

Детектор Плагиата v. 2867 - Отчёт оригинальности: 05.06.2025 10:03:33

Проанализированный документ: Бак_пояснювальна_Озеров Даниїл Віталійович – копія.docx
Лицензия: ВОЛОДИМИР МАТІЄВСЬКИЙ

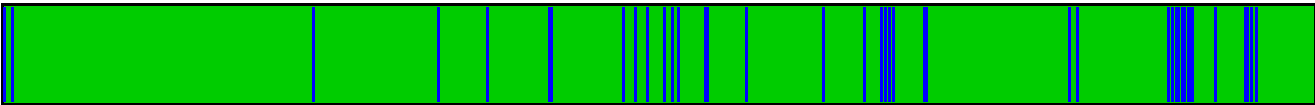
Тип поиска: Поиск переписанного Язык: Uk
Тип проверки: Интернет
ТЕЕ и кодировка: DocX n/a

Детальный анализ тела документа:

Диаграмма соотношения частей:



Граф распределения зон:



Источники плагиата: 3

	→ 0,2%	10	1. https://metod.vntu.edu.ua/getfile.php/6467.pdf
	→ 0,1%	5	2. https://krs.chmnu.edu.ua/jspui/bitstream/123456789/3279/1/KPM_Постриган_О_О_608м_.pdf
	→ 0,1%	5	3. https://fizmat.7mile.net/informatika-10/06-tablici-v-dokumentah.htm

Детали обработанных ресурсов: 180 - ОК / 7 - Ошибка

Важные замечания:

Википедия:	Google Книги:	Сервисы платных работ:	Античит:
[не обнаружено]	[не обнаружено]	[не обнаружено]	Обнаружено соккрытие!

Античит-отчет UACE:

1. Статус: Анализатор Включен Нормализатор Включен сходство символов установлено на 100%
2. Обнаруженный процент загрязнения UniCode: 14% с лимитом: 4%
3. Процент нераспознанных символов после нормализации: 8%
4. Все подозрительные символы будут отмечены фиолетовым цветом: <u>Abcd...</u>
5. Найдены невидимые символы: 0
Рекомендации по оценке: Особое внимание следует уделить анализу этого отчета! Предполагается, что этот документ содержит значительное количество символов, чуждых языку документа. Это прямое указание на то, что автор документа использовал специальное программное обеспечение\онлайн-веб-сервис, чтобы эффективно

скрыть текст в попытке избежать обнаружения потенциального плагиата. Настоятельно рекомендуется передать это дело на более высокий уровень! В случае сомнений обращайтесь: в службу поддержки Детектора плагиата!

Алфавитная статистика и анализ символов:

 Активные ссылки (URL-адреса, извлеченные из документа):

URL не найдены

 Исключённые ресурсы:

URL не найдены

 Включённые ресурсы:

URL не найдены

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ДЗ

Цитування: 0,07%

id: 1

«ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА»

Навчально-науковий інститут математики та інформаційних технологій (назва факультету, інституту) Кафедра інформаційних технологій та систем (назва кафедри) Пояснювальна записка до кваліфікаційної роботи за першим (бакалаврським) рівнем освіти на тему: РОЗРОБКА [WINDOWS FORMS](#)-ДОДАТКУ ДЛЯ ПЛАНУВАННЯ ТА ОБЛІКУ СПОРТИВНИХ ТРЕНУВАНЬ Виконав: здобувач вищої освіти 4 курсу спеціальності 121

Цитування: 0,04%

id: 2

«Інженерія програмного забезпечення»

(шифр і назва напрямку підготовки, спеціальності) _____ Данііл ОЗЕРОВ (прізвище та ініціали) Керівник _____ Микола СЕМЕНОВ (прізвище та ініціали) Рецензент _____ Владислав ІЩЕНКО (прізвище та ініціали) Полтава – 2025 ЗМІСТ ВСТУП4 РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОГРАМНИХ РІШЕНЬ6 1.1. Особливості організації індивідуального тренувального процесу та потреби у цифрових інструментах6 1.2. Огляд та аналіз існуючих алгоритмів, методик і схем тренувань11 1.3. Порівняльний аналіз функціоналу, переваг і недоліків аналогів18 1.4. Вимоги до локального [Windows](#)-додатку для обліку фізичної активності26 РОЗДІЛ 2. ПРОЄКТУВАННЯ МОДЕЛІ [WINDOWS FORMS](#)-ДОДАТКУ30 2.1. Архітектура програмного забезпечення: головні компоненти та їх зв'язки30 2.2. Концептуальна модель бази даних: таблиці, зв'язки, логіка збереження34 2.3. Розробка інтерфейсної структури додатку: головне меню, форма тренування, статистика41 2.4. Вибір технологій: [C#](#), [Windows Forms](#), [SQLite](#), додаткові бібліотеки46 РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ48 3.1. Реалізація базових функцій: створення, редагування та перегляд тренувань48 3.2. Реалізація аналітики: побудова графіків прогресу (час, дистанція, навантаження)56 3.3. Експорт даних у звіт (формати [Excel](#), [PDF](#))60 3.4. Тестування, перевірка коректності роботи, приклади використання63 ЗАГАЛЬНІ ВИСНОВКИ65 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ67 ДОДАТКИ70 ВСТУП У сучасному світі дедалі більше людей веде активний спосіб життя, займається фітнесом або професійним спортом. Цифрові технології відіграють важливу роль у підвищенні ефективності тренувального процесу завдяки можливостям обліку, планування та аналізу фізичної активності. Проте наявні програмні продукти здебільшого орієнтовані на мобільні платформи, вимагають підключення до інтернету або не дозволяють гнучко адаптувати систему під потреби окремого користувача. Проблема полягає в тому, що не існує універсального офлайн-інструменту для персонального обліку тренувань, який би був простим у використанні, мав зручний інтерфейс та дозволяв здійснювати базовий аналіз спортивних показників. Особливо це актуально для тренерів, викладачів фізичного виховання та спортсменів-початківців, які працюють у локальному середовищі без постійного доступу до мережі. Питанням цифрового супроводу фізичної активності присвячено роботи як у сфері фітнес-технологій, так і в галузі освітньої інформатики. Зокрема, дослідження у напрямі персоналізованих спортивних застосунків проводились у працях [R. Bartlett](#) та інших. Однак, питання реалізації таких систем у форматі офлайн-додатків для ПК на базі [Windows Forms](#) залишається малодослідженим. Вибір теми обумовлено потребою створення зручного інструменту для планування та аналізу тренувань у настільному середовищі [Windows](#) з можливістю подальшого розширення функціоналу. Платформа [Windows Forms](#) обрана через швидкість розробки, наявність великої бази компонентів і підтримку інтеграції з базами даних. Мета роботи – розробити настільний програмний застосунок для [Windows](#) на базі [Windows Forms](#) для обліку та планування спортивних тренувань з можливістю збереження результатів, візуалізації даних та формування звітів. Завдання дослідження: Проаналізувати існуючі рішення та вимоги до систем обліку спортивних тренувань. Розробити концептуальну модель додатку, включаючи базу даних, інтерфейс і логіку. Реалізувати функціональний прототип програми з використанням [C#](#) та [Windows Forms](#). Провести тестування та оцінити зручність використання програмного продукту. Об'єкт дослідження – програмні засоби підтримки фізичної активності. Предмет дослідження – методи та засоби створення [Windows Forms](#)-застосунків для планування спортивних тренувань. У першому розділі роботи подано

аналіз програмних рішень у сфері планування фізичної активності, виявлено їх сильні та слабкі сторони. Другий розділ присвячено проектуванню архітектури майбутнього застосунку: структури даних, інтерфейсу, логіки взаємодії. У третьому розділі розглянуто процес реалізації, описано основні форми програми, механізми збереження та виведення даних, а також наведено приклади роботи з програмою.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОГРАМНИХ РІШЕНЬ

Особливості організації індивідуального тренувального процесу та потреби у цифрових інструментах

Організація тренувального процесу у спорті є складним завданням, яке вимагає систематичного підходу, ретельного планування і контролю за результатами. Незалежно від рівня підготовленості спортсмена, ключовими елементами успішного тренування є чітке визначення цілей, регулярність занять, контроль фізичного навантаження, а також моніторинг прогресу та корекція навантаження залежно від отриманих результатів. У традиційному форматі тренувань планування зазвичай здійснюється тренером або самостійно спортсменом із використанням записів у щоденниках, таблицях або журналах тренувань. Традиційні методи планування та ведення обліку спортивних тренувань ґрунтуються на паперових носіях, таких як щоденники тренувань, таблиці та журнали. Попри поширення цифрових інструментів, багато спортсменів та тренерів і досі використовують ці формати завдяки їх простоті й доступності. Розглянемо основні формати паперових інструментів для тренування.

Щоденник тренувань є найпоширенішим форматом ведення обліку є паперовий або друкований щоденник тренувань, у якому спортсмен занотовує такі дані: дата тренування; тип фізичної активності (біг, силові вправи тощо); тривалість тренування; інтенсивність (наприклад, пульс або суб'єктивні відчуття); самопочуття та інші коментарі. Приклад запису в щоденнику приведено в табл. 1.1:

Дата	Вид тренування	Тривалість	Інтенсивність (пульс)	Самопочуття	Коментар
15.03.2025	Біг	45 хв	145 уд/хв	добре	Погода сприятлива
17.04.2025	Силові вправи	60 хв	-	трохи втомлений	Збільшив вагу на 5%

Щоденники зазвичай прості у використанні, але ускладнюють аналіз динаміки через потребу вручну переглядати записи. Іншою формою є журнали тренувань (табличні форми), які ведуться у формі таблиць (приклад – табл. 1.2), що дозволяє краще впорядкувати інформацію та полегшити візуальний аналіз.

