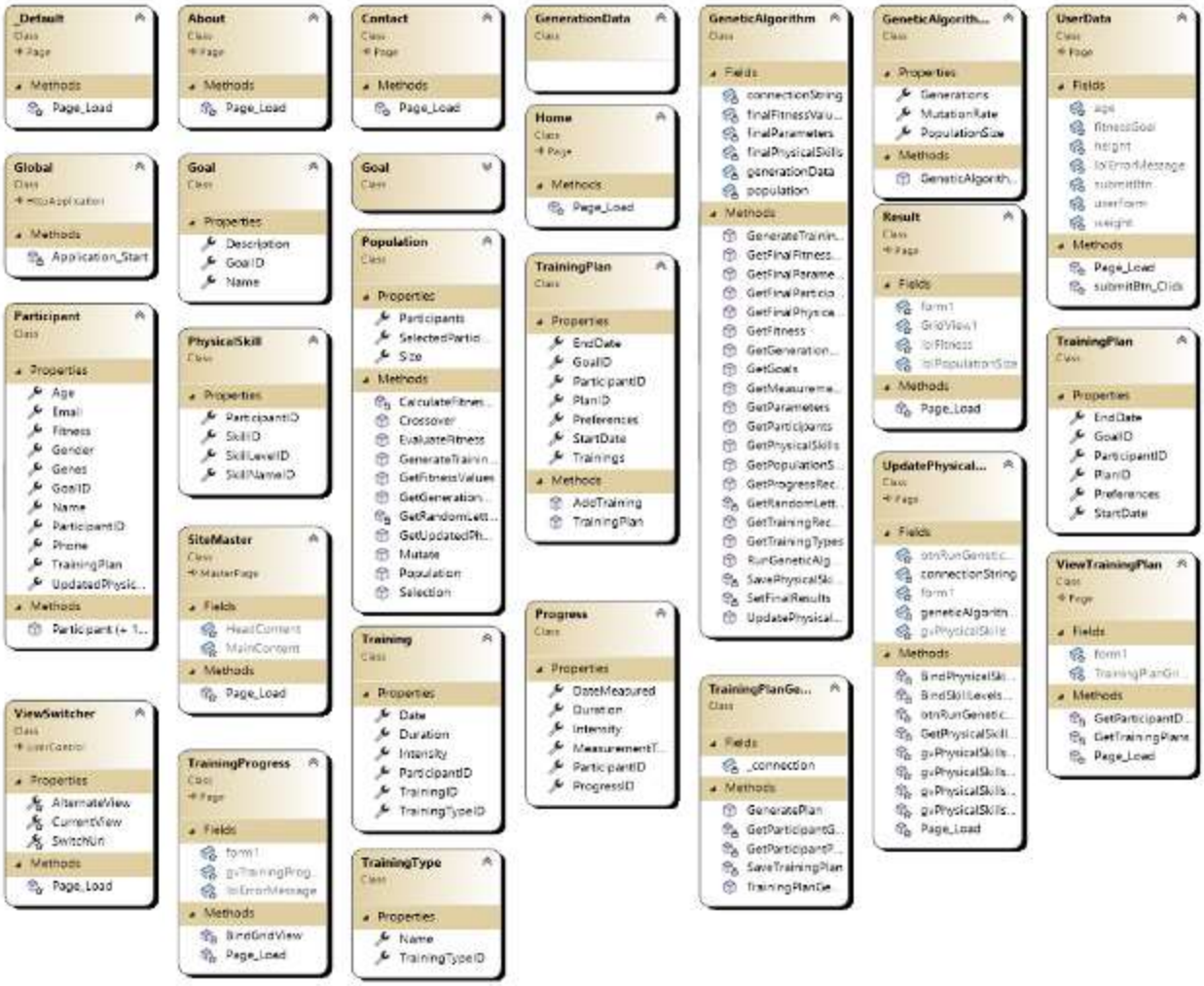
Додаток А

Структура бази даних планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб-системи велоклубу



Додаток Б

Розгорнута структура класів планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб-системи велоклубу



Додаток В

Програмні коди

Лістинг GeneticAlgorithm.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Collections.Generic;
using System.Data.SqlClient;
using BikeClub.Algorithm;
using System.Diagnostics;
using System.Configuration;
using BikeClub.Data;
using System.Drawing;

public class GeneticAlgorithm
{
    private string connectionString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Admin\source\repos\BikeClub\Bike
Club\App_Data\Database1.mdf;Integrated Security=True";

    private List<PhysicalSkill> finalPhysicalSkills;
    private List<double> finalFitnessValues;
    private List<GeneticAlgorithmParameter> finalParameters;
    private List<GenerationData> generationData;
    private Population population;

    public void RunGeneticAlgorithm()
    {
        // Підключення до бази даних
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();

            // отримання даних з бази даних
            List<Participant> participants = GetParticipants(connection);
            List<PhysicalSkill> physicalSkills = GetPhysicalSkills(connection);
            List<GeneticAlgorithmParameter> parameters = GetParameters(connection);
            List<Progress> progressRecords = GetProgressRecords(connection);
            List<Training> trainingRecords = GetTrainingRecords(connection);
            List<TrainingType> trainingTypes = GetTrainingTypes(connection);
            List<MeasurementType> measurementTypes = GetMeasurementTypes(connection);
            List<Goal> goals = GetGoals(connection);

            // конвертація PopulationSize в int
            int populationSize = parameters[0].PopulationSize;

            // ініціалізація популяції
            population = new Population(populationSize, participants);

            // виконання відбору, кросинговеру та мутації
            int generations = parameters[0].Generations;
            double mutationRate = parameters[0].MutationRate;

            for (int i = 0; i < generations; i++)
            {
                // оцінювання придатності
                population.EvaluateFitness(physicalSkills, progressRecords,
```

```
trainingRecords, trainingTypes, measurementTypes);

                // відбір
                population.Selection(physicalSkills, progressRecords, trainingRecords,
trainingTypes, measurementTypes);

                // кросинговер
                population.Crossover(population.SelectedParticipants);

                // мутація
                population.Mutate(population.SelectedParticipants, mutationRate);

                population.GenerateTrainingPlans(trainingTypes, goals, connection);
            }

            // Після завершення циклу генетичного алгоритму, встановити фінальні
результати
            SetFinalResults(population.GetUpdatedPhysicalSkills(),
population.GetFitnessValues(), parameters, population.GetGenerationData());

            // зберегти оновлені фізичні навички в базі даних
            SavePhysicalSkills(population.GetUpdatedPhysicalSkills(), connection);
        }
    }


    public List<Participant> GetParticipants(SqlConnection connection)
    {
        List<Participant> participants = new List<Participant>();

        // Виконання SQL-запиту для отримання учасників
        using (SqlCommand command = new SqlCommand("SELECT * FROM Participant",
connection))
        {
            using (SqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    string genesValue = "example_genes"; // Замініть на реальне значення
генів, якщо необхідно
                    Participant participant = new Participant(genesValue)
                    {
                        ParticipantID = reader["ParticipantID"] != DBNull.Value ?
Convert.ToInt32(reader["ParticipantID"]) : 0,
                        Age = reader["Age"] != DBNull.Value ? reader["Age"].ToString() :
string.Empty,
                        Gender = reader["Gender"] != DBNull.Value ?
reader["Gender"].ToString() : string.Empty,
                        Weight = reader["Weight"] != DBNull.Value ?
reader["Weight"].ToString() : string.Empty,
                        Height = reader["Height"] != DBNull.Value ?
reader["Height"].ToString() : string.Empty,
                        GoalID = reader["GoalID"] != DBNull.Value ?
Convert.ToInt32(reader["GoalID"]) : 0,
                        UserID = reader["UserID"] != DBNull.Value ?
Convert.ToInt32(reader["UserID"]) : 0,
                    };
                    participants.Add(participant);
                }
            }
        }

        return participants;
```

```
    }

    public List<PhysicalSkill> GetPhysicalSkills(SqlConnection connection)
    {
        List<PhysicalSkill> physicalSkills = new List<PhysicalSkill>();

        // Виконання SQL-запиту для отримання фізичних навичок з JOIN для назв
        string query = @"
SELECT ps.SkillID, ps.SkillNameID, sn.Name AS SkillName, ps.SkillLevelID, sl.Name AS
SkillLevel, ps.ParticipantID
FROM PhysicalSkills ps
JOIN SkillName sn ON ps.SkillNameID = sn.SkillNameID
JOIN SkillLevel sl ON ps.SkillLevelID = sl.SkillLevelID";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            using (SqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    PhysicalSkill skill = new PhysicalSkill
                    {
                        SkillID = Convert.ToInt32(reader["SkillID"]),
                        SkillNameID = Convert.ToInt32(reader["SkillNameID"]), //
Correctly obtaining the integer value
                        SkillLevelID = reader["SkillLevelID"] != DBNull.Value ?
Convert.ToInt32(reader["SkillLevelID"]) : (int?)null,
                        ParticipantID = Convert.ToInt32(reader["ParticipantID"])
                    };
                    physicalSkills.Add(skill);
                }
            }
        }

        return physicalSkills;
    }

    public List<Progress> GetProgressRecords(SqlConnection connection)
    {
        List<Progress> progressRecords = new List<Progress>();

        // Виконання SQL-запиту для отримання записів прогресу
        using (SqlCommand command = new SqlCommand("SELECT * FROM Progress",
connection))
        {
            using (SqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    Progress progressRecord = new Progress
                    {
                        ProgressID = Convert.ToInt32(reader["ProgressID"]),
                        DateMeasured = Convert.ToDateTime(reader["DateMeasured"]),
                        Duration = reader["Duration"].ToString(),
                        Intensity = reader["Intensity"].ToString(),
                        ParticipantID = Convert.ToInt32(reader["ParticipantID"]),
                        MeasurementTypeID = Convert.ToInt32(reader["MeasurementTypeID"])
                    };
                    progressRecords.Add(progressRecord);
                }
            }
        }
```

```
        return progressRecords;
    }

    public List<Training> GetTrainingRecords(SqlConnection connection)
    {
        List<Training> trainingRecords = new List<Training>();

        // Виконання SQL-запиту для отримання записів тренувань
        using (SqlCommand command = new SqlCommand("SELECT * FROM Training",
connection))
        {
            using (SqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    Training trainingRecord = new Training
                    {
                        TrainingID = Convert.ToInt32(reader["TrainingID"]),
                        Date = Convert.ToDateTime(reader["Date"]),
                        Duration = reader["Duration"].ToString(),
                        Intensity = reader["Intensity"].ToString(),
                        ParticipantID = Convert.ToInt32(reader["ParticipantID"]),
                        TrainingTypeID = Convert.ToInt32(reader["TrainingTypeID"])
                    };
                    trainingRecords.Add(trainingRecord);
                }
            }
        }

        return trainingRecords;
    }

    public List<TrainingType> GetTrainingTypes(SqlConnection connection)
    {
        List<TrainingType> trainingTypes = new List<TrainingType>();

        // Виконання SQL-запиту для отримання типів тренувань
        using (SqlCommand command = new SqlCommand("SELECT * FROM TrainingType",
connection))
        {
            using (SqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    TrainingType trainingType = new TrainingType
                    {
                        TrainingTypeID = Convert.ToInt32(reader["TrainingTypeID"]),
                        Name = reader["Name"].ToString()
                    };
                    trainingTypes.Add(trainingType);
                }
            }
        }

        return trainingTypes;
    }

    public List<MeasurementType> GetMeasurementTypes(SqlConnection connection)
    {
        List<MeasurementType> measurementTypes = new List<MeasurementType>();

        // Виконання SQL-запиту для отримання типів вимірювань
        using (SqlCommand command = new SqlCommand("SELECT * FROM MeasurementType",
```

```
connection))
        {
            using (SqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    MeasurementType measurementType = new MeasurementType
                    {
                        MeasurementTypeID =
Convert.ToInt32(reader["MeasurementTypeID"]),
                        Name = reader["Name"].ToString()
                    };
                    measurementTypes.Add(measurementType);
                }
            }
        }

        return measurementTypes;
    }

    public List<Goal> GetGoals(SqlConnection connection)
    {
        List<Goal> goals = new List<Goal>();

        string query = "SELECT GoalID, Name, Description FROM Goals";
        using (SqlCommand command = new SqlCommand(query, connection))
        {
            using (SqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    Goal goal = new Goal
                    {
                        GoalID = reader.GetInt32(reader.GetOrdinal("GoalID")),
                        Name = reader.IsDBNull(reader.GetOrdinal("Name")) ? null :
reader.GetString(reader.GetOrdinal("Name")),
                        Description = reader.IsDBNull(reader.GetOrdinal("Description"))
? null : reader.GetString(reader.GetOrdinal("Description"))
                    };
                    goals.Add(goal);
                }
            }
        }

        return goals;
    }

    private void SavePhysicalSkills(List<PhysicalSkill> physicalSkills, SqlConnection
connection)
    {
        // Виконання SQL-запиту для оновлення фізичних навичок
        foreach (PhysicalSkill skill in physicalSkills)
        {
            string query = "UPDATE PhysicalSkills SET SkillLevelID = @SkillLevelID WHERE
SkillID = @SkillID";
            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@SkillLevelID", skill.SkillLevelID);
                command.Parameters.AddWithValue("@SkillID", skill.SkillID);
                command.ExecuteNonQuery();
            }
        }
    }
```

```
    public List<GeneticAlgorithmParameter> GetParameters(SqlConnection connection)
    {
        List<GeneticAlgorithmParameter> parameters = new
List<GeneticAlgorithmParameter>();

        // Виконання SQL-запиту для отримання параметрів
        using (SqlCommand command = new SqlCommand("SELECT * FROM
GeneticAlgorithmParameters", connection))
        {
            using (SqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    GeneticAlgorithmParameter parameter = new GeneticAlgorithmParameter
                    {
                        PopulationSize =
reader.GetInt32(reader.GetOrdinal("PopulationSize")),
                        MutationRate =
reader.GetDouble(reader.GetOrdinal("MutationRate")),
                        Generations = reader.GetInt32(reader.GetOrdinal("Generations"))
                    };
                    parameters.Add(parameter);
                }
            }
        }

        return parameters;
    }



    private char GetRandomLetter()
    {
        // Генерація випадкової літери для мутації
        Random random = new Random();
        int num = random.Next(0, 26); // Генеруємо випадкове число від 0 до 25
        char letter = (char)('a' + num); // Перетворюємо це число у літеру англійського
алфавіту
        return letter;
    }

    //фінальні результати
    public List<PhysicalSkill> GetFinalPhysicalSkills()
    {
        return finalPhysicalSkills;
    }

    public List<double> GetFinalFitnessValues()
    {
        return finalFitnessValues;
    }

    public List<GeneticAlgorithmParameter> GetFinalParameters()
    {
        return finalParameters;
    }

    public List<GenerationData> GetGenerationData()
    {
        return generationData;
    }
```

```
    public double GetFitness()
    {
        // Перевірка на нуль та порожність колекції перед викликом Average()
        if (finalFitnessValues != null && finalFitnessValues.Any())
        {
            return finalFitnessValues.Average();
        }
        else
        {
            // Повернення 0, якщо список порожній або нульовий
            return 0.0;
        }
    }

    public int GetPopulationSize()
    {
        // Отримати розмір популяції з фінальних параметрів
        if (finalParameters != null && finalParameters.Count > 0)
        {
            return finalParameters[0].PopulationSize;
        }
        else
        {
            // Обробити ситуацію, коли finalParameters є null або порожнім
            return 0; // або якесь інше значення за замовчуванням
        }
    }

    public List<Participant> GetFinalParticipants()
    {
        // Повернути учасників з останньої популяції
        if (population != null)
        {
            return population.Participants;
        }
        else
        {
            // Обробити ситуацію, коли population є null
            return new List<Participant>(); // або якесь інше значення за замовчуванням
        }
    }

    private void SetFinalResults(List<PhysicalSkill> updatedPhysicalSkills, List<double>
fitnessValues, List<GeneticAlgorithmParameter> parameters, List<GenerationData>
generationData)
    {
        finalPhysicalSkills = updatedPhysicalSkills;
        finalFitnessValues = fitnessValues;
        finalParameters = parameters;
        this.generationData = generationData;
    }

    public void UpdatePhysicalSkill(int skillID, string skillLevelID, SqlConnection
connection)
    {
        string query = "UPDATE PhysicalSkills SET SkillLevelID = @SkillLevelID WHERE
SkillID = @SkillID";
        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@SkillLevelID", skillLevelID);
            command.Parameters.AddWithValue("@SkillID", skillID);
```

```
            command.ExecuteNonQuery();
        }
    }

    public void GenerateTrainingPlansForParticipants()
    {
        // Retrieve updated participants' details after running the genetic algorithm
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();
            List<Participant> participants = GetParticipants(connection); // Now passing
the connection object

            foreach (var participant in participants)
            {
                // Create a new instance of the TrainingPlanGenerator for each
participant
                // Assuming TrainingPlanGenerator is defined in the same project and
namespace
                TrainingPlanGenerator planGenerator = new
TrainingPlanGenerator(connection);
                planGenerator.GeneratePlan(participant);
            }
        }
    }


}

public class Participant
{
    public int ParticipantID { get; set; }
    public string Gender { get; set; }
    public string Weight { get; set; }
    public string Height { get; set; }
    public string Age { get; set; }
    public double Fitness { get; set; }
    public int GoalID { get; set; }
    public int UserID { get; set; }

    public string Genes { get; set; }

    public TrainingPlan TrainingPlan { get; set; }
    public List<PhysicalSkill> UpdatedPhysicalSkills { get; set; } = new
List<PhysicalSkill>(); // Initialize with an empty list

    // Default constructor without parameters
    public Participant()
    {
    }

    // Constructor with Genes parameter
    public Participant(string genes)
    {
        Genes = genes;
    }
}

public class TrainingPlan
{
    public int PlanID { get; set; }
    public int ParticipantID { get; set; }
    public DateTime StartDate { get; set; }
```

```
    public DateTime EndDate { get; set; }
    public int GoalID { get; set; }
    public string Preferences { get; set; }
    public List<Training> Trainings { get; set; }

    public TrainingPlan()
    {
        Trainings = new List<Training>();
    }

    public void AddTraining(Training training)
    {
        Trainings.Add(training);
    }
}

public class PhysicalSkill
{
    public int SkillID { get; set; }
    public int SkillNameID { get; set; } // Assuming this is also an integer based on
your schema
    public int? SkillLevelID { get; set; } // Nullable to handle possible null values
from the database
    public int ParticipantID { get; set; }
}
public class Progress
{
    public int ProgressID { get; set; }
    public DateTime DateMeasured { get; set; }
    public string Duration { get; set; }
    public string Intensity { get; set; }
    public int ParticipantID { get; set; }
    public int MeasurementTypeID { get; set; }
}
public class Goal
{
    public int GoalID { get; set; }
    public string Name { get; set; }
    public string Description { get; set; }
}
public class Training
{
    public int TrainingID { get; set; }
    public DateTime Date { get; set; }
    public string Duration { get; set; }
    public string Intensity { get; set; }
    public int ParticipantID { get; set; }
    public int TrainingTypeID { get; set; }
}

public class TrainingType
{
    public int TrainingTypeID { get; set; }
    public string Name { get; set; }
}

public class SkillLevel
{
    public int SkilllevelID { get; set; }
    public string Name { get; set; }
}

public class MeasurementType
```

```
{
    public int MeasurementTypeID { get; set; }
    public string Name { get; set; }
}

public class GeneticAlgorithmParameter
{
    public int PopulationSize { get; set; }
    public double MutationRate { get; set; }
    public int Generations { get; set; }
    // Додайте інші параметри алгоритму

    public GeneticAlgorithmParameter() { }

    public GeneticAlgorithmParameter(int populationSize, double mutationRate, int
generations)
    {
        PopulationSize = populationSize;
        MutationRate = mutationRate;
        Generations = generations;
        // Ініціалізуйте інші параметри
    }

}

public class GenerationData
{
    // Оголошення полів та методів
}
```

Лістинг Population.cs:

```
using BikeClub.Data;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using BikeClub.Data;
using System.Data.SqlClient;

namespace BikeClub.Algorithm
{
    public class Population
    {
        public int Size { get; set; }
        public List<Participant> Participants { get; set; }
        public List<Participant> SelectedParticipants { get; set; }
        private int generationsWithoutImprovement = 0;
        private double previousBestFitness = double.MinValue;

        public Population(int size, List<Participant> participants)
        {
            Size = size;
            Participants = participants;
        }

        public void EvaluateFitness(List<PhysicalSkill> physicalSkills, List<Progress>
progressRecords, List<Training> trainingRecords, List<TrainingType> trainingTypes,
List<MeasurementType> measurementTypes)
        {
            foreach (Participant participant in Participants)
            {
                double fitness = CalculateFitnessForParticipant(participant,
```

```
physicalSkills, progressRecords, trainingRecords, trainingTypes, measurementTypes);
                participant.Fitness = fitness;
            }

            // Update generationsWithoutImprovement
            double currentBestFitness = Participants.Max(p => p.Fitness);
            if (currentBestFitness > previousBestFitness)
            {
                generationsWithoutImprovement = 0;
                previousBestFitness = currentBestFitness;
            }
            else
            {
                generationsWithoutImprovement++;
            }
        }

        public void Selection(List<PhysicalSkill> physicalSkills, List<Progress>
progressRecords, List<Training> trainingRecords, List<TrainingType> trainingTypes,
List<MeasurementType> measurementTypes)
        {
            // Оцінювання придатності
            EvaluateFitness(physicalSkills, progressRecords, trainingRecords,
trainingTypes, measurementTypes);

            // Сортування учасників за придатністю у зворотному порядку (найкращий
першим)
            Participants.Sort((p1, p2) => p2.Fitness.CompareTo(p1.Fitness));

            // Вибір верхніх 50% учасників
            SelectedParticipants = Participants.Take(Size / 2).ToList();
        }

        private double CalculateFitnessForParticipant(Participant participant,
List<PhysicalSkill> physicalSkills, List<Progress> progressRecords, List<Training>
trainingRecords, List<TrainingType> trainingTypes, List<MeasurementType>
measurementTypes)
        {
            double fitness = 0.0;

            // рівень фізичних навичок
            foreach (var skill in physicalSkills)
            {
                if (skill.ParticipantID == participant.ParticipantID)
                {
                    if (skill.SkillLevelID.Equals("Beginner"))
                    {
                        fitness += 0.5;
                    }
                    else if (skill.SkillLevelID.Equals("Intermediate"))
                    {
                        fitness += 0.75;
                    }
                    else if (skill.SkillLevelID.Equals("Advanced"))
                    {
                        fitness += 1.0;
                    }
                }
            }

            // Врахуйте прогрес
            foreach (var progress in progressRecords)
            {
```

```
                if (progress.ParticipantID == participant.ParticipantID)
                {
                    fitness += 0.1;
                }
            }

            // Поверніть загальну оцінку придатності
            return fitness;
        }

        public List<Participant> Selection()
        {
            Participants.Sort((p1, p2) => p2.Fitness.CompareTo(p1.Fitness));
            SelectedParticipants = Participants.Take(Size / 2).ToList();
            return SelectedParticipants;
        }

        public List<Participant> Crossover(List<Participant> selectedParticipants)
        {
            List<Participant> newParticipants = new List<Participant>();
            Random random = new Random();

            for (int i = 0; i < selectedParticipants.Count - 1; i += 2)
            {
                Participant parent1 = selectedParticipants[i];
                Participant parent2 = selectedParticipants[i + 1];

                // Багаторівневий кросинговер
                int crossoverPoint1 = random.Next(1, parent1.Genes.Length / 2);
                int crossoverPoint2 = random.Next(parent1.Genes.Length / 2,
parent1.Genes.Length);

                string childGenes1 = parent1.Genes.Substring(0, crossoverPoint1) +
parent2.Genes.Substring(crossoverPoint1, crossoverPoint2 - crossoverPoint1) +
parent1.Genes.Substring(crossoverPoint2);
                string childGenes2 = parent2.Genes.Substring(0, crossoverPoint1) +
parent1.Genes.Substring(crossoverPoint1, crossoverPoint2 - crossoverPoint1) +
parent2.Genes.Substring(crossoverPoint2);

                Participant child1 = new Participant(childGenes1);
                Participant child2 = new Participant(childGenes2);

                newParticipants.Add(child1);
                newParticipants.Add(child2);
            }

            return newParticipants;
        }

        public void Mutate(List<Participant> participants, double baseMutationRate)
        {
            Random random = new Random();
            double dynamicMutationRate = baseMutationRate +
generationsWithoutImprovement * 0.01; // Adjust mutation rate based on generations
without improvement

            foreach (Participant participant in participants)
            {
                if (random.NextDouble() < dynamicMutationRate)
                {
                    int mutationIndex = random.Next(0, participant.Genes.Length);
                    char[] genesArray = participant.Genes.ToCharArray();
                    genesArray[mutationIndex] = GetRandomLetter();
```

```
                    participant.Genes = new string(genesArray);
                }
            }
        }

        public List<PhysicalSkill> GetUpdatedPhysicalSkills()
        {
            List<PhysicalSkill> updatedPhysicalSkills = new List<PhysicalSkill>();

            foreach (var participant in Participants)
            {
                if (participant.UpdatedPhysicalSkills != null)
                {
                    updatedPhysicalSkills.AddRange(participant.UpdatedPhysicalSkills);
                }
            }

            return updatedPhysicalSkills;
        }

        public List<double> GetFitnessValues()
        {
            List<double> fitnessValues = new List<double>();

            foreach (var participant in Participants)
            {
                fitnessValues.Add(participant.Fitness);
            }

            return fitnessValues;
        }

        public List<GenerationData> GetGenerationData()
        {
            return new List<GenerationData>();
        }

        public void GenerateTrainingPlans(List<TrainingType> trainingTypes, List<Goal>
goals, SqlConnection connection)
        {
            TrainingPlanGenerator generator = new TrainingPlanGenerator(connection);

            foreach (var participant in SelectedParticipants)
            {
                generator.GeneratePlan(participant);
            }
        }

        private char GetRandomLetter()
        {
            Random random = new Random();
            int num = random.Next(0, 26);
            char letter = (char)('a' + num);
            return letter;
        }
    }
}
```

Лістинг SkillLevelrepository.cs:

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using BikeClub.Data;
using BikeClub.Algorithm;
using System.Data.SqlClient;

namespace BikeClub.Data
{
    public class SkillLevelRepository
    {
        private readonly string connectionString;

        public SkillLevelRepository(string connectionString)
        {
            this.connectionString = connectionString;
        }

        public List<SkillLevel> GetSkillLevels()
        {
            List<SkillLevel> skillLevels = new List<SkillLevel>();

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();

                string query = "SELECT SkilllevelID, Name FROM SkillLevel";

                using (SqlCommand command = new SqlCommand(query, connection))
                {
                    using (SqlDataReader reader = command.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            SkillLevel skillLevel = new SkillLevel
                            {
                                SkilllevelID =
reader.GetInt32(reader.GetOrdinal("SkilllevelID")),
                                Name = reader.GetString(reader.GetOrdinal("Name"))
                            };

                            skillLevels.Add(skillLevel);
                        }
                    }
                }
            }

            return skillLevels;
        }
    }
}
```

Лістинг TrainingPlan.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace BikeClub.Data
{
    public class TrainingPlan
    {
        public int PlanID { get; set; } // Auto-incremented by the database
```

```
        public int ParticipantID { get; set; }
        public DateTime StartDate { get; set; }
        public DateTime EndDate { get; set; }
        public int GoalID { get; set; }
        public string Preferences { get; set; }
    }
}
```

Лістинг TrainingPlanGenerator.cs

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using BikeClub.Data;

namespace BikeClub.Data
{
    public class TrainingPlanGenerator
    {
        private readonly SqlConnection _connection;

        public TrainingPlanGenerator(SqlConnection connection)
        {
            _connection = connection;
        }

        public void GeneratePlan(Participant participant)
        {
            // Retrieve participant goals and preferences
            Goal participantGoal = GetParticipantGoal(participant.GoalID);
            string participantPreferences =
GetParticipantPreferences(participant.ParticipantID);

            // Determine the appropriate start date, end date, and other details for the
training plan
            // based on the participant's current skill level, goals, and preferences.
            DateTime startDate = DateTime.Today; // Placeholder for demonstration
purposes
            DateTime endDate = startDate.AddMonths(1); // Placeholder for an example
one-month plan

            // Create a new TrainingPlan object
            TrainingPlan plan = new TrainingPlan
            {
                ParticipantID = participant.ParticipantID,
                StartDate = startDate,
                EndDate = endDate,
                GoalID = participant.GoalID,
                Preferences = participantPreferences
            };

            // Save the training plan to the database
            SaveTrainingPlan(plan);
        }

        private Goal GetParticipantGoal(int goalId)
        {
            string query = "SELECT GoalID, Name, Description FROM Goals WHERE GoalID =
@GoalID";

            using (SqlCommand command = new SqlCommand(query, _connection))
```

```
            {
                command.Parameters.AddWithValue("@GoalID", goalId);

                using (SqlDataReader reader = command.ExecuteReader())
                {
                    if (reader.Read())
                    {
                        return new Goal
                        {
                            GoalID = reader.GetInt32(0),
                            Name = reader.IsDBNull(1) ? null : reader.GetString(1),
                            Description = reader.IsDBNull(2) ? null :
reader.GetString(2)
                        };
                    }
                }
            }

            return null; // Повернути null, якщо Goal не знайдено
        }

        private string GetParticipantPreferences(int participantId)
        {
            // Implement logic to retrieve the participant's preferences
            // Placeholder for demonstration purposes
            return "Example preferences";
        }

        private void SaveTrainingPlan(TrainingPlan plan)
        {
            // Перевірити, чи існує GoalID у таблиці Goals
            bool goalExists = CheckGoalExists(plan.GoalID);

            if (!goalExists)
            {
                // Якщо GoalID не існує, не вставляти TrainingPlan
                return;
            }

            string query = @"
        INSERT INTO TrainingPlans (ParticipantID, StartDate, EndDate, GoalID,
Preferences)
        VALUES (@ParticipantID, @StartDate, @EndDate, @GoalID, @Preferences)";

            using (SqlCommand command = new SqlCommand(query, _connection))
            {
                command.Parameters.AddWithValue("@ParticipantID", plan.ParticipantID);
                command.Parameters.AddWithValue("@StartDate", plan.StartDate);
                command.Parameters.AddWithValue("@EndDate", plan.EndDate);
                command.Parameters.AddWithValue("@GoalID", plan.GoalID);
                command.Parameters.AddWithValue("@Preferences", (object)plan.Preferences
?? DBNull.Value);

                command.ExecuteNonQuery();
            }
        }

        private bool CheckGoalExists(int goalId)
        {
            string query = "SELECT COUNT(*) FROM Goals WHERE GoalID = @GoalID";

            using (SqlCommand command = new SqlCommand(query, _connection))
            {
```

```
                command.Parameters.AddWithValue("@GoalID", goalId);

                int count = (int)command.ExecuteScalar();
                return count > 0;
            }
        }

    }
    // Placeholder class for goals, adjust as needed for your application
    public class Goal
    {
        public int GoalID { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
    }
}
```

Лістинг Login.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace BikeClub
{
    public partial class Login : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void btnLogin_Click(object sender, EventArgs e)
        {
            string username = txtUsername.Text;
            string password = txtPassword.Text;

            // Підключення до бази даних
            string connectionString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Admin\source\repos\BikeClub\Bike
Club\App_Data\Database1.mdf;Integrated Security=True";

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                SqlCommand command = new SqlCommand("SELECT UserID FROM Users WHERE
Username = @Username AND Password = @Password", connection);
                command.Parameters.AddWithValue("@Username", username);
                command.Parameters.AddWithValue("@Password", password);

                SqlDataReader reader = command.ExecuteReader();

                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        // Збереження UserID в сеансі
                        int userID = Convert.ToInt32(reader["UserID"]);
                        Session["UserID"] = userID;
```

```
                        Session["Username"] = username; // Збереження ім'я користувача в
сеансі
                        Response.Redirect("UserDashboard.aspx");
                    }
                }
                else
                {
                    // Якщо користувача не знайдено, відображення повідомлення про
помилку
                    lblErrorMessage.Text = "Невірний логін або пароль. Спробуйте ще
раз.";
                }
            }
        }
    }
}
```

Лістинг Registration.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace BikeClub
{
    public partial class Registration : Page
    {
        protected void btnRegister_Click(object sender, EventArgs e)
        {
            string firstName = txtFirstName.Text;
            string lastName = txtLastName.Text;
            string userName = txtUserName.Text; // Використовуйте відповідну змінну
            string password = txtPassword.Text; // Використовуйте відповідну змінну

            string connectionString = "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\Admin\\source\\repos\\BikeClu
b\\BikeClub\\App_Data\\Database1.mdf;Integrated Security=True";

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();

                // Вставка даних у таблицю Users
                string insertUserQuery = "INSERT INTO Users (FirstName, LastName,
Username, Password) VALUES (@FirstName, @LastName, @Username, @Password)";
                SqlCommand userCommand = new SqlCommand(insertUserQuery, connection);
                userCommand.Parameters.AddWithValue("@FirstName", firstName);
                userCommand.Parameters.AddWithValue("@LastName", lastName);
                userCommand.Parameters.AddWithValue("@Username", userName);
                userCommand.Parameters.AddWithValue("@Password", password);

                // Виконання запиту без очікування повернення значення
                userCommand.ExecuteNonQuery();
            }

            Response.Redirect("Home.aspx");
        }
    }
}
```

Лістинг TrainingPlan.aspx.cs

```
using BikeClub.Data;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace BikeClub
{
    public partial class TrainingPlan : System.Web.UI.Page
    {
        private string connectionString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Admin\source\repos\BikeClub\Bike
Club\App_Data\Database1.mdf;Integrated Security=True";

        protected void Page_Load(object sender, EventArgs e)
        {
            lblErrorMessage.Text = "Page_Load called";
            if (!IsPostBack)
            {
                if (Session["UserID"] != null)
                {
                    int userId = Convert.ToInt32(Session["UserID"]);
                    LoadTrainingPlans(userId);
                    LoadTrainingData(userId);
                }
                else
                {
                    Response.Redirect("Login.aspx");
                }
            }
        }

        protected void btnGeneratePlan_Click(object sender, EventArgs e)
        {
            if (Session["UserID"] != null)
            {
                int userId = Convert.ToInt32(Session["UserID"]);

                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    connection.Open();
                    Participant participant = GetParticipant(userId, connection);

                    if (participant != null)
                    {
                        TrainingPlanGenerator generator = new
TrainingPlanGenerator(connection);
                        bool newPlanCreated = generator.GeneratePlan(participant);

                        if (newPlanCreated)
                        {
                            lblSuccessMessage.Text = "New training plan generated
successfully.";
                        }
                        else
```

```
                        {
                            lblSuccessMessage.Text = "You already have a training plan.
To generate a new one, please complete or delete the existing plan.";
                        }

                        GeneticAlgorithm ga = new GeneticAlgorithm();
                        ga.RunGeneticAlgorithm();

                        List<PhysicalSkill> skills = ga.GetFinalPhysicalSkills()
                                                       .Where(s => s.ParticipantID ==
participant.ParticipantID)
                                                       .ToList();

                        string recommendation = GenerateRecommendation(skills,
participant);
                        lblRecommendation.Text = recommendation;
                    }
                }

                LoadTrainingPlans(userId);  // Оновити список планів
            }
            else
            {
                Response.Redirect("Login.aspx");
            }
        }

        private void GenerateTrainingPlan(int userId)
        {
            try
            {
                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    connection.Open();
                    string query = "SELECT * FROM Participant WHERE UserID = @UserID";
                    using (SqlCommand command = new SqlCommand(query, connection))
                    {
                        command.Parameters.AddWithValue("@UserID", userId);
                        using (SqlDataReader reader = command.ExecuteReader())
                        {
                            if (reader.Read())
                            {
                                Participant participant = new Participant
                                {
                                    ParticipantID = (int)reader["ParticipantID"],
                                    Gender = reader["Gender"].ToString(),
                                    Weight = reader["Weight"].ToString(),
                                    Height = reader["Height"].ToString(),
                                    Age = reader["Age"].ToString(),
                                    GoalID = (int)reader["GoalID"]
                                };

                                reader.Close();

                                TrainingPlanGenerator generator = new
TrainingPlanGenerator(connection);
                                generator.GeneratePlan(participant);

                                string insertQuery = @"
INSERT INTO [TrainingPlans] (ParticipantID, StartDate, EndDate, GoalID, Preferences)
VALUES (@ParticipantID, @StartDate, @EndDate, @GoalID, @Preferences)";
                                using (SqlCommand insertCommand = new
SqlCommand(insertQuery, connection))
```

```
                                {
insertCommand.Parameters.AddWithValue("@ParticipantID", participant.ParticipantID);
                                    insertCommand.Parameters.AddWithValue("@StartDate",
DateTime.Now.Date);
                                    insertCommand.Parameters.AddWithValue("@EndDate",
DateTime.Now.Date.AddMonths(1));
                                    insertCommand.Parameters.AddWithValue("@GoalID",
participant.GoalID);

insertCommand.Parameters.AddWithValue("@Preferences", DBNull.Value);

                                    insertCommand.ExecuteNonQuery();
                                }

                                lblSuccessMessage.Text = "Training plan generated
successfully.";
                            }
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                lblErrorMessage.Text = "Error generating training plan: " + ex.Message;
            }
        }

        private void LoadTrainingPlans(int userId)
        {
            try
            {
                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    connection.Open();
                    string query = @"
SELECT TP.[StartDate], TP.[EndDate], G.[Name] as GoalName, TP.[Preferences]
FROM [TrainingPlans] TP
INNER JOIN [Goals] G ON TP.[GoalID] = G.[GoalID]
INNER JOIN [Participant] P ON TP.[ParticipantID] = P.[ParticipantID]
WHERE P.[UserID] = @UserID";
                    using (SqlCommand command = new SqlCommand(query, connection))
                    {
                        command.Parameters.AddWithValue("@UserID", userId);
                        using (SqlDataReader reader = command.ExecuteReader())
                        {
                            DataTable dataTable = new DataTable();
                            dataTable.Load(reader);
                            gvTrainingPlans.DataSource = dataTable;
                            gvTrainingPlans.DataBind();
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                lblErrorMessage.Text = "Error loading training plans: " + ex.Message;
            }
        }

        private void LoadTrainingData(int userId)
        {
            try
```

```
            {
                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    connection.Open();
                    Participant participant = GetParticipant(userId, connection);

                    if (participant != null)
                    {
                        GeneticAlgorithm ga = new GeneticAlgorithm();
                        ga.RunGeneticAlgorithm();

                        List<PhysicalSkill> skills = ga.GetFinalPhysicalSkills()
                                                       .Where(s => s.ParticipantID ==
participant.ParticipantID)
                                                       .ToList();

                        string recommendation = GenerateRecommendation(skills,
participant);
                        lblRecommendation.Text = recommendation;
                    }
                }
            }
            catch (Exception ex)
            {
                lblErrorMessage.Text = "Error loading training data: " + ex.Message;
            }
        }

        private Participant GetParticipant(int userId, SqlConnection connection)
        {
            string query = "SELECT * FROM Participant WHERE UserID = @UserID";
            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@UserID", userId);
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    if (reader.Read())
                    {
                        return new Participant
                        {
                            ParticipantID = (int)reader["ParticipantID"],
                            Gender = reader["Gender"].ToString(),
                            Weight = reader["Weight"].ToString(),
                            Height = reader["Height"].ToString(),
                            Age = reader["Age"].ToString(),
                            GoalID = (int)reader["GoalID"]
                        };
                    }
                }
            }
            return null;
        }

        private string GenerateRecommendation(List<PhysicalSkill> skills, Participant
participant)
        {
            string recommendation = "";
            var lowSkills = skills.Where(s => s.SkillLevelID <= 2).ToList();
            var highSkills = skills.Where(s => s.SkillLevelID >= 4).ToList();

            if (lowSkills.Any(s => s.SkillNameID == 1))
                recommendation += "За вашими даними, рекомендуємо більше уваги приділити
тренуванням на витривалість. ";
```

```
            if (highSkills.Any(s => s.SkillNameID == 2))
                recommendation += "Ви досягли чудового прогресу в швидкості. Час
зосередитись на силових тренуваннях! ";

            // Отримайте ціль з бази даних
            Goal goal = GetParticipantGoal(participant.GoalID);

            if (goal != null)
            {
                switch (goal.Name.ToLower())
                {
                    case "схуднення":
                        recommendation += "Оскільки ваша мета - схуднення, додайте
більше кардіотренувань. ";
                        break;
                    case "збільшення м'язової маси":
                        recommendation += "Для збільшення м'язової маси зосередьтеся на
силових тренуваннях з важкою вагою і більшою кількістю підходів. ";
                        break;
                    case "підвищення витривалості":
                        recommendation += "Щоб підвищити витривалість, виконуйте довгі
велосипедні прогулянки на помірній інтенсивності. ";
                        break;
                        // Додайте інші цілі за потреби
                }
            }

            return recommendation.Trim();
        }

        private Goal GetParticipantGoal(int goalId)
        {
            string query = "SELECT GoalID, Name, Description FROM Goals WHERE GoalID =
@GoalID";

            using (SqlCommand command = new SqlCommand(query, new
SqlConnection(connectionString)))
            {
                command.Parameters.AddWithValue("@GoalID", goalId);
                command.Connection.Open();

                using (SqlDataReader reader = command.ExecuteReader())
                {
                    if (reader.Read())
                    {
                        return new Goal
                        {
                            GoalID = reader.GetInt32(0),
                            Name = reader.IsDBNull(1) ? null : reader.GetString(1),
                            Description = reader.IsDBNull(2) ? null :
reader.GetString(2)
                        };
                    }
                }
            }

            return null;
        }

        protected void btnDeletePlan_Click(object sender, EventArgs e)
        {
            if (Session["UserID"] != null)
```

```
            {
                int userId = Convert.ToInt32(Session["UserID"]);
                DeleteCurrentPlan(userId);
                LoadTrainingPlans(userId);
            }
            else
            {
                Response.Redirect("Login.aspx");
            }
        }

        private void DeleteCurrentPlan(int userId)
        {
            try
            {
                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    connection.Open();
                    int participantId = GetParticipantIdByUserId(userId, connection);

                    if (participantId > 0)
                    {
                        string query = @"
DELETE FROM TrainingPlans
WHERE ParticipantID = @ParticipantID
  AND EndDate >= @CurrentDate";

                        using (SqlCommand command = new SqlCommand(query, connection))
                        {
                            command.Parameters.AddWithValue("@ParticipantID",
participantId);
                            command.Parameters.AddWithValue("@CurrentDate",
DateTime.Now.Date);

                            int rowsAffected = command.ExecuteNonQuery();

                            if (rowsAffected > 0)
                            {
                                lblSuccessMessage.Text = "Поточний план тренувань було
успішно видалено.";
                            }
                            else
                            {
                                lblSuccessMessage.Text = "Не знайдено активних планів
тренувань для видалення.";
                            }
                        }
                    }
                    else
                    {
                        lblErrorMessage.Text = "Не вдалося знайти учасника для поточного
користувача.";
                    }
                }
            }
            catch (Exception ex)
            {
                lblErrorMessage.Text = "Помилка при видаленні плану тренувань: " +
ex.Message;
            }
        }

        private int GetParticipantIdByUserId(int userId, SqlConnection connection)
```

```
        {
            string query = "SELECT ParticipantID FROM Participant WHERE UserID =
@UserID";
            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@UserID", userId);
                object result = command.ExecuteScalar();
                return result != null ? Convert.ToInt32(result) : -1;
            }
        }
    }
}
```

Лістинг TrainingProgress.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Diagnostics;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using BikeClub.Algorithm;

namespace BikeClub
{
    public partial class TrainingProgress : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                if (Session["UserID"] != null)
                {
                    int userId = Convert.ToInt32(Session["UserID"]);
                    BindGridView(userId);
                }
                else
                {
                    // Перенаправлення на сторінку входу, якщо користувач не увійшов
                    Response.Redirect("Login.aspx");
                }
            }
        }

        private void BindGridView(int userId)
        {
            string connectionString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Admin\source\repos\BikeClub\Bike
Club\App_Data\Database1.mdf;Integrated Security=True";

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();

                string query = @"
            SELECT p.Age AS ParticipantAge, sn.Name AS SkillName, sl.Name AS SkillLevel
            FROM Participant p
            INNER JOIN PhysicalSkills ps ON p.ParticipantID = ps.ParticipantID
            INNER JOIN SkillName sn ON ps.SkillNameID = sn.SkillNameID
            INNER JOIN SkillLevel sl ON ps.SkillLevelID = sl.SkillLevelID
```

```
            WHERE p.UserID = @UserID";

                using (SqlCommand command = new SqlCommand(query, connection))
                {
                    command.Parameters.AddWithValue("@UserID", userId);

                    SqlDataAdapter adapter = new SqlDataAdapter(command);
                    DataTable dataTable = new DataTable();
                    adapter.Fill(dataTable);

                    gvTrainingProgress.DataSource = dataTable;
                    gvTrainingProgress.DataBind();
                }
            }
        }
    }
}
```

Лістинг UpdatePhysicallSkill.aspx.cs

```
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using BikeClub.Algorithm;
using System;
using System.Diagnostics;
using System.Configuration;
using BikeClub.Algorithm;
using BikeClub.Data;
using System.Runtime.InteropServices;
using System.Web.Http;
using System.Web.Http.Results;


namespace BikeClub
{
    public partial class UpdatePhysicalSkills : System.Web.UI.Page
    {
        private string connectionString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Admin\source\repos\BikeClub\Bike
Club\App_Data\Database1.mdf;Integrated Security=True";
        private SkillLevelRepository skillLevelRepository;

        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                DisplayParticipantInfo();
                LoadSkillLevels();
                LoadPhysicalSkills();
                LoadGoals();
            }
        }

        private void BindSkillLevelsToDropDown()
        {
            foreach (GridViewRow row in gvPhysicalSkills.Rows)
            {
                DropDownList ddlSkillLevel =
```

```
(DropDownList)row.FindControl("ddlSkillLevel");
                if (ddlSkillLevel != null)
                {
                    ddlSkillLevel.DataSource = skillLevelRepository.GetSkillLevels();
                    ddlSkillLevel.DataTextField = "Name";
                    ddlSkillLevel.DataValueField = "SkilllevelID";
                    ddlSkillLevel.DataBind();
                    ddlSkillLevel.Items.Insert(0, new ListItem("-- Виберіть рівень
навички --", ""));

                    // Встановлюємо SelectedValue на основі даних з бази даних
                    PhysicalSkill skill = (PhysicalSkill)row.DataItem;
                    ddlSkillLevel.SelectedValue = skill.SkillLevelID?.ToString() ?? "";
                }
            }
        }

        private void DisplayParticipantInfo()
        {
            // Зчитуємо інформацію про учасника з сесії і відображаємо на сторінці
            User user = (User)Session["User"];
            if (user != null)
            {
                dvParticipant.DataSource = new List<User> { user };
                dvParticipant.DataBind();
            }
        }

        private void LoadSkillLevels()
        {
            skillLevelRepository = new SkillLevelRepository(connectionString);
            List<SkillLevel> skillLevels = skillLevelRepository.GetSkillLevels();
        }

        private void LoadPhysicalSkills()
        {
            int participantID = GetParticipantIDFromSession();
            List<PhysicalSkill> physicalSkills = new List<PhysicalSkill>();
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                physicalSkills = GetPhysicalSkillsForParticipant(participantID,
connection);
            }
            gvPhysicalSkills.DataSource = physicalSkills;
            gvPhysicalSkills.DataBind();

            // Зв'язуємо дані з ddlSkillLevel після привязки даних до GridView
            BindSkillLevelsToDropDown();
        }

        private List<PhysicalSkill> GetPhysicalSkillsForParticipant(int participantID,
SqlConnection connection)
        {
            List<PhysicalSkill> physicalSkills = new List<PhysicalSkill>();
            string query = "SELECT ps.*, sl.Name as SkillLevelName FROM PhysicalSkills
ps LEFT JOIN SkillLevel sl ON ps.SkillLevelID = sl.SkilllevelID WHERE ps.ParticipantID =
@ParticipantID";
            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@ParticipantID", participantID);
                using (SqlDataReader reader = command.ExecuteReader())
                {
```

```
                    while (reader.Read())
                    {
                        int? skillLevelID = reader["SkillLevelID"] as int?;
                        if (skillLevelID.HasValue &&
!string.IsNullOrEmpty(reader["SkillLevelName"] as string))
                        {
                            PhysicalSkill skill = new PhysicalSkill
                            {
                                SkillID = Convert.ToInt32(reader["SkillID"]),
                                SkillNameID = Convert.ToInt32(reader["SkillNameID"]),
                                SkillLevelID = skillLevelID,
                                ParticipantID = participantID
                            };
                            physicalSkills.Add(skill);
                        }
                    }
                }
            }
            return physicalSkills;
        }

        private void UpdateSkills()
        {
            int participantID = GetParticipantIDFromSession();
            List<PhysicalSkill> updatedSkills = new List<PhysicalSkill>();
            foreach (GridViewRow row in gvPhysicalSkills.Rows)
            {
                int skillID = Convert.ToInt32(row.Cells[0].Text);
                string skillLevelID =
((DropDownList)row.FindControl("ddlSkillLevel")).SelectedValue;
                PhysicalSkill skill = new PhysicalSkill
                {
                    SkillID = skillID,
                    SkillLevelID = Convert.ToInt32(skillLevelID),
                    ParticipantID = participantID
                };
                updatedSkills.Add(skill);
            }
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                UpdatePhysicalSkillsForParticipant(participantID, updatedSkills,
connection);
            }
            LoadPhysicalSkills();
        }

        private void UpdatePhysicalSkillsForParticipant(int participantID,
List<PhysicalSkill> updatedSkills, SqlConnection connection)
        {
            foreach (PhysicalSkill skill in updatedSkills)
            {
                string query = "UPDATE PhysicalSkills SET SkillLevelID = @SkillLevelID
WHERE SkillID = @SkillID AND ParticipantID = @ParticipantID";
                using (SqlCommand command = new SqlCommand(query, connection))
                {
                    command.Parameters.AddWithValue("@SkillLevelID",
skill.SkillLevelID);
                    command.Parameters.AddWithValue("@SkillID", skill.SkillID);
                    command.Parameters.AddWithValue("@ParticipantID", participantID);
                    command.ExecuteNonQuery();
                }
            }
```

```
        }

        private void LoadGoals()
        {
            // Завантажуємо цілі з бази даних (припускаючи, що цілі зберігаються у
відповідній таблиці)
        }

        protected void UpdateButton_Click(object sender, EventArgs e)
        {
            UpdateSkills();
            lblMessage.Text = "Навички успішно оновлені!";
        }

        protected void gvPhysicalSkills_RowEditing(object sender, GridViewEditEventArgs
e)
        {
            gvPhysicalSkills.EditIndex = e.NewEditIndex;
            LoadPhysicalSkills();
        }

        protected void gvPhysicalSkills_RowUpdating(object sender,
GridViewUpdateEventArgs e)
        {
            int rowIndex = e.RowIndex;
            int skillID =
Convert.ToInt32(gvPhysicalSkills.DataKeys[rowIndex].Values["SkillID"]);
            int participantID = GetParticipantIDFromSession();

            try
            {
                // Отримати нове значення SkillLevelID з параметру елемента GridView
                int skillLevelID = Convert.ToInt32(e.NewValues["SkillLevelID"]);

                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    connection.Open();
                    string query = "UPDATE PhysicalSkills SET SkillLevelID =
@SkillLevelID WHERE SkillID = @SkillID AND ParticipantID = @ParticipantID";
                    using (SqlCommand command = new SqlCommand(query, connection))
                    {
                        command.Parameters.AddWithValue("@SkillLevelID", skillLevelID);
                        command.Parameters.AddWithValue("@SkillID", skillID);
                        command.Parameters.AddWithValue("@ParticipantID",
participantID);
                        command.ExecuteNonQuery();
                    }
                }
                gvPhysicalSkills.EditIndex = -1;
                LoadPhysicalSkills();
            }
            catch (Exception ex)
            {
                // Обробка помилок, наприклад, вивід на екран або журналування
                lblMessage.Text = "Помилка під час оновлення фізичних навичок: " +
ex.Message;
            }
        }

        protected void gvPhysicalSkills_RowCancelingEdit(object sender,
GridViewCancelEditEventArgs e)
        {
            gvPhysicalSkills.EditIndex = -1;
```

```
            LoadPhysicalSkills();
        }

        protected void btnRunGeneticAlgorithm_Click(object sender, EventArgs e)
        {
            GeneticAlgorithm geneticAlgorithm = new GeneticAlgorithm();
            geneticAlgorithm.RunGeneticAlgorithm();
        }

        private int GetParticipantIDFromSession()
        {
            return 1; // Повертаємо ідентифікатор учасника, наприклад
        }

    }

}
```

Лістинг ViewTrainingPlan.aspx.cs

```
using BikeClub.Data;
using System.Collections.Generic;
using System.Data.SqlClient;
using System;
using BikeClub.Data;
using TrainingPlanData = BikeClub.Data.TrainingPlan;


namespace BikeClub
{
    public partial class ViewTrainingPlan : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                // Підключення до бази даних
                SqlConnection connection = new SqlConnection("Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\Admin\\source\\repos\\BikeClu
b\\BikeClub\\App_Data\\Database1.mdf;Integrated Security=True");
                connection.Open();

                // Створення екземпляра TrainingPlanGenerator
                TrainingPlanGenerator planGenerator = new
TrainingPlanGenerator(connection);

                // Отримання даних про учасника (наприклад, з URL-параметрів або з
іншого джерела)
                int participantId = 1; // Приклад: отримання ID учасника
                Participant participant = GetParticipantDetails(connection,
participantId);

                // Генерація плану тренувань
                planGenerator.GeneratePlan(participant);

                // Отримання згенерованого плану тренувань
                List<TrainingPlanData> trainingPlans = GetTrainingPlans(connection,
participant.ParticipantID);

                // Відображення плану тренувань на сторінці
                TrainingPlanGridView.DataSource = trainingPlans;
                TrainingPlanGridView.DataBind();
```

```
                // Закриття підключення до бази даних
                connection.Close();
            }
        }

        // Функція для отримання деталей учасника
        private Participant GetParticipantDetails(SqlConnection connection, int
participantId)
        {
            string query = "SELECT ParticipantID, GoalID FROM Participant WHERE
ParticipantID = @ParticipantID";
            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@ParticipantID", participantId);
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    if (reader.Read())
                    {
                        return new Participant
                        {
                            ParticipantID = (int)reader["ParticipantID"],
                            GoalID = (int)reader["GoalID"]
                        };
                    }
                }
            }
            return null;
        }

        // Функція для отримання планів тренувань для учасника
        private List<BikeClub.Data.TrainingPlan> GetTrainingPlans(SqlConnection
connection, int participantId)
        {
            List<BikeClub.Data.TrainingPlan> plans = new
List<BikeClub.Data.TrainingPlan>();
            string query = "SELECT PlanID, ParticipantID, StartDate, EndDate, GoalID,
Preferences FROM TrainingPlans WHERE ParticipantID = @ParticipantID";
            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@ParticipantID", participantId);
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        plans.Add(new BikeClub.Data.TrainingPlan
                        {
                            PlanID = (int)reader["PlanID"],
                            ParticipantID = (int)reader["ParticipantID"],
                            StartDate = (DateTime)reader["StartDate"],
                            EndDate = (DateTime)reader["EndDate"],
                            GoalID = (int)reader["GoalID"],
                            Preferences = reader["Preferences"].ToString()
                        });
                    }
                }
            }
            return plans;
        }
    }
}
```

Додаток Г

Презентаційний матеріал

Кваліфікаційна робота бакалавра

МЕТОД ПЛАНУВАННЯ РОЗВИТКУ ФІЗИЧНИХ НАВИЧОК УЧАСНИКІВ ЗА ДОПОМОГОЮ ГЕНЕТИЧНОГО АЛГОРИТМУ ДЛЯ ВЕБ-СИСТЕМИ ВЕЛОКЛУБУ

Виконав:
студент 4 курсу, групи КН-20-2
Максим Ґоляж

Керівник:
Доцент кафедри КН
Олександр Пасічник

02

АКТУАЛЬНІСТЬ

У сучасному світі, де технології стрімко розвиваються, а спосіб життя стає все більш урбанізованим та сидячим, велосипедний спорт набуває все більшої популярності як ефективний засіб підтримання фізичної форми та здорового способу життя. Велоклуби, які об'єднують любителів активного відпочинку, стають місцем зустрічі однодумців та майданчиком для обміну досвідом і вдосконалення своїх навичок.

03

ГЕНЕТИЧНИЙ АЛГОРИТМ В СПОРТІ?

Враховуючи складність організації та планування розвитку фізичних навичок членів велоклубу, доцільно використовувати генетичний алгоритм. Такий підхід можна ефективно інтегрувати через веб-систему, що значно полегшить управління тренувальними процесами.

04

ВИМОГИ ДО РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Розробити веб-систему для ефективного планування та відстеження розвитку фізичних навичок учасників велоклубу шляхом застосування генетичного алгоритму. Система повинна генерувати персоналізовані плани тренувань, враховуючи індивідуальні цілі, рівень підготовки та медичні показання учасників, і забезпечувати зручний інтерфейс для моніторингу прогресу.

05

ЗАГАЛЬНА СТРУКТУРА

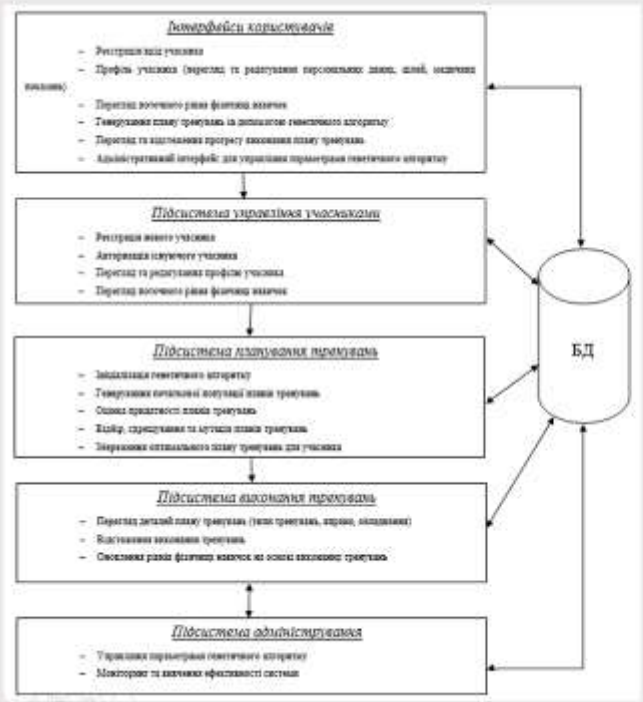
Функціональна структура інформаційної системи була ретельно спроектована з урахуванням вимог та особливостей предметної області. Були визначені основні модулі та компоненти системи, а також їх взаємозв'язки та способи взаємодії.

05

АРХІТЕКТУРА НЕЙРОННОЇ МЕРЕЖІ

Для вирішення задачі було розроблено архітектуру нейронної мережі, яка враховувала специфіку вхідних даних та забезпечувала необхідну точність та ефективність моделі.

05



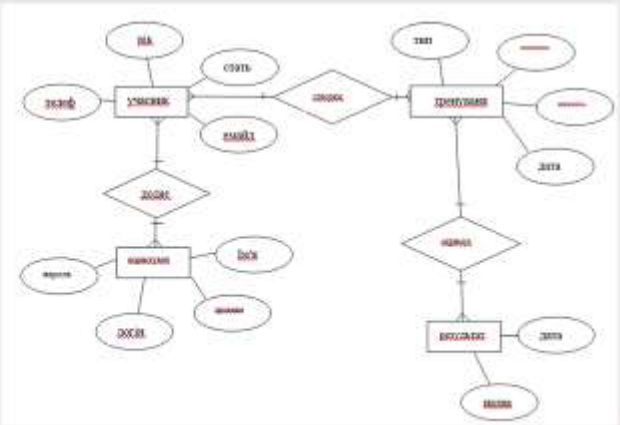
ЗАГАЛЬНА СТРУКТУРА

Проектна архітектура системи була спроектована з урахуванням сучасних підходів до розробки програмного забезпечення та вимог до масштабованості, надійності та гнучкості системи. Було визначено взаємозв'язок компонентів та інтерфейси їх взаємодії.

05

ІНФОРМАЦІЙНА СТРУКТУРА

Інформаційна структура системи була спроектована відповідно до вимог предметної області та з урахуванням особливостей обробки та зберігання даних. Було розроблено схему бази даних програмної системи, яка забезпечує ефективне зберігання та доступ до даних.



06

ВИБІР ЗАСОБІВ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

ПЛАТФОРМА
ASP.NET WEB FORMS

МОВА ПРОГРАМУВАННЯ
C#

СЕРЕДОВИЩЕ ПРОГРАМУВАННЯ
VISUAL STUDIO

КЕРУВАННЯ БАЗАМИ ДАНИХ
SQL SERVER

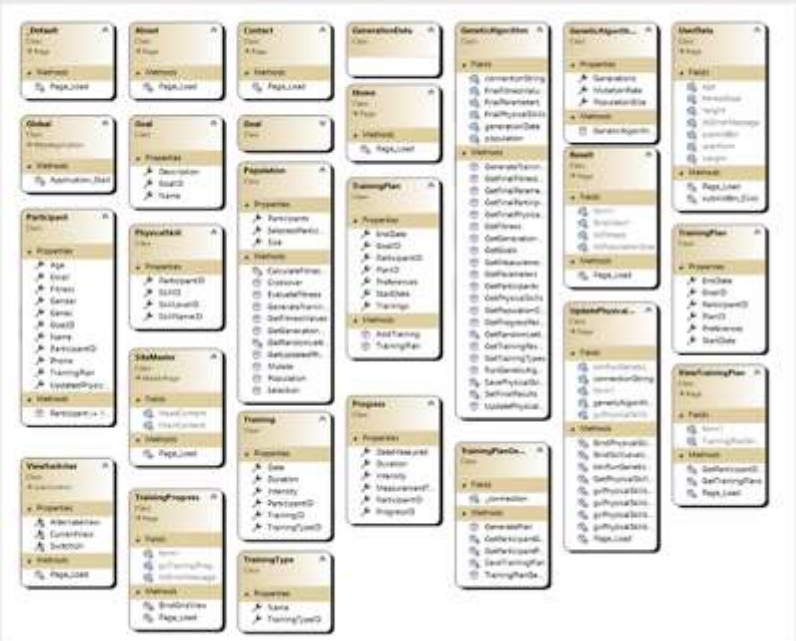
МОВА ЗАПИТІВ
T-SQL

ВЕБ СЕРВЕР
INTERNET INFORMATION SERVICES

07

ДІАГРАМА КЛАСІВ

Застосунок використовує об'єктно-орієнтований підхід і складається з декількох взаємопов'язаних класів, які відповідають за різні аспекти функціональності додатку.




08
ДЯКУЮ ЗА
УВАГУ!

Tue Jun 18 21:58:15 EEST 2024, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 7%**

ID: 131421 Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА на тему Метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для вебсистеми велоклубу Додано в БД: 2024-06-18 Автора: Максим ҐОЛЯЖ Керівники: Олександр ПАСІЧНИК Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	89879	1325	4099 (5%)	51 (4%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

Ім'я користувача:
Кафедра КН

ID перевірки:
1016373379

Дата перевірки:
18.06.2024 22:19:16 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
18.06.2024 22:40:56 EEST

ID користувача:
100005671

Назва документа: **КН-20-2 Ґоляж_ ЗАПИСКА**

Кількість сторінок: **83** Кількість слів: **13255** Кількість символів: **110866** Розмір файлу: **2.85 MB** ID файлу: **1016180866**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

5.48%
Схожість

Найбільша схожість: **3.15%** з джерелом з Бібліотеки (ID файлу: **1016162162**)

3.29% Джерела з Інтернету	445	Сторінка 55
4.65% Джерела з Бібліотеки	172	Сторінка 33

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	17 сторінок

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *Метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб-системи велоклубу*

Автор: *студент групи КН-20-2 Максим Ґоляж*

Спеціальність: *122 – Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *к.т.н., доцент Олександр Пасічник*

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	***відповідає***
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі Максима Ґоляжа, не є плагіатом, оскільки: запозичення розміщені в розділі огляду існуючих підходів, не описують безпосередньо авторську роботу і не стосуються її результатів; усі запозичення фрагментарні; до запозичень входять фрагменти програмного коду, що не мають авторства і містять поширені конструкції; серед запозичень знаходяться загальновідомі терміни, скорочення та визначення.

Обсяг запозичень, визначений системами виявлення збігів/ідентичності/схожості, складає:

- за системою Anti-Plagiarism: 2%;

- за системою Unicheck: 5,48 %,

що є допустимими запозиченнями.

Керівник роботи *Олександр ПАСІЧНИК*

Гарант ОП *Олександр МАЗУРЕЦЬ*

Завідувач кафедри КН *Олександр БАРМАК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КН-20-2 Максима Голяжа*

за темою *Метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб-системи*

1. Актуальність теми

Однією з областей, де комп'ютерні науки демонструють свій значний потенціал, є спорт. Сьогодні спортсмени та тренери використовують передові технології для аналізу даних, моніторингу продуктивності та вдосконалення тренувальних програм. Одним з найперспективніших напрямків у комп'ютерних науках є застосування генетичних алгоритмів - методу оптимізації, натхненного процесами природного відбору та генетики. Велоспорт - це вид спорту, який особливо виграє від інтеграції комп'ютерних наук. Участь у велосипедних змаганнях вимагає високого рівня витривалості, сили та тактичного мислення. Тому розробка оптимального плану тренувань для велосипедиста - це складне завдання, яке вимагає врахування багатьох факторів.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

За стандартом, а саме описом предметної області, об'єктами вивчення та діяльності є математичні, інформаційні, імітаційні моделі реальних явищ, об'єктів, систем і процесів та методи і технології отримання, зберігання, обробки, передачі та використання інформації. Метою роботи саме є розробка методу планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб-системи. При вирішенні поставленої задачі використано відповідні математичні моделі, методи та алгоритми розв'язання теоретичних і прикладних задач. Тому результати виконання кваліфікаційної роботи бакалавра відповідають стандарту бакалавра спеціальності 122 – Комп'ютерні науки.

3. Професійні та особистісні якості бакалавра

При роботі над кваліфікаційною роботою бакалавра Безпалий Сергій проявив себе кваліфікованим фахівцем та дисциплінованим студентом, вчасно виконуючи поставлені етапи дослідження. Як в процесі написання пояснювальної записки, так і при розробці методу планування розвитку фізичних навичок учасників за допомогою генетичного

алгоритму для веб-системи та його програмній реалізації, тестування та експериментальній перевірці проявив достатні для одержання успішного результату компетентності та результати навчання. Опанував професійні скіли за напрямком «Комп'ютерні науки» та достатньо значний софт скіл .

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Одержані в роботі результати є наслідком особистої діяльності студента, який самостійно виконував всі поставлені задачі.

5. Ступінь оволодіння методами дослідження

При реалізації кваліфікаційної роботи показав достатній рівень компетентностей та володіння необхідними інструментами та обладнанням, методами, методиками та технологіями предметної області комп'ютерних наук.

6. Повнота та якість розкриття теми роботи

Тема роботи в повній мірі обґрунтована й розкрита, проведено аналіз актуальності та відомих досліджень в межах обраної теми, поставлені завдання, які у роботі виконані, та виконано програмну реалізацію для валідації та верифікації запропонованого методу.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Структура роботи та послідовність викладення логічні та відповідають поставленій меті. Викладення матеріалу послідовне, аргументоване, літературно грамотне.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Розроблений у роботі метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб-системи метод та його програмна реалізація може бути використана для покращення стану фізичної культури широких прошарків населення.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

Враховуючи високий рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка «задовільно» .

Керівник _____ *к.т.н., доцент каф. КН Олександр ПАСІЧНИК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КН-20-2 Максима Ґоляжа*
за темою: *Метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб-системи вело клубу.*

1. Актуальність обраної теми

У сучасному світі, де технології невпинно розвиваються, комп'ютерні науки стали невід'ємною частиною нашого повсякденного життя. Їхній вплив поширюється на різноманітні сфери, від комунікацій та освіти до охорони здоров'я та спорту. Однією з областей, де комп'ютерні науки демонструють свій значний потенціал, є спорт. Сьогодні спортсмени та тренери використовують передові технології для аналізу даних, моніторингу продуктивності та вдосконалення тренувальних програм, зокрема у велоспорті.

2. Повнота розкриття мети та завдань роботи

Мета роботи розкрита повністю, всі завдання виконані.

3. Зміст кожного розділу роботи

В першому розділі виконано аналіз сучасних підходів до планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб-системи велоклубу. Визначено мету роботи та виконано постановку завдань. В другому розділі реалізовано метод планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб-системи велоклубу. Визначено критерії оцінки покращення. В третьому розділі виконано програмну реалізацію методу та її тестування. Виконано оцінку покращення.

4. Оцінка розробленої інформаційної системи, її практична цінність

Розроблений планування розвитку фізичних навичок учасників за допомогою генетичного алгоритму для веб-системи велоклубу може бути використаний в системі фізичного виховання дітей та дорослих.

5. Якість оформлення кваліфікаційної роботи бакалавра

Структура роботи та послідовність викладення логічні та відповідають поставленій меті. Викладення матеріалу послідовне, аргументоване, літературно грамотне.

6. Недоліки кваліфікаційної роботи бакалавра

В роботі використано певні критерії для оцінки досягнення мети, але не зазначено чи є цей перелік вичерпним.

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

Враховуючи рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка «задовільно».

Рецензент *к.т.н., доц. каф. КІІС Пісторук Л.О.* _____