

Міністерство освіти і науки України
Державний заклад
«Луганський національний університет імені Тараса Шевченка»

Навчально-науковий інститут математики та інформаційних технологій

Кафедра інформаційних технологій та систем

Озеров Данііл Віталійович

**РОЗРОБКА WINDOWS FORMS-ДОДАТКУ ДЛЯ ПЛАНУВАННЯ ТА
ОБЛІКУ СПОРТИВНИХ ТРЕНУВАНЬ**

кваліфікаційна робота

**здобувача вищої освіти першого (бакалаврського) рівня
освітньої програми «Інженерія програмного забезпечення»
за спеціальністю F2 Інженерія програмного забезпечення**

Особистий підпис _____ Данііл ОЗЕРОВ

Науковий керівник _____ Микола СЕМЕНОВ,
кандидат педагогічних наук, доцент
кафедри інформаційних технологій
та систем

Завідувач кафедри _____ Микола СЕМЕНОВ,
кандидат педагогічних наук, доцент
кафедри інформаційних технологій
та систем

Полтава – 2025

Міністерство освіти і науки України
Державний заклад „Луганський національний університет
імені Тараса Шевченка”

Інститут Навчально-науковий інститут математики та
інформаційних технологій
Кафедра, циклова комісія Кафедра інформаційних технологій та систем
Рівень освіти перший (бакалаврський)
Напрямок підготовки (спеціальність) 121 «Інженерія програмного забезпечення»
(код, назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТС
М.А. Семенов

(підпис) (ініціали, прізвище)
“ ” 2024 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Озерова Данііла Віталійовича
(прізвище, ім'я, по батькові)

**1. Тема проекту (роботи) Розробка Windows Forms-додатку для
планування та обліку спортивних тренувань**

Керівник кваліфікаційної роботи Семенов М.А.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по університету Від“ ” 2024 року №_

2. Строк подання студентом проекту (роботи)
3. Вихідні дані до роботи (проекту) у результаті виконання роботи
повинно бути розроблено Windows Forms-додаток для планування та обліку
спортивних тренувань

(визначаються кількісні або (та) якісні показники, яким повинен відповідати об'єкт розробки)

**4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно
розробити) МЕТОДОЛОГІЯ РОЗРОБКИ ЗАСТОСУНКУ
РОЗРОБКА WINDOWS FORMS-додатку**

(визначаються назви розділів або (та) перелік питань, які повинні увійти до тексту ПЗ)

**5. Перелік графічного матеріалу (з точним зазначенням обов'язкових
креслень)**

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання „_____” _____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
	Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника.	До 15 жовтня	
	Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно-експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень.	Другий тиждень листопада (10 листопада)	
	Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником.	До 15 грудня	
	Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання.	До 28 січня	
	Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи.	Перший тиждень березня	
	Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації.	До 31 березня	
	Попередній захист роботи на кафедрі	квітень	
	Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії	За 10 днів до державної атестації	
	Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом.	За 5 днів до державної атестації	

Студент

підпис

Д.В. Озеров

(ініціали, прізвище)

Керівник проекту (роботи)

підпис

М.А. Семенов

АНОТАЦІЯ

Озеров Д. В.

Тема: Розробка Windows Forms-додатку для планування та обліку спортивних тренувань.

Спеціальність: F2 "Інженерія програмного забезпечення"

Установа: ЛНУ імені Тараса Шевченка, 2025 р.

Бакалаврська робота містить: 85 с., 29 рис., 5 табл., 7 дод., 18джерел.

Об'єкт дослідження – програмні засоби підтримки фізичної активності.

Предмет дослідження – методи та засоби створення Windows Forms-застосунків для планування спортивних тренувань

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..

Результати роботи.. У роботі досліджено особливості традиційних та цифрових підходів до ведення щоденників тренувань, проведено аналіз існуючих програмних аналогів та методик тренувань (періодизація, інтервальне тренування тощо). Визначено вимоги до системи та розроблено архітектуру застосунку. Реалізовано функціональний Windows Forms-додаток TrainMind з такими модулями: планувальник тренувань, журнал активності, графічна аналітика, система рекомендацій та експорт звітів у PDF/Excel. Застосунок працює без підключення до мережі Інтернет та використовує SQLite як вбудовану СУБД. Проведено тестування, що підтвердило відповідність програмного продукту функціональним вимогам.

Висновки. Розроблений програмний продукт може бути рекомендований спортсменам-аматорам, тренерам і викладачам фізичної культури для підтримки індивідуального тренувального процесу.

Ключові слова. ТРЕНУВАННЯ, СПОРТ, АВТОМАТИЗАЦІЯ ПРОЦЕСІВ, WINDOWS FORM

ABSTRACT

Ozerov Daniil

Theme: Development of a Windows Forms Application for Planning and Tracking Sports Training.