Тиждень	Понеділок	Вівторок	Середа	Четвер	П'ятниця	Субота	Неділя
Біг	5 км	Силові	Відпочинок	Біг	6 км	Силові	Велосипед
Відпочинок	2 Біг	5,5 км	Силові	Відпочинок	Біг	6,5 км	Силові
Велосипед	Відпочинок	Такі формати	дають змогу	краще контролювати	регулярність	занять та коригувати	навантаження, однак ускладнюють

детальний аналіз показників спортсмена. Тренувальні картки є специфічним форматом, що переважно використовується тренерами у спортзалах для ведення персонального контролю. Приклад тренувальної картки приведено в табл. 1.3:

Вправа	Підхід 1	Підхід 2	Підхід 3	Примітки
Присідання	12x50 кг	10x55 кг	8x60 кг	важко
Жим лежачи	12x40 кг	10x45 кг	8x50 кг	добре
Планка	60 с	60 с	60 с	легко

Недолік тренувальних карток — відсутність зручних засобів для аналізу прогресу, оскільки необхідно постійно переглядати попередні записи вручну. Традиційні методи планування тренувань ґрунтуються на результатах досліджень у галузі фізіології, біомеханіки та спортивної медицини. Відомі роботи таких вчених як Т. Бомпа, Ю.Верхошанський, В.Платонов і Р.Бартлетт визначають основні принципи побудови тренувальних програм, особливо щодо періодизації навантажень, інтенсивності та обсягу тренувань. Зокрема, у роботах Бомпи [7] науково обґрунтовуються методи класичної періодизації, тоді як Верхошанський [17] акцентує увагу на силових і спеціалізованих тренуваннях. Плануючи тренувальний процес, необхідно враховувати цілу низку факторів, які суттєво впливають на його ефективність: регулярність – постійність занять дозволяє досягати сталого прогресу. інтенсивність – правильно підібране навантаження забезпечує адаптацію організму без перетренованості. тривалість – час тренувань повинен відповідати цілям спортсмена (витривалість, сила, гнучкість). відновлення – якісний відпочинок після тренувань важливий не менше, ніж саме тренування. індивідуалізація – кожен спортсмен має індивідуальні фізичні й фізіологічні особливості, які потрібно враховувати під час планування. Таким чином, традиційні формати планування тренувань і ведення записів мають певні переваги у простоті використання, проте суттєво обмежені в плані оперативного аналізу та довгострокового моніторингу результатів. Враховуючи ці недоліки, розробка цифрового додатку на платформі [Windows Forms](#) дозволить суттєво покращити процес планування тренувань, автоматизувати аналіз та візуалізувати динаміку спортивних досягнень, що підвищить ефективність тренувального процесу та зробить його більш керованим і прогнозованим.

Сучасні цифрові інструменти дозволяють значно спростити і покращити цей процес завдяки автоматизації збору, зберігання та аналізу даних. Зокрема, електронні засоби дозволяють зручно вести календар тренувань, аналізувати інтенсивність навантажень, виявляти тренди прогресу та попереджувати перетренованість або травми. Основними потребами користувачів при використанні таких систем є: швидкий доступ до історії тренувань із можливістю порівняння результатів; гнучкість у налаштуванні індивідуальних програм тренувань; інтуїтивно зрозумілий інтерфейс для внесення та редагування даних; можливість побудови аналітичних графіків та експорту звітів для подальшого аналізу або надання тренеру; конфіденційність даних, особливо якщо система використовується тренерами або у спортивних школах. Таким чином, виникає потреба у спеціалізованих програмах, орієнтованих на індивідуальне використання в автономному режимі, які були б простими у використанні, мали достатній набір функцій і не залежали б від постійного підключення до мережі Інтернет. З огляду на це, розробка додатку з використанням технології [Windows Forms](#) є актуальною задачею, яка дозволить спортсменам-любителям та тренерам отримати ефективний інструмент для планування і контролю тренувального процесу у локальному середовищі [Windows](#). Огляд та аналіз існуючих алгоритмів, методик і схем тренувань

Процес організації тренувань у спорті ґрунтується на різних методиках, алгоритмах та схемах, які використовують спортсмени та тренери для досягнення поставлених цілей. Ці підходи різняться залежно від виду спорту, рівня підготовки спортсмена, поставлених цілей (розвиток витривалості, сили, гнучкості тощо) та інших факторів. Класична періодизація тренувань

Однією з найпоширеніших методик є класична періодизація, яка передбачає розподіл тренувального процесу на декілька фаз: підготовчу, змагальну і перехідну. В межах кожної фази змінюється інтенсивність та обсяг навантаження відповідно до мети тренування. Формалізація алгоритму класичної періодизації (спрощений вигляд):

Початок Вибір періоду тренування (підготовчий, змагальний, перехідний)

Якщо період == підготовчий: Виконання вправ із низькою та середньою інтенсивністю, високий обсяг

Якщо період == змагальний: Виконання вправ із високою інтенсивністю, зменшений обсяг

Якщо період == перехідний: Мінімальний обсяг і низька інтенсивність

Перегляд результатів Кінець

Рис. 1.1 Блок-схема алгоритму класичної періодизації

Перевагами такого підходу є структурованість, чітка послідовність навантажень, недоліком — недостатня гнучкість до індивідуальних особливостей спортсменів. Інтервальний метод тренування широко застосовується в кардіотренуваннях (біг, плавання, велоспорт). Його ідея полягає у чергуванні інтервалів з високим і низьким навантаженням (рис. 1.2).

Рис. 1.2 Блок-схема алгоритму інтервального методу

Алгоритм інтервального тренування: Початок Задати кількість циклів (N), інтенсивність (I_1 , I_2)

Для циклу = 1 до N : Виконати інтервал високої інтенсивності (I_1 , тривалість T_1)

Виконати інтервал низької інтенсивності (I_2 , тривалість T_2)

Перегляд загальної тривалості та середньої інтенсивності Кінець

Перевага такого підходу — швидке покращення фізичних кондицій, розвиток витривалості. Недолік — можливість перетренованості при неправильному підборі інтервалів.

Спліт-тренування ([Split Training](#)) широко використовуються в силових видах спорту та фітнесі, коли тренування розподіляють за групами м'язів у різні дні тижня (рис. 1.3).

Алгоритм спліт-тренування: Початок Розподіл груп м'язів за днями тижня: День 1: Група м'язів A День 2: Група м'язів B День 3: Група м'язів C ...

Для кожного дня тренувань: Виконання вправ для заданої групи м'язів

Відпочинок наступного дня або навантаження інших груп

Повторення циклу щотижня Кінець

Рис. 1.3 Блок-схема алгоритму спліт-тренування

Серед переваг цієї методики — інтенсивне навантаження конкретних груп м'язів, кращий відпочинок для інших груп. Недоліки — потребує ретельного планування, складність для початківців. Метод прогресивного навантаження передбачає поступове збільшення навантаження від тренування до тренування за чітко визначеною прогресією (наприклад, +5% ваги щотижня).

Алгоритм прогресивного навантаження: Початок Встановлення базового навантаження (W)

Встановлення темпу приросту навантаження (P , наприклад, +5% щотижня)

Для кожного тренування: Виконати вправу з поточним навантаженням (W)

Зафіксувати результат $W = W + W * (P / 100)$

Перегляд результатів через заданий період Кінець

Рис. 1.4 Блок-схема алгоритму прогресивного навантаження

Переваги — забезпечує постійний прогрес, висока мотивація. Недоліки — ризик перевтоми або травм при неправильному підборі навантажень.

1.3. Порівняльний аналіз функціоналу, переваг і недоліків аналогів

На сучасному ринку програмного забезпечення представлено чимало продуктів, що призначені для планування, моніторингу та аналізу тренувального процесу. Більшість таких систем орієнтовані на мобільні платформи ([iOS](#), [Android](#)) або працюють у

хмарному середовищі через браузер. Проте серед них є як складні професійні рішення для тренерів і команд, так і простіші інструменти для особистого використання спортсменами-аматорами. [TrainingPeaks](https://home.trainingpeaks.com/) (<https://home.trainingpeaks.com/>) (логотип – рис. 1.5)— потужна платформа для тренерів і спортсменів, що дозволяє створювати індивідуальні плани тренувань, відстежувати прогрес, аналізувати навантаження, зберігати дані з гаджетів ([Garmin](#), [Polar](#), [Suunto](#)), вести планування та виводити аналітичні графіки (рис. 1.6, рис. 1.7 та рис. 1.8). Рис. 1.5 Логотип [TrainingPeaks](#) Рис. 1.6 Інтеграція [TrainingPeaks](#) з гаджетами Рис. 1.7 Щоденник [TrainingPeaks](#) Як бачимо з рис. 1.7 тренування розфарбовуються в залежності від відповідності результатів запланованим параметрам. Рис. 1.8 Аналітичні графіки [TrainingPeaks](#) Більшість показників у [TrainingPeaks](#) базуються на функціональному порозі (потужність, темп або частота серцевих скорочень). Функціональний поріг являє собою максимальні зусилля, які може підтримувати спортсмен протягом години без втоми. Для деяких спортсменів це буде менше години, а для інших може бути трохи довше, але важливо те, що це забезпечує корисний орієнтир, за яким можна вимірювати свої тренування. Нормалізована потужність (**NP**) або нормалізований градуваний темп (**NGP**). У велоспорті функція

Цитування: 0,02%

id: 3

«Нормалізована потужність»

надає оцінку метаболічних витрат, якби поїздка проходила в стабільному темпі, і робить акцент на стрибках під час їзди. Коефіцієнт інтенсивності (**IF**) визначає наскільки інтенсивним було це тренування відносно нашого порогу за допомогою коефіцієнта інтенсивності (**IF**), наскільки інтенсивним було тренування відносно порогу, і його можна прочитати як відсоток від 1, де 1.0 — це поріг. Якщо коефіцієнт інтенсивності тренування становив 80, то можна сказати, що це тренування на 80 відсотків від свого порогу. Загальна робота на витривалість знаходиться в діапазоні від 60 до 70 відсотків, тоді як тренування з більш важким темпом буде ближче до 80 до 90 відсотків. Крім того, залежно від тривалості вашої гонки **IF** змінюється до 105 відсотків або 1,05. Коефіцієнт інтенсивності розраховується шляхом ділення **NP** або **NGP** на функціональний поріг. Оцінка тренувального стресу (**TSS**) оцінка стресу під час тренування. Оцінка стресу під час тренування враховує як інтенсивність, так і тривалість, і надає вам повніше уявлення про те, наскільки напруженим було це тренування в загальній картині вашого тренування. [Strava](https://www.strava.com/) (<https://www.strava.com/>) — популярний застосунок серед бігунів, велосипедистів та інших спортсменів. Основна функціональність полягає у реєстрації тренувань із **GPS**, соціальній взаємодії (лайки, коментарі), побудові маршрутів та перегляді прогресу. Рис. 1.9 Логотип [STRAVA](#) Рис. 1.10 Зовнішній вигляд веб сторінки з результатами тренування Переваги застосунку [STRAVA](#) простий інтерфейс, автоматичне збереження маршрутів, елементи гейміфікації, к недолікам треба віднести: обмежена гнучкість у створенні власних планів, залежність від інтернету та **GPS**. [FitNotes](#) — **Android**-додаток для силових тренувань, який дозволяє вести журнал тренувань вручну, зберігати прогрес у таблицях і будувати прості графіки на основі даних. [FitNotes](#) — це зручний застосунок для відстеження тренувань, який допомагає спортсменам записувати свої заняття, аналізувати прогрес і покращувати тренувальний процес. Основні функції застосунку включають: - Журнал тренувань: можливість записувати кожне тренування, додаючи вправи, кількість повторень і вагу. - Аналіз прогресу: перегляд статистики та графіків для оцінки результатів. - Налаштування категорій: створення власних категорій вправ для зручності. - Експорт даних: можливість експортувати тренувальні записи для резервного копіювання або аналізу. - Гнучке налаштування: вибір кольорових тем, налаштування таймерів відпочинку та інших параметрів. Рис. 1.11 [FitNotes SportTracks](#) (<https://sporttracks.en.softonic.com/>)— це десктопний застосунок для планування, аналізу та відстеження тренувань спортсменів. Основні функції [SportTracks](#): Прогнозування продуктивності Діаграми тренувального навантаження та продуктивності обчислюють минулі та заплановані тренування у календарі, що дозволяє передбачити майбутні рівні продуктивності та коригувати фізичну форму перед змаганнями. Досягнення та рекорди Фільтрація особистих рекордів за датою та типом спорту допомагає спортсменам та тренерам аналізувати сезонні показники та виявляти приховані досягнення. Підтримка мультиспорту Інструменти аналізу дозволяють відстежувати пробіжки, запливи та велосипедні тренування. Доступна можливість маркування занять для різних видів спорту, включаючи велокрос та водні пробіжки. Синхронізація даних Автоматичне завантаження тренувань та показників розумних ваг із пристроїв [Garmin](#), [Polar](#), [Suunto](#), [Coros](#), [Wahoo](#),

[TrainerRoad](#), [Apple Watch](#), [Withings](#) та [TomTom](#). Можливість ручного завантаження даних з [Fitbit](#) та інших пристроїв. Аналіз показників Перегляд окремих параметрів, таких як темп або висота, а також комбінований аналіз декількох показників для глибшого розуміння ефективності. Інструменти для тренерів Створення календаря тренувань, аналіз та коментування занять, обмін повідомленнями та управління особистими рекордами спортсменів. [Microsoft Excel](#) та [Google Sheets](#) — це потужні інструменти для планування та аналізу тренувань спортсменів. Ось як їх можна використовувати: - створення розкладу тренувань: можна створити таблиці з датами, видами вправ, кількістю повторень та підходів. - відстеження прогресу: за допомогою графіків та діаграм можна аналізувати зміни у вазі, силових показниках та витривалості. - автоматичні розрахунки: формули допомагають швидко обчислювати середні значення, навантаження та інші параметри. - шаблони тренувань: існують готові шаблони для фітнесу та здоров'я, які можна налаштовувати під власні потреби. - спільний доступ: [Google Sheets](#) дозволяє тренерам та спортсменам працювати над планами тренувань разом у реальному часі. Ці інструменти допомагають систематизувати тренувальний процес та досягати кращих результатів. Зробимо порівняльний аналіз, результати якого представлено в табл. 1.4

Таблиця 1.4 Порівняння програмних засобів для тренування

Назва	Тип платформи	Автономність	Персоналізація	Аналітика	Недоліки
TrainingPeaks	Веб/мобільна	+	+	Платна, складна	
Strava	Мобільна	-	+	Обмежений функціонал для планів	
FitNotes	Мобільна	+	+	-	Немає десктопної версії
Excel/Sheets	Універсальна	+	+	+	Потрібні навички, ручне введення
SportTracks	Десктоп	+	+	+	Застарілий інтерфейс

Як бачимо з таблиці 1.4 існують готові рішення для тренування спортсменів та аналізу їх результатів. Основний недолік – необхідність або мати навички ([Excel](#)) або висока плата за користування. Єдиний безкоштовний аналог – [SportTracks](#). Тому робимо висновок, що мета цієї роботи має більш навчальний характер та може бути корисна для спортсменів, які хочуть мати безкоштовний нескладний засіб із спрощеним інтерфейсом.

1.4. Вимоги до локального [Windows](#)-додатку для обліку фізичної активності

У цьому параграфі сформульовані функціональні вимоги до десктопного застосунку для планування та обліку спортивних тренувань на базі [Windows Forms](#), з урахуванням попереднього аналізу, цілей користувачів і потенційного розширення інтелектуального функціоналу: Рис. 1.12

[Use Case](#) діаграма застосунку

На рис. 1.12 представлені Актори: Спортсмен, Тренер, Наставник (ШІ). Основні [use cases](#): Планувати тренування, Зберігати результати тренування, Аналізувати результати тренування. Додаткові функції включені як [use cases](#) з відповідними зв'язками

Цитування: 0,01%

id: 4

«include»

та

Цитування: 0,01%

id: 5

«extend».

На основі рис. 1.12 деталізуємо функціональні вимоги до застосунку:

- Основні функції користувача
 - Планування тренувань: можливість створення індивідуального плану тренувань із вибором типу фізичної активності (біг, силові, гнучкість тощо).
 - підтримка декількох методик тренувань: періодизація, інтервальне тренування, спліт-програма, прогресивне навантаження.
 - автоматичний перемикач методик (адаптивна модель): система пропонує методику залежно від цілей користувача, обраного рівня та історії тренувань.
 - вибір тривалості циклу тренувань (наприклад, на тиждень, місяць) з можливістю редагування графіка.
 - Ведення журналу тренувань
 - Зручна форма введення тренувальних даних: - дата; - тип і назва вправи; - тривалість; - обсяг (кількість повторень, дистанція, час); - самопочуття/рівень втоми (шкала, коментар);
 - Можливість копіювання минулих тренувань, створення шаблонів;
 - Позначення пропущених тренувань та автоматичне перенесення за бажанням користувача.
 - Перегляд та аналітика
 - Відображення тренувань у вигляді календаря (план-факт).
 - Побудова персоналізованих графіків динаміки:
 - обсяг тренувань по днях/тижнях;
 - порівняння запланованих і виконаних тренувань;
 - динаміка навантаження (пульс, повторення, вага тощо).
 - Побудова графіків для кожного типу активності окремо.
 - Експорт та збереження
 - Експорт обраного періоду у форматі: [PDF](#)-звіт (структурований журнал з графіками); [Excel](#) (для подальшої обробки);
 - Можливість створення архіву тренувань для резервного копіювання.
 - Додатковий функціонал
 - Рекомендаційний модуль (опційно – з елементами ШІ)
 - Система генерує рекомендації щодо навантажень на основі історії тренувань, самопочуття та виконання плану.
 - Пропозиції

щодо зміни типу активності або інтенсивності у разі зниження результативності або перетренованості. Аналіз факторів ефективності: частота тренувань, прогрес, втома. Візуальні індикатори ризику перетренованості. Налаштування профілю користувача Введення антропометричних даних (вік, вага, стать). Визначення мети: схуднення, набір маси, витривалість тощо. Вибір бажаної методики або довіра автоматичному режиму. Архітектурні та технічні вимоги Автономність Повна офлайн-робота: всі функції реалізовані без необхідності підключення до Інтернету. Зберігання даних у локальній базі даних (наприклад, [SQLite](#)). Простий інтерфейс [Windows Forms](#)-дизайн з доступними елементами керування. Головне меню з 4 ключовими розділами:

» Цитування: 0,01%	id: 6
«Планування»	
» Цитування: 0,01%	id: 7
«Журнал»	
» Цитування: 0,01%	id: 8
«Аналітика»	
» Цитування: 0,04%	id: 9
«Експорт / Налаштування»	

Розширення (майбутні можливості) Синхронізація з хмарними сервісами через [API](#) ([Google Drive](#), [OneDrive](#)). Мобільний клієнт або вебверсія як розширення основного ПК-додатку. Інтеграція з фітнес-трекерами ([Garmin](#), [Mi Band](#) тощо). Цей набір вимог забезпечує: зручність використання, гнучкість планування, корисну аналітику, автономну роботу, потенціал для інтелектуального супроводу тренувального процесу. РОЗДІЛ 2. ПРОЄКТУВАННЯ МОДЕЛІ [WINDOWS FORMS](#)-ДОДАТКУ 2.1. Архітектура програмного забезпечення: головні компоненти та їх зв'язки Архітектура [Windows Forms](#) додатку для планування та обліку спортивних тренувань, реалізована за патерном [MVC](#) ([Model-View-Controller](#)) на основі вимог до функціональності у п. 1.4 цієї бакалаврської роботи. Зпроектована модель містить такі елементи: 1. [Model](#) (Модель) Відповідає за логіку бізнес-процесів, зберігання та обробку даних. Основні компоненти: [TrainingPlan](#) – модель плану тренувань (тип активності, методика, тривалість циклу). [TrainingSession](#) – окрема сесія тренування (дата, тип, тривалість, обсяг, самопочуття). [UserProfile](#) – профіль користувача (вік, вага, стать, мета). [AnalyticsData](#) – агреговані дані для побудови графіків. [RecommendationEngine](#) – модуль ШІ для аналізу історії та рекомендацій. [StorageManager](#) – робота з локальною БД ([SQLite](#)). 2. [View](#) (Подання) Відповідає за інтерфейс користувача ([Windows Forms](#)). Основні форми: [MainForm](#) – головне меню з вкладками: Планування, Журнал, Аналітика, Експорт/Налаштування. [TrainingPlannerForm](#) – створення/редагування плану тренувань. [TrainingJournalForm](#) – введення та перегляд сесій. [AnalyticsForm](#) – графіки динаміки, план-факт, навантаження. [ExportForm](#) – експорт у [PDF/Excel](#), резервне копіювання. [ProfileSettingsForm](#) – налаштування профілю користувача. 3. [Controller](#) (Контролер) Керує взаємодією між [Model](#) і [View](#). Основні контролери: [TrainingController](#) – логіка створення, редагування, копіювання тренувань. [AnalyticsController](#) – обробка даних для графіків. [ExportController](#) – генерація звітів, експорт. [ProfileController](#) – збереження налаштувань користувача. [RecommendationController](#) – виклик модуля ШІ для порад. Додаткові модулі: [AI Recommendation Engine](#) (опціонально): окремий сервіс або бібліотека, яка аналізує історію тренувань і дає поради. [Data Exporter](#): модуль для генерації [PDF/Excel](#). [BackupManager](#): створення архіву даних. Діаграмою архітектури [Windows Forms](#) додатку для планування та обліку спортивних тренувань, реалізованого за патерном [MVC](#), представлено на рис. 2.1. На рис. 2.1 включено: [View](#) ([Forms](#)) – інтерфейс користувача [Controller](#) ([Logic](#)) – логіка взаємодії [Model](#) ([Data](#)) – бізнес-логіка та зберігання [RecommendationEngine](#) – модуль ШІ [Exporter](#) – модуль експорту [StorageManager](#) – робота з локальною БД Рис. 2.1 Архітектура [Windows Forms](#) додатку ([MVC](#)) На рис. 2.1 показано зв'язки в системі: [View](#) → [Controller](#) [Controller](#) → [Model](#) [Controller](#) → [RecommendationEngine](#) / [Exporter](#) [Model](#) → [StorageManager](#) [View](#) → [Controller](#). [View](#) (форми) не містить бізнес-логіки, але реагує на дії користувача (натискання кнопок, введення даних). Вона передає події (наприклад,

» Цитування: 0,02%	id: 10
---------------------------	---------------

“зберегти тренування”,

Цитування: 0,02%

id: 11

“побудувати графік”

) до [Controller](#). [View](#) також інформує [Controller](#), які елементи інтерфейсу активні, які дані введено, які компоненти потрібно оновити. [Controller](#) → [Model](#). [Controller](#) керує логікою обробки даних: створення, оновлення, зчитування об'єктів моделі. Він викликає методи моделей ([TrainingPlan](#), [TrainingSession](#), [UserProfile](#)) для збереження або отримання інформації. [Controller](#) → [RecommendationEngine](#). [Controller](#) ініціює запит до модуля ШІ, коли потрібно згенерувати рекомендації. Передає також історію тренувань, самопочуття, цілі користувача — отримує у відповідь поради щодо навантаження, типу активності тощо. [Controller](#) → [Exporter](#). [Controller](#) передає дані (журнал, графіки, профіль) до модуля експорту. [Exporter](#) формує [PDF](#) або [Excel](#)-файл, який користувач може зберегти або надіслати. [Model](#) → [StorageManager](#). Моделі не працюють безпосередньо з базою даних — вони делегують це [StorageManager](#). [StorageManager](#) реалізує [CRUD](#)-операції ([create](#), [read](#), [update](#), [delete](#)) з локальною БД ([SQLite](#)). Послідовність шарів архітектури зображено на рис. 2.2 Рис. 2.2 Діаграма послідовності для архітектури застосунку Таким чином, архітектуру системи сконструйовано. 2.2. Концептуальна модель бази даних: таблиці, зв'язки, логіка збереження Концептуальну модель бази даних для [Windows Forms](#)-додатку розроблено з урахуванням функціональних вимог та архітектури [MVC](#). Логіка збереження наступна: тренування зберігаються через [TrainingSession](#), прив'язуються до [TrainingPlan](#) і [User](#); пропущені тренування позначаються через [IsMissed](#), а перенесення реалізується копіюванням з [CopiedFromSessionID](#); рекомендації генеруються ШІ-модулем і зберігаються в [Recommendations](#); аналітика може кешуватись у [AnalyticsCache](#) для швидкого доступу. Експорт фіксується в [ExportLogs](#) для історії. Загальну схему бази даних та зв'язки представлено на рис. 2.3. Рис. 2.3 [ER](#) діаграма бази даних Кожна таблиця представлена як [entity](#). Відображено типи зв'язків: ||--o{ — один до багатьох (один користувач → багато тренувань). }o--|| — самозв'язок для копіювання сесій. Всі основні поля, типи [ENUM](#) і [nullable FK](#) враховані. Таблиця 2.1 Загальна модель зв'язків таблиці Таблиця Зв'язок Призначення [Users](#) 1 → ∞ до [TrainingPlans](#) Один користувач має кілька планів [Users](#) 1 → ∞ до [TrainingSessions](#) Один користувач має багато тренувань [TrainingPlans](#) 1 → ∞ до [TrainingSessions](#) Один план містить багато тренувань [TrainingSessions](#) ∞ → 1 до [TrainingPlans](#), [Users](#) Тренування належить одному плану та користувачу [TrainingSessions](#) опціонально → [TrainingSessions](#) Перенесення або копіювання тренування [Users](#) → [Recommendations](#) 1 → ∞ Рекомендації, згенеровані для користувача [Users](#) → [AnalyticsCache](#) 1 → ∞ Кеш для відображення метрик [Users](#) → [ExportLogs](#) 1 → ∞ Логування експорту користувача У табл. 2.1 представлено загальну модель зв'язків бази даних. Опишемо таблиці бази даних. Таблиця [Users](#) (рис. 2.4) зберігає основну інформацію про користувачів системи: спортсменів, для яких формуються плани тренувань, фіксуються сесії та рекомендації. Поля: [UserID](#) (PK) — унікальний ідентифікатор користувача. [Name](#) — ім'я користувача. [Age](#) — вік користувача. [Gender](#) — стать. [Weight](#), [Height](#) — антропометричні дані, що можуть бути використані для аналізу та рекомендацій. [Goal](#) — ціль тренування ([ENUM](#): схуднення, маса, витривалість). [PreferredMethod](#) — обрана методика тренування ([ENUM](#) або [FK](#) на таблицю методик, якщо буде реалізована). Зв'язки: 1 → ∞ до [TrainingPlans](#) (один користувач може мати багато планів). 1 → ∞ до [TrainingSessions](#) (один користувач має багато тренувань). 1 → ∞ до [Recommendations](#), [ExportLogs](#), [AnalyticsCache](#). Рис. 2.4 Таблиця

Цитування: 0,01%

id: 12

«[Users](#)»

[TrainingPlans](#) (рис. 2.5) містить дані про створені тренувальні плани для кожного користувача. План об'єднує групу тренувальних сесій. Поля: [PlanID](#) (PK) — унікальний ідентифікатор плану. [UserID](#) (FK) — посилання на користувача, для якого створено план. [StartDate](#), [EndDate](#) — часові межі дії плану. [Method](#) — обрана методика ([ENUM](#): періодизація, інтервальне, спліт, прогресивне). [CycleLength](#) — довжина одного циклу плану (в днях). [IsAutoGenerated](#) — позначає, чи був план згенерований автоматично ([true/false](#)). Зв'язки: 1 → ∞ до [TrainingSessions](#) — план включає тренування. Рис. 2.5 Таблиця

Цитування: 0,01%

id: 13

«[TraininfPlans](#)»

Recommendations (рис. 2.6) зберігає поради від ШІ-модуля або логіки програми щодо змін у тренуваннях користувача. Поля: **RecommendationID** (PK) — унікальний ідентифікатор рекомендації. **UserID** (FK) — кому адресована рекомендація. **DateGenerated** — дата створення поради. **SuggestedChange** — текстова пропозиція щодо змін. **RiskIndicator** — ризик, виявлений системою (ENUM: **low**, **medium**, **high**). **Reason** — пояснення, чому сформовано цю рекомендацію. Зв'язки: $\infty \rightarrow 1$ до **Users**. **TrainingSessions** (рис. 2.7) фіксує кожну тренувальну сесію, її параметри, результати та статус виконання. Поля: **SessionID** (PK) — унікальний ідентифікатор сесії. Рис. 2.6 Таблиця

Цитування: 0,01%

id: 14

«Recommendations»

PlanID (FK) — до якого плану належить тренування. **UserID** (FK) — користувач, що виконує тренування. **Date** — дата проведення тренування. **ActivityType** — тип активності (біг, силові, гнучкість тощо). **ExerciseName** — назва вправи. **Duration** — тривалість (хвилини). **Volume** — обсяг навантаження (наприклад, 3 підходи по 10 повторень або 5 км). **FatigueLevel** — рівень втоми після тренування (шкала 1-10). **Comment** — вільний коментар до сесії. **IsMissed** — чи було пропущено тренування. **CopiedFromSessionID** — FK на інше тренування, з якого скопійовано дані (для перенесення або повторення). Зв'язки: $\infty \rightarrow 1$ до **TrainingPlans**, **Users**. Може мати FK до самої себе через **CopiedFromSessionID**. **AnalyticsCache** (рис. 2.8) це кешована аналітика за певними метриками для пришвидшення відображення графіків і звітів. Рис. 2.7 Таблиця

Цитування: 0,01%

id: 15

«TrainingSession»

Поля: **UserID** (FK) — користувач, для якого зберігається аналітика. **MetricType** — тип метрики (ENUM: обсяг, навантаження, план-факт). **Period** — період, за який збережена аналітика (ENUM: день, тиждень, місяць). **DataJSON** — структура даних у JSON-форматі (наприклад, для графіків). Зв'язки: $1 \rightarrow 1$ або ∞ до **Users** (одна метрика на один період). Рис. 2.8 Таблиця

Цитування: 0,01%

id: 16

«AnalyticsCache»

ExportLogs фіксує історію експорту даних користувача (звіти, журнали, таблиці). Поля: **ExportID** (PK) — ідентифікатор експорту. **UserID** (FK) — кому належить експорт. **ExportDate** — дата створення експорту. **Format** — формат (ENUM: **PDF**, **Excel**). **PeriodStart**, **PeriodEnd** — за який період сформовано звіт. **FilePath** — шлях до збереженого файлу. Зв'язки: $\infty \rightarrow 1$ до **Users**. Рис. 2.9 Таблиця

Цитування: 0,01%

id: 17

«ExportLogs»

2.3. Розробка інтерфейсної структури додатку: головне меню, форма тренування, статистика Інтерфейс застосунку **TrainMind** реалізовано з урахуванням принципів доступності, інтуїтивності та логічного поділу функціональності за вкладками. Всі форми розроблені на основі технології **Windows Forms** з використанням вкладеного меню та панелей управління, що забезпечує швидкий доступ до ключових функцій без перевантаження екрану. Головна форма **MainForm** (рис. 2.10) реалізована у вигляді інтерфейсу з чотирма переходами за основними розділами: Планування, Журнал, Аналітика, Експорт / Налаштування (головне меню). У верхній частині розміщено навігаційне меню, ліворуч логотип застосунку, праворуч кнопки виклику інших форм а в нижній частині форми — коротка інформаційна панель зі станом підключення до бази даних, профілем користувача та поточною датою. Рис.2.10 Інтерфейс головної форми **MainForm** Рис. 2.11 Меню **MainForm** На рис. 2.11 зображено меню головної форми застосунку. Форма **TrainingPlannerForm** (рис. 2.12) дозволяє створювати та редагувати тренувальні плани. Вона містить такі елементи: комбінований список вибору методики (періодизація, інтервальне тощо); введення початкової та кінцевої дати плану; поле вводу довжини циклу (в днях); перемикач

Цитування: 0,04%

id: 18

«Згенерувати план автоматично»

; кнопки

Цитування: 0,01%	id: 19
"Зберегти",	
Цитування: 0,01%	id: 20
"Очистити",	
Цитування: 0,02%	id: 21
"Завантажити шаблон".	

Для кожного дня плану відображається таблиця із попередньо заповненими сесіями або можливістю вручну додати тип активності. 2.12 Форма створення плану тренувань [TrainingPlannerForm](#) Форма журналу тренувань [TrainingJournalForm](#) (рис. 2.13) реалізована у вигляді табличного подання з можливістю фільтрації за датами та типом активності. У таблиці відображаються: дата, назва вправи, тип активності; тривалість, обсяг, рівень втоми; коментар та статус (виконано / пропущено). Праворуч розміщено панель швидкого додавання або редагування обраної сесії з усіма полями: [ExerciseName](#), [Duration](#), [Volume](#), [FatigueLevel](#), [Comment](#). Рис.2.13 Інтерфейс форми журналу тренувань [TrainingJournalForm](#)]

[AnalyticsForm](#) (рис. 2.14) – статистика та графіки, у цій формі реалізовано відображення аналітики на основі даних тренувального журналу: графік план-факт виконання тренувань; динаміка навантаження по тижнях; зміна рівня втоми з часом; порівняння активностей (біг, силові, гнучкість). Інтерфейс поділено на ліву частину з фільтрами (період, тип метрики) і праву — з динамічними графіками на базі [ScottPlot](#). Рис. 2.14 – Графічний інтерфейс форми аналітики [AnalyticsForm](#)]

Форма [ExportForm](#) (рис. 2.15) – дозволяє користувачу експортувати результати тренувань за обраний період: вибір формату ([PDF](#) або [Excel](#)); вибір діапазону дат; кнопка генерації; перегляд шляху збереженого файлу; можливість зберегти резервну копію бази даних. Також передбачено журнал останніх експортів. Рис. 2.15 Форма експорту та резервного копіювання [ExportForm](#)]

[ProfileSettingsForm](#) (рис. 2.16 – налаштування профілю користувача. У профільній формі користувач може налаштувати: Ім'я, вік, стать, зріст, вагу; Мету тренувань ([Goal](#)); Улюблену методику; Переглянути або змінити пароль (за потреби); Очистити історію або скинути налаштування до типових. Інтерфейс складається з простих текстових полів і комбобоксів, що спрощує введення. Рис. 2.16 Інтерфейс форми налаштувань профілю [ProfileSettingsForm](#)]

Загальна концепція інтерфейсу базується на принципі

Цитування: 0,04%	id: 22
"розумної простоти":	

кожна функція доступна у відповідному розділі, а взаємодія з системою не потребує спеціальних навичок. Такий підхід підвищує зручність користування та забезпечує позитивний досвід роботи з програмою як для новачків, так і для досвідчених спортсменів.

2.4. Вибір технологій: [C#](#), [Windows Forms](#), [SQLite](#), додаткові бібліотеки Для реалізації десктопного застосунку було обрано перевірений і ефективний стек технологій, що забезпечує швидкість розробки, підтримку локальної автономної роботи та зручність роботи з базами даних. Основною мовою програмування виступає [C#](#), а в якості інтерфейсної платформи використано [Windows Forms](#) — надійну та просту у використанні технологію для створення віконних застосунків у середовищі [Windows](#). Для збереження даних використовується [SQLite](#) — легка файлова реляційна СУБД, яка не потребує окремого серверного розгортання. [SQLite](#) зберігає всі дані в одному файлі, що значно спрощує процес резервного копіювання та перенесення застосунку на інші пристрої. Для взаємодії із СУБД обрано [Entity Framework Core](#) — сучасний [ORM](#)-фреймворк від [Microsoft](#), який дозволяє працювати з базою даних на рівні об'єктів. Це дає змогу уникати прямого написання [SQL](#)-запитів, забезпечити рефакторинг, відстеження змін, а також просте створення міграцій бази даних. У проєкті також використано низку сторонніх бібліотек для реалізації додаткового функціоналу: [Microsoft.EntityFrameworkCore.Sqlite](#) — провайдер для використання [Entity Framework](#) з [SQLite](#); [ScottPlot](#) — побудова графіків тренувальної активності; [PdfSharp](#) — генерація звітів у форматі [PDF](#); [Newtonsoft.Json](#) — обробка та збереження структурованих даних у форматі [JSON](#) (для кешу аналітики). При виборі технології було враховано такі міркування: швидкість розробки завдяки [Entity Framework](#) та готовим [UI](#)-компонентам [Windows Forms](#); автономність — додаток повністю працює без підключення до Інтернету; гнучкість і масштабованість — за рахунок використання [ORM](#) та структури моделей; легке розгортання — достатньо скопіювати [.exe](#) та [.db](#) файл; можливість подальшого розвитку — легко перейти до [MAUI](#) або [WPF](#) з цим кодовим

базисом. Отже, обрана технологічна основа забезпечує оптимальну рівновагу між функціональністю, простотою, продуктивністю та розширюваністю, що робить її ідеальною для навчальних і прикладних проєктів. РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ 3.1. Реалізація базових функцій: створення, редагування та перегляд тренувань Відповідно до розробленої архітектури в другому розділі застосунку реалізовано код окремих файлів коду проєкту [TrainMind](#). У результаті було реалізовано структуру, як зображено на рис. 3.1. Рис. 3.1 Структура застосунку [TrainMind](#) На першому кроці реалізовано структуру бази даних відповідно до розробленої моделі в п.2.2 з використанням фреймворку [Entity Framework Core](#). Код представлено в додатку А. Опишемо лише ключовий клас [AppDbContext](#) є контекстом бази даних. Він визначає, які таблиці існують у базі, як вони пов'язані між собою, і як з ними працювати. Листинг 3.1 [internal class AppDbContext](#):

```
DbContext { public DbSet<User> Users { get; set; } public DbSet<TrainingPlan> TrainingPlans { get; set; } public DbSet<TrainingSession> TrainingSessions { get; set; } public DbSet<Recommendation> Recommendations { get; set; } public DbSet<AnalyticsCache> AnalyticsCaches { get; set; } public DbSet<ExportLog> ExportLogs { get; set; } }
```

Кожне з цих полів на листингу 3.1 представляє таблицю в базі даних: [Users](#) — користувачі [TrainingPlans](#) — плани тренувань [TrainingSessions](#) — окремі тренування [Recommendations](#) — рекомендації [AnalyticsCaches](#) — кеш аналітики [ExportLogs](#) — журнали експорту Метод [OnModelCreating](#) описує структуру таблиць та зв'язки між ними: [HasKey\(...\)](#) — визначає первинний ключ таблиці. [HasOne\(...\).WithMany\(...\)](#) — описує зв'язок

Цитування: 0,04%

id: 23

"один до багатьох".

[OnDelete\(DeleteBehavior.Cascade\)](#) — при видаленні батьківського запису, пов'язані дочірні також видаляються. [OnDelete\(DeleteBehavior.Restrict\)](#) — забороняє видалення, якщо є пов'язані записи. Метод [OnConfiguring](#) вказує, що використовується база даних [SQLite](#), яка зберігається у файлі [TrainMind.db](#). Цей клас описує структуру бази даних для застосунку, який пов'язаний із тренуваннями, аналітикою та рекомендаціями для користувачів. Він дозволяє працювати з даними через об'єкти, не використовуючи напряму [SQL](#)-запити. Форма [TrainingPlannerForm](#) реалізує графічний інтерфейс для створення, перегляду та видалення індивідуальних планів тренувань користувачів. Вона є частиною [Windows Forms](#)-застосунку й використовує [Entity Framework Core](#) для доступу до бази даних. Код наведено у додатку Б. Під час ініціалізації форми у конструкторі відбувається завантаження списку користувачів з бази даних у компонент [comboBox1](#), де відображається ім'я кожного користувача. Це дозволяє обрати, кому призначається план. Дані про створені плани автоматично підвантажуються в [DataGridView](#) для перегляду. Це реалізується у методі [Refresh\(\)](#), який викликається як під час відкриття форми, так і після додавання або видалення записів. Листинг 3.2 Код методу [Refresh\(\)](#) [public void Refresh\(\)](#) { [using \(var context = new AppDbContext\(\)\)](#) { [var tp = context.TrainingPlans.ToList\(\)](#); [dataGridView1.DataSource = tp](#); } } Форма підтримує два способи створення планів тренувань: ручне обрання методик — користувач може самостійно обрати одну або кілька типових методик (наприклад, кардіо, силові тренування тощо) за допомогою відповідних [CheckBox](#). автоматичне генерування плану — передбачене використання спеціального режиму, який у майбутньому буде вдосконалено за допомогою штучного інтелекту. У поточній версії автоматично додається лише базова методика. Також форма дає змогу вказати період тренування: обирається початкова дата, після чого система або автоматично рахує кінцеву дату залежно від кількості днів, або навпаки — обчислює тривалість циклу, виходячи з кінцевої дати. Після натискання кнопки

Цитування: 0,01%

id: 24

"Зберегти",

введені дані збираються з інтерфейсу й створюється об'єкт типу [TrainingPlan](#). Далі цей об'єкт зберігається в базу даних через контекст [AppDbContext](#), після чого таблиця оновлюється, а користувач отримує повідомлення про успішне збереження плану. Листинг 3.3 Основний код створення плану тренувань // Збір даних з форми [DateTime startDate = dateTimePicker1.Value.Date](#); [DateTime endDate = dateTimePicker2.Value.Date](#); [int cycleLength = \(int\)numericUpDown1.Value](#); // Типи активності [List<string> methods = new List<string>\(\)](#); [if \(checkBox1.Checked\) methods.Add\(checkBox1.Text\)](#); [if \(checkBox2.Checked\) methods.Add\(checkBox2.Text\)](#); [if \(checkBox3.Checked\) methods.Add\(checkBox3.Text\)](#); [if \(checkBox4.Checked\) methods.Add\(checkBox4.Text\)](#); [if \(checkBox8.Checked\)](#) { // Автоматичне


```

генерування плану methods.Add(checkBox1.Text); // планується лише кардіо. Далі буде ШІ }
if (methods.Count == 0) { MessageBox.Show(

```

Цитування: 0,08% id: 25

"Будь ласка, оберіть хоча б одну методику."

```

); return; } if (numericUpDown1.Value == 0) { MessageBox.Show(

```

Цитування: 0,06% id: 26

"Будь ласка, оберіть період тренування."

```

); return; } // MessageBox.Show(comboBox1.SelectedIndex.ToString()); // Створення нового
плану var newPlan = new TrainingPlan { UserID = comboBox1.SelectedIndex + 1, StartDate =
startDate, EndDate = endDate, CycleLength = cycleLength, Method = string.Join(

```

Цитування: 0% id: 27

", ",

```

methods), IsAutoGenerated = false }; using (var db = new AppDbContext()) {
db.TrainingPlans.Add(newPlan); db.SaveChanges(); } Refresh(); MessageBox.Show(

```

Цитування: 0,05% id: 28

"План тренувань успішно збережено!"

); Крім того, передбачено можливість видалення створених планів. Після вибору відповідного запису в таблиці та підтвердження дії, запис видаляється з бази даних, а таблиця оновлюється. Ця функціональність дозволяє ефективно керувати тренувальними планами, підтримуючи індивідуалізацію та адаптацію навчального процесу до потреб кожного користувача. Форма `TrainingJournalForm` (див. Додаток В) реалізує функціонал ведення журналу тренувань користувача. Вона надає можливість переглядати, додавати та видаляти записи тренувальних сесій, а також зв'язувати кожне тренування з відповідним користувачем і планом. Код в додатку В. При ініціалізації форми (`TrainingJournalForm`) завантажується список користувачів із бази даних та підключається до `comboBox1`. Відразу викликається метод `LoadSessions()`, який заповнює два основні компоненти інтерфейсу: `dataGridView1` — журнал тренувань. `dataGridView2` — список існуючих тренувальних планів користувача. Метод `LoadSessions()` завантажує пов'язані з обраним користувачем: Доступні плани тренувань, що підтягуються до `comboBox3`. Список тренувальних сесій із таблиці `TrainingSessions`. Інформацію про плани, що відображаються у вигляді таблиці у `dataGridView2`. `public void LoadSessions() { var context = new AppDbContext(); using (var db = new AppDbContext()) { var plans = db.TrainingPlans .Where(p = p.UserID == comboBox1.SelectedIndex + 1) // Assuming UserID is 1-based index .Select(p = new { p.PlanID, DisplayText = $`

Цитування: 0,02% id: 29

"{p.PlanID}"

```

} ) .ToList(); comboBox3.DataSource = plans; comboBox3.DisplayMember =

```

Цитування: 0,01% id: 30

"DisplayText"

```

; // що бачить користувач comboBox3.ValueMember =

```

Цитування: 0,01% id: 31

"PlanID"

```

; // що зберігається як значення } var TrainingPlans = context.TrainingPlans .Where(x =
x.UserID == comboBox1.SelectedIndex + 1) .ToList(); // Set the DataSource of the DataGridView
to the list of training sessions dataGridView2.DataSource = TrainingPlans.Select(x = new {
x.PlanID, x.UserID, x.StartDate, x.EndDate, x.Method, x.CycleLength }).Where(x = x.UserID ==
comboBox1.SelectedIndex + 1) .ToList(); var trainingSessions = context.TrainingSessions
.Where(x = x.UserID == comboBox1.SelectedIndex + 1) .ToList(); dataGridView1.DataSource =
trainingSessions.Select(x = new { x.SessionID, x.PlanID, x.Date, x.ActivityType, x.ExerciseName,
x.Duration, x.Volume, x.FatigueLevel, x.Comment, x.IsMissed }).ToList(); } Зміна користувача в
comboBox1 автоматично перезавантажує відповідні дані через подію
comboBox1_SelectedIndexChanged. Для додавання нового запису про тренування
передбачено кнопку, обробник якої реалізовано в методі button1_Click. Після перевірки

```


наявності обраного користувача, створюється новий об'єкт [TrainingSession](#) із параметрами: Дата тренування (поточна або задана через додатковий компонент). Назва вправи, тип активності, тривалість, об'єм, рівень втоми, коментар тощо. Цей об'єкт зберігається в базі даних, після чого таблиця [dataGridView1](#) оновлюється. Для видалення запису передбачено кнопку з обробником [button2_Click](#). Після вибору відповідного запису та підтвердження дії, сесія видаляється з бази, а журнал тренувань оновлюється. [comboBox2](#) дозволяє обирати тип активності; при зміні значення динамічно оновлюється відповідний напис ([label9](#)). Кнопка [button3](#) закриває форму. Цей модуль реалізує ключовий функціонал запису та моніторингу тренувального процесу в системі, що дозволяє вести персоналізований журнал активності користувачів і формує основу для подальшого аналізу ефективності тренувань у застосунку. З метою спрощення повторного планування однакових або подібних тренувань у системі реалізовано контролер [CopyTrainingSession](#), який дозволяє скопіювати існуючу тренувальну сесію за її ідентифікатором. Ця функціональність є корисною, коли користувач повторює ті самі вправи або цикли тренувань. [public void CopyTrainingSession\(int sessionId\) { var session = _context.TrainingSessions.Find\(sessionId\); if \(session != null\) { var newSession = new TrainingSession { PlanID = session.PlanID, UserID = session.UserID, Date = session.Date, ActivityType = session.ActivityType, ExerciseName = session.ExerciseName, Duration = session.Duration, Volume = session.Volume, FatigueLevel = session.FatigueLevel, Comment = session.Comment, IsMissed = session.IsMissed, CopiedFromSessionID = session.SessionID }; _context.TrainingSessions.Add\(newSession\); _context.SaveChanges\(\); } }](#) Метод приймає параметр [sessionId](#) — ідентифікатор оригінального запису тренування, що зберігається в базі даних. Після знаходження відповідної сесії ([_context.TrainingSessions.Find\(sessionId\)](#)), створюється новий об'єкт типу [TrainingSession](#), який копіює всі поля: Ідентифікатори плану та користувача; Дату тренування; Тип активності та назву вправи; Тривалість і об'єм; Рівень втоми, коментар і статус виконання. Додатково в полі [CopiedFromSessionID](#) зберігається ідентифікатор оригінальної сесії. Це дає змогу відстежувати походження копії для подальшого аналізу або аналітичного обліку. Новий запис додається в базу даних за допомогою [Add](#), після чого зміни зберігаються методом [SaveChanges](#). Таким чином, метод [CopyTrainingSession](#) забезпечує зручний механізм повторного використання тренувальних сесій і підвищує гнучкість користувача у веденні свого журналу тренувань.

3.2. Реалізація аналітики: побудова графіків прогресу (час, дистанція, навантаження)

У застосунку реалізовано окремий контролер [AnalyticsController](#), який відповідає за обробку даних тренувальних сесій та формування узагальненої аналітики для візуалізації й аналізу ефективності тренувань. Контролер взаємодіє з базою даних через об'єкт [AppDbContext](#). Код наведено у додатку Г. Основні функції контролера:

- Отримання тренувальних сесій Метод [GetTrainingSessions\(int userId\)](#) повертає повний список усіх тренувальних сесій для заданого користувача. Це є базовим джерелом даних для подальших агрегованих розрахунків.
- Аналітика навантаження за датами Реалізовано три варіанти розрахунку навантаження: [GetLoadByDate](#) — навантаження визначається як сума тривалостей ([Duration](#)) сесій за кожен день. [GetLoadByDate2](#) — розрахунок базується на добутку об'єму ([Volume](#)) на тривалість, що є більш точним показником тренувального навантаження. [GetLoadByDate3](#) — використовується рівень втоми ([FatigueLevel](#)) як критерій інтенсивності, що дозволяє аналізувати суб'єктивне навантаження. У кожному з цих методів дані групуються за датами, відсортовуються за хронологією та повертаються у вигляді об'єкта [AnalyticsData](#), який містить списки дат і відповідних значень навантаження. [public static AnalyticsData GetLoadByDate\(int userId, DateTime start, DateTime end\) { using \(var db = new AppDbContext\(\)\) { var sessions = db.TrainingSessions.Where\(s => s.UserID == userId && s.Date == start && s.Date == end\).ToList\(\); var grouped = sessions.GroupBy\(s => s.Date.Date\).OrderBy\(g => g.Key\).Select\(g => new { Date = g.Key, Load = g.Sum\(s => s.Duration\) // приклад формули навантаження }\); return new AnalyticsData { Dates = grouped.Select\(g => g.Date\).ToList\(\), Loads = grouped.Select\(g => \(float\)g.Load\).ToList\(\) }; }](#)
- Агрегована аналітика за періодами Реалізовано методи, що дозволяють агрегувати обсяг тренувального навантаження за різними періодами: [GetTrainingVolumeByDay](#) — агрегація за кожен день. [GetTrainingVolumeByWeek](#) — агрегація за тижнями року, з використанням системного календаря. [GetTrainingVolumeByMonth](#) — підсумки за кожен місяць року. Усі методи повертають словники ([Dictionary string, float](#)) з ключами у форматі дати, тижня або місяця, що дозволяє легко будувати графіки зміни навантаження у динаміці. Для реалізації аналізу тренувального навантаження користувачів у застосунку створено форму [AnalyticsForm](#), яка дозволяє будувати графіки на основі вибраного періоду та методу оцінювання. Інтерфейс

реалізовано за допомогою бібліотеки [ScottPlot](#), яка забезпечує інтеграцію графічного компонента [formsPlot1](#) у [Windows Forms](#). При відкритті форми в конструкторі завантажуються список користувачів у [comboBox1](#), що дозволяє обрати конкретного користувача для аналізу його тренувальних сесій. Метод [AnalyticsForm_Load](#) показує приклад двох простих графіків при завантаженні форми — це допоміжний код, який може бути замінений або використаний для тестування. Натискання кнопки викликає метод [GetLoadByDate](#) з [AnalyticsController](#), що обчислює загальну тривалість тренувань у вибраному періоді. Отримані значення перетворюються у масиви [double\[\]](#) і виводяться на графіку як лінія залежності навантаження від дати. [private void button1_Click\(object sender, EventArgs e\)](#) { [var start = dateTimePicker1.Value.Date](#); [var end = dateTimePicker2.Value.Date](#); [var data = AnalyticsController.GetLoadByDate\(comboBox1.SelectedIndex + 1, start, end\)](#); if ([data.Dates.Count == 0](#)) [MessageBox.Show](#)(\$

” Цитування: **0,19%** id: 32

"Знайдено {[data.Dates.Count](#)} записів з {[start.ToShortDateString\(\)](#)} по {[end.ToShortDateString\(\)](#)}."); else { [double\[\]](#)

[x = data.Dates.Select\(d => d.ToOADate\(\)\).ToArray\(\)](#); // [Convert DateTime to OLE Automation Date for plotting formsPlot1.Plot.Clear\(\)](#); [formsPlot1.Plot.Axes.SetLimits\(start.ToOADate\(\), end.ToOADate\(\), 0, data.Loads.Max\(\) * 1.1\)](#); [formsPlot1.Plot.Add.Scatter\(x, data.Loads.ToArray\(\)\)](#); [formsPlot1.Plot.XLabel](#)(

” Цитування: **0,01%** id: 33

"Дата"

); [formsPlot1.Plot.YLabel](#)(

” Цитування: **0,01%** id: 34

"Виконання"

); [formsPlot1.Plot.Title](#)(\$

” Цитування: **0,11%** id: 35

"Тренування з {[start.ToShortDateString\(\)](#)} по {[end.ToShortDateString\(\)](#)}"

); [formsPlot1.Refresh\(\)](#); } } Натискання кнопки [button2](#) формує графік на основі обчислення добутку [Volume x Duration](#) для кожної сесії ([GetLoadByDate2](#)). Цей графік дозволяє точніше оцінити сумарне фізичне навантаження в кожний день. Кнопка [button3](#) генерує графік рівня втоми, визначеного користувачем після кожного тренування ([GetLoadByDate3](#)). Це дозволяє оцінити відновлення та інтенсивність програми з точки зору самопочуття користувача. Візуальні параметри графіків: Горизонтальна вісь ([X](#)) відображає дати (перетворені у формат [OLE Automation Date](#)). Вертикальна вісь ([Y](#)) показує відповідні значення навантаження, тривалості або втоми. На графіку виводиться заголовок, підписи осей та лінії побудови ([Scatter](#)). У разі, якщо у вибраному періоді немає жодного тренування, користувачеві виводиться відповідне повідомлення з підрахованою кількістю записів. Форма [AnalyticsForm](#) виконує роль інструменту для візуального моніторингу ефективності тренувань. Завдяки використанню графіків, користувачі та тренери можуть оперативно аналізувати зміни навантаження, виявляти періоди перенавантаження або недотренування, а також коригувати планування майбутніх сесій. 3.3. Експорт даних у звіт (формати [Excel](#), [PDF](#)) У складі застосунку реалізовано контролер [DataExporter](#) (додаток [E](#)), що відповідає за базову обробку та збереження даних про тренувальні сесії користувачів у зовнішні файли. Цей модуль демонструє основні підходи до експорту інформації в популярні формати та створення резервної копії. 1. [ExportToPdf](#) Метод [ExportToPdf\(List TrainingSession sessions\)](#) формує текстовий файл із розширенням [.pdf](#), у якому пострічково виводиться інформація про кожну тренувальну сесію: дата, тип активності та об'єм. Хоча в даній реалізації не використовується справжня [PDF](#)-бібліотека, структура функції демонструє базовий принцип формування змісту майбутнього [PDF](#)-документа. 2. [ExportToExcel](#) Метод [ExportToExcel\(List TrainingSession sessions\)](#) аналогічно генерує файл із розширенням [.xlsx](#), виводячи дані у табличному форматі з заголовками колонок. В поточній версії метод створює звичайний текстовий файл, але з можливістю його подальшого відкриття у табличному редакторі (наприклад, [Excel](#)). Реалізація може бути надалі розширена з використанням бібліотек на кшталт [EPPlus](#) або [ClosedXML](#). 3. [CreateBackup](#) Метод [CreateBackup\(List TrainingSession sessions\)](#) створює резервну копію інформації про тренування в текстовому форматі ([training_sessions_backup.txt](#)). Це дозволяє зберегти

критично важливу інформацію у випадку втрати основної бази даних або для перенесення в інші системи. Всі методи використовують клас [StreamWriter](#), який відкриває або створює файл на диску та послідовно записує туди структуровану інформацію. Кожен файл містить заголовок, а далі — пострічкові записи кожної сесії, що містять ключові атрибути: дата проведення, тип активності, об'єм тренування тощо. Клас [DataExporter](#) виконує роль службового інструмента для виведення даних з бази у формати, зручні для читання, друку або збереження. У наступних версіях застосунку даний модуль може бути розширений використанням повноцінних бібліотек для створення [PDF](#) та [Excel](#)-файлів, а також інтеграцією з хмарними сервісами резервного копіювання. Форма [ExportForm](#) (додаток Ж) реалізує інтерфейс для експорту тренувальних даних користувачів у різні формати файлів, а також для створення резервних копій. Основна мета цієї форми — забезпечити збереження інформації про тренування в зручному для аналізу або архівування вигляді. Компонент [comboBox1](#) ініціалізується значенням за замовчуванням (перший варіант — [PDF](#)). У [comboBox2](#) підвантажуються користувачі з бази даних для вибору користувача, чия активність потрібно експортувати. Таблиця [dataGridView1](#) заповнюється журналом попередніх експортів з таблиці [ExportLogs](#). Натискання кнопки [button2](#) викликає метод [ExportController.BackupData\(...\)](#), що створює резервну копію даних залежно від вибраного типу. Деталі реалізації методу резервного копіювання винесені в окремий контролер. Натискання кнопки [button1](#) ініціює процес експорту: вибирається користувач та діапазон дат ([start](#), [end](#)) за допомогою [DateTimePicker](#). Завантажуються відповідні тренування користувача. Після вибору типу експорту з [comboBox1](#) та шляху збереження у [SaveFileDialog](#) система виконує дію залежно від обраного формату.

```
PdfDocument document = new PdfDocument(); document.Info.Title =
```

Цитування: 0,04%

id: 36

"Training Sessions Report"

```
; PdfPage page = document.AddPage(); XGraphics gfx = XGraphics.FromPdfPage(page);
GlobalFontSettings.UseWindowsFontsUnderWindows = true; XFont font = new XFont(
```

Цитування: 0,01%

id: 37

"Arial",

```
10, XFontStyleEx.Regular); double y = 40; foreach (var session in sessions2) { if (y page.Height -
100) { page = document.AddPage(); gfx = XGraphics.FromPdfPage(page); y = 40; }
gfx.DrawString($
```

Цитування: 0,04%

id: 38

"Date: {session.Date:yyyy-MM-dd}",

```
font, XBrushes.Black, new XRect(40, y, page.Width, 20)); y += 15; gfx.DrawString($
```

Цитування: 0,04%

id: 39

"Activity: {session.ActivityType}",

```
font, XBrushes.Black, new XRect(40, y, page.Width, 20)); y += 15; gfx.DrawString($
```

Цитування: 0,04%

id: 40

"Exercise: {session.ExerciseName}",

```
font, XBrushes.Black, new XRect(40, y, page.Width, 20)); y += 15; gfx.DrawString($
```

Цитування: 0,05%

id: 41

"Duration: {session.Duration} min",

```
font, XBrushes.Black, new XRect(40, y, page.Width, 20)); y += 15; gfx.DrawString($
```

Цитування: 0,04%

id: 42

"Volume: {session.Volume}",

```
font, XBrushes.Black, new XRect(40, y, page.Width, 20)); y += 15; gfx.DrawString($
```

Цитування: 0,04%

id: 43

"Fatigue: {session.FatigueLevel}",

```
font, XBrushes.Black, new XRect(40, y, page.Width, 20)); y += 15; gfx.DrawString($
```

Цитування: 0,04%

id: 44

"Comment: {session.Comment}",

```
font, XBrushes.Black, new XRect(40, y, page.Width, 20)); y += 30; } document.Save(filePath2); Process.Start(
```

Цитування: 0,01%

id: 45

"explorer",

filePath2); Створюється PDF-файл за допомогою бібліотеки PdfSharp. Встановлюється базовий шрифт (Arial). Дані про кожну сесію (дата, тип активності, назва вправи, тривалість, об'єм, рівень втоми, коментар) виводяться посторінково з автоматичним переходом на нову сторінку за потреби. Збережений PDF відкривається автоматично після завершення експорту. Інші формати (Excel, JSON). Форма ExportForm є важливою складовою застосунку, що дозволяє користувачам зберігати історію своїх тренувань у зручному форматі, здійснювати резервне копіювання та, за потреби, експортувати дані для подальшого аналізу, друку або передачі. У подальшій розробці можливо розширення підтримки інших форматів та інтеграція з хмарними сховищами. 3.4. Тестування, перевірка коректності роботи, приклади використання Після реалізації функціональних модулів застосунку TrainMind було проведено тестування з метою перевірки коректності роботи інтерфейсів, збереження даних, аналітики та експорту. Тестування виконувалося вручну методом

Цитування: 0,02%

id: 46

«чорного ящика»

із залученням тест-користувачів, а також частково автоматизовано за допомогою unit-тестів для логічних компонентів. Було протестовано форми: створення плану (TrainingPlannerForm) — перевірка обов'язковості полів, допустимості діапазонів дат, валідація числових значень; додавання тренування (TrainingJournalForm) — перевірка цілісності записів у базі, обробка порожніх і помилкових значень; коректне оновлення даних при редагуванні існуючих записів. Додаток успішно обробляв помилкові дані, наприклад: спроба створити план без вказаної методики; введення некоректного значення обсягу (наприклад, від'ємна вага або 0 хв). В AnalyticsForm тестувались: правильність побудови графіків у реальному часі при зміні фільтрів; відповідність графіків фактичним даним з бази; оновлення динаміки при зміні або видаленні тренувань; сценарії, коли тренування пропущено або скопійовано. Додаток коректно обчислював тижневий обсяг тренувань, зберігав пропорції між активностями, автоматично відновлював кешовану аналітику після редагування даних. Форма ExportForm дозволила протестувати: генерацію PDF-файлу з переліком тренувань та графіком; створення Excel-звіту з фільтром за датами; наявність запису у журналі експорту (таблиця ExportLogs); правильність шляху до файлу та його збереження. Усі експортовані звіти були читабельні, мали відповідну структуру та не містили порожніх або невідповідних полів. Було сформовано три сценарії для налагодження. Приклади використання (сценарії) Сценарій 1. Новий користувач Користувач відкриває програму, переходить у

Цитування: 0,02%

id: 47

«Налаштування профілю»,

вводить свої дані (стать, вік, мета — наприклад,

Цитування: 0,01%

id: 48

«витривалість»

), створює план тренувань на місяць з методом

Цитування: 0,01%

id: 49

«періодизація».

Після цього він заповнює журнал, щодня додаючи активність. Сценарій 2. Автоматичне оновлення графіків Користувач додає два тренування у вигляді інтервальних сесій. У вкладці

Цитування: 0,01%

id: 50

«Аналітика»

автоматично відображається графік навантаження за тиждень. Після редагування одного з тренувань графік оновлюється миттєво. Сценарій 3. Формування звіту для тренера Користувач натискає

Цитування: 0,01%
«Експорт»,

id: 51

вибирає період з 01.04 по 30.04, обирає формат [Excel](#). Програма формує файл з табличними даними і динамікою навантаження. Звіт зберігається автоматично і фіксується у журналі. Результати тестування показали, що застосунок [TrainMind](#) відповідає поставленим функціональним вимогам, забезпечує стабільну роботу в автономному режимі та дозволяє ефективно керувати процесом тренувань, що підтверджує доцільність використання його як цифрового інструменту для підтримки фізичної активності. ЗАГАЛЬНІ ВИСНОВКИ У процесі виконання бакалаврської роботи було досягнуто поставленої мети – розроблено функціональний настільний застосунок ****TrainMind****, який дозволяє здійснювати планування, ведення журналу та аналіз спортивних тренувань користувача. Проведені дослідження, реалізація програмного забезпечення та його тестування дозволили сформулювати такі висновки: 1. Аналіз предметної області виявив актуальність розробки цифрових інструментів для індивідуального спортивного планування, які поєднують простоту використання з потужною аналітикою. Сучасні рішення переважно орієнтовані на мобільні та онлайн-середовища, що залишає відкритою нішу для офлайн-інструментів із широким функціоналом. 2. Виконано аналіз існуючих методик тренувань (класична періодизація, інтервальний метод, спліт-тренування, прогресивне навантаження), що дозволило вибрати оптимальні алгоритми для програмної реалізації. Запропонована модель додатку дозволяє автоматично адаптувати методики залежно від цілей користувача. 3. Було розроблено детальну концептуальну модель бази даних та взаємодії модулів додатку. Використано сучасні технології та бібліотеки ([.NET](#), [Windows Forms](#), [SQLite](#), [Entity Framework](#)), які забезпечують швидку розробку, автономність, гнучкість та стабільність роботи програмного продукту. 4. Інтерфейсна структура додатку реалізована за принципами інтуїтивної зрозумілості та мінімалізму. Основні форми (головне меню, планувальник тренувань, журнал активності, аналітика, налаштування профілю та експорт) мають логічну організацію та забезпечують швидкий доступ до всіх функцій системи. 5. Тестування застосунку показало, що програмний продукт стабільно виконує покладені на нього завдання, коректно обробляє помилкові сценарії та забезпечує високу продуктивність при роботі з великими обсягами тренувальних даних. Реалізовані можливості аналітики, візуалізації результатів та експорту відповідають практичним вимогам користувачів. Таким чином, розроблений [Windows Forms](#)-додаток ****TrainMind**** можна рекомендувати як ефективний інструмент для планування та контролю індивідуального тренувального процесу спортсменів-аматорів і тренерів. Подальший розвиток застосунку може бути пов'язаний із інтеграцією штучного інтелекту для персоналізованих рекомендацій, розширенням аналітичних можливостей, а також створенням кросплатформної версії на базі сучасних технологій, таких як [MAUI](#) або [Flutter](#).

Заявление об ограничении ответственности:

Этот отчет должен быть правильно истолкован и проанализирован квалифицированным специалистом, который несет ответственность за оценку!

Любая информация, представленная в этом отчете, не является окончательной и подлежит ручному просмотру и анализу. Пожалуйста, следуйте инструкциям: [Рекомендации по оценке](#)

Детектор Плагиата - Ваше право на оригинальность! ☐ SkyLine LLC