Speciality: "F2 Software Engineering"

Institution: Luhansk Taras Shevchenko National University (LTSNU), 2025.

Bachelor's Thesis consists of: 85 pages, 29 Fig., 5 tbls., 7 adj., 18 source.

Object of research – the process of digital support of an individual sports training program.

Subject of research – methods and tools for implementing a desktop application for training planning and tracking.

The aim of research is to develop an autonomous Windows Forms-based desktop application for planning, recording, and analyzing sports training sessions with support for data export.

Research results. The thesis investigates traditional and digital approaches to training journal management, analyzes existing training software and methods (e.g., periodization, interval training), defines functional requirements, and designs the system architecture. A fully functional Windows Forms application, TrainMind, was implemented. The application includes modules for training planning, session logging, analytical visualization, intelligent recommendations, and exporting reports in PDF and Excel formats. The system operates offline using SQLite as an embedded database. Testing confirmed compliance with the declared functional specifications.

Conclusion. The developed software product is recommended for amateur athletes, coaches, and physical education professionals to support and manage personal training processes.

Keywords. TRAINING, SPORTS, PROCESS AUTOMATION, WINDOWS FORM

ІТС. ІПЗ4.1225-ВП
РОЗРОБКА WINDOWS FORMS-ДОДАТКУ ДЛЯ ПЛАНУВАННЯ ТА
ОБЛІКУ СПОРТИВНИХ ТРЕНУВАНЬ

[illegible]

Міністерство освіти і науки України
Державний заклад «Луганський національний університет
імені Тараса Шевченка»
Факультет (інститут) Навчально-науковий інститут математики та
інформаційних технологій
(повна назва)
Кафедра Кафедра інформаційних технологій та систем
(повна назва)

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання програмної розробки (ПР) :
"РОЗРОБКА WINDOWS FORMS-ДОДАТКУ
ДЛЯ ПЛАНУВАННЯ ТА ОБЛІКУ
СПОРТИВНИХ ТРЕНУВАНЬ"

ІТС. ІПЗ4.1225-02-ТЗ

ПОГОДЖЕНО
Керівник кваліфікаційної роботи

Семенов М. А.

“ ” 2025р.

ВИКОНАВЕЦЬ
Студент групи 4 ІПЗ

Озеров Д. В.

“ ” 2025р.

Полтава 2025

ЗМІСТ

ВСТУП	3
1. ХАРАКТЕРИСТИКА ОБ'ЄКТА	3
2. ПРИЗНАЧЕННЯ ПРОДУКТУ	3
3. ОСНОВНІ ВИМОГИ ДО ПРОГРАМНОГО КОМПЛЕКСУ	3
4. ТЕХНІКО-ЕКОНОМІЧНІ ВИМОГИ	4
5. ВИМОГИ ДО КОМПЛЕКТУЮЧИХ І МАТЕРІАЛІВ	4
6. ЕТАПИ ВИКОНАННЯ ПР	5
7. ПРИЙОМ І ОЦІНКА РЕЗУЛЬТАТІВ	5
8. ПОРЯДОК ВНЕСЕННЯ ЗМІН ДО ТЗ	5

ВСТУП

Найменування: Десктопний застосунок TrainMind для планування та обліку спортивних тренувань.

Шифр ПР: TRM-1

Підстава для виконання ПР: Потреба в автономному програмному інструменті для планування тренувального процесу, обліку фізичної активності, візуалізації прогресу та формування звітів без підключення до мережі Інтернет.

Термін розробки:

Початок: 1 жовтня 2024 р.

Завершення: 1 травня 2025 р.

1. ХАРАКТЕРИСТИКА ОБ'ЄКТА

Об'єктом розробки є настільний програмний продукт, який дозволяє створювати індивідуальні плани тренувань, вести журнал виконаних сесій, переглядати статистику у вигляді графіків та експортувати дані у вигляді звітів.

2. ПРИЗНАЧЕННЯ ПРОДУКТУ

2.1. Призначення — персональний інструмент для спортсменів та тренерів для планування та контролю тренувального процесу в автономному режимі.

2.2. Основні критерії ефективності:

зручний інтерфейс (Windows Forms);

можливість роботи без підключення до Інтернету;

наявність графічної аналітики та зручних форм для введення даних;

підтримка експорту у формати PDF та Excel.

3. ОСНОВНІ ВИМОГИ ДО ПРОГРАМНОГО КОМПЛЕКСУ

3.1. Загальні вимоги

ОС: Windows 10/11;

Платформа: .NET 6 або вище;

Тип додатку: десктопний (Windows Forms);

Зберігання даних: SQLite (локальна база).

3.2. Архітектура

Використання ORM: Entity Framework Core;

Поділ на модулі: планування, журнал, аналітика, експорт, налаштування профілю;

Автономний запуск без зовнішніх серверів.

3.3. Основний функціонал:

створення та редагування індивідуальних планів тренувань;

ведення щоденника тренувань з фіксацією параметрів (обсяг, тривалість, пульс тощо);

побудова графіків прогресу;

автоматичні рекомендації на основі історії тренувань;

експорт даних у звіти (PDF/Excel);

налаштування профілю користувача.

3.4. Додаткові вимоги:

підтримка копіювання тренувань;

кешування аналітики;

журнал експорту;

стійкість до помилок користувача (валідація даних).

4. ТЕХНІКО-ЕКОНОМІЧНІ ВИМОГИ

Вартість розробки не визначається, оскільки продукт створюється в рамках освітнього проекту.

Вибрані бібліотеки — з відкритими ліцензіями (EPPlus, PdfSharp, ScottPlot тощо).

5. ВИМОГИ ДО КОМПЛЕКТУЮЧИХ І МАТЕРІАЛІВ

5.1. Додаткові апаратні вимоги — не передбачаються.

5.2. Вимоги до екологічної та техногенної безпеки — не висуваються.

6. ЕТАПИ ВИКОНАННЯ ПР

№	Етап	Строк виконання	Звітні матеріали
1	Аналіз та моделювання	01.10.2024 — 15.11.2024	Структура БД, діаграми, опис алгоритмів
2	Реалізація ядра функцій	16.11.2024 — 01.01.2025	Функціонуючі модулі, початковий код
3	Реалізація інтерфейсу	01.01 – 15.02.2025	Форми, навігація, збереження даних
4	Тестування, аналітика	16.02 – 10.03.2025	Звіти про помилки, графіки
5	Документація та завершення	11.03 – 1.05.2025	Керівництво, звіти, ТЗ

7. ПРИЙОМ І ОЦІНКА РЕЗУЛЬТАТІВ

7.1. Для приймання пред'являються:

робоча версія застосунку на ПК;

база даних з прикладами;

технічна документація: технічне завдання, пояснювальна записка, інструкція користувача, програма та методика тестування;

перелік файлів на цифровому носії.

7.2. Замовник (керівник або комісія) здійснює оцінку функціоналу та якості реалізації згідно з цим ТЗ.

8. ПОРЯДОК ВНЕСЕННЯ ЗМІН ДО ТЗ

У разі виникнення необхідності внесення змін до функціоналу, архітектури або інтерфейсу в процесі розробки — допускається уточнення ТЗ

за погодженням між виконавцем та керівником роботи з обов'язковим оформленням доповнення до ТЗ.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЗ «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 «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

ВСТУП.....	3
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОГРАМНИХ РІШЕНЬ	5
1.1. Особливості організації індивідуального тренувального процесу та потреби у цифрових інструментах	5
1.2. Огляд та аналіз існуючих алгоритмів, методик і схем тренувань	9
1.3. Порівняльний аналіз функціоналу, переваг і недоліків аналогів	14
1.4. Вимоги до локального Windows-додатку для обліку фізичної активності	21
РОЗДІЛ 2. ПРОЄКТУВАННЯ МОДЕЛІ WINDOWS FORMS-ДОДАТКУ ..	25
2.1. Архітектура програмного забезпечення: головні компоненти та їх зв'язки	25
2.2. Концептуальна модель бази даних: таблиці, зв'язки, логіка збереження..	28
2.3. Розробка інтерфейсної структури додатку: головне меню, форма тренування, статистика.....	34
2.4. Вибір технологій: C#, Windows Forms, SQLite, додаткові бібліотеки	39
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ	41
3.1. Реалізація базових функцій: створення, редагування та перегляд тренувань	41
3.2. Реалізація аналітики: побудова графіків прогресу (час, дистанція, навантаження).....	48
3.3. Експорт даних у звіт (формати Excel, PDF)	52
3.4. Тестування, перевірка коректності роботи, приклади використання	55
ЗАГАЛЬНІ ВИСНОВКИ	58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	60
ДОДАТКИ.....	63

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:

1. Проаналізувати існуючі рішення та вимоги до систем обліку спортивних тренувань.
2. Розробити концептуальну модель додатку, включаючи базу даних, інтерфейс і логіку.
3. Реалізувати функціональний прототип програми з використанням C# та Windows Forms.
4. 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 проєктуванню 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

1.1. 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:

Таблиця 1.1

Формат паперового щоденнику тренувань

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.

Таблиця 1.2

Формат журналу тренувань

Тиждень	Понеділок	Вівторок	Середа	Четвер	П'ятниця	Субота	Неділя
1	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:

Таблиця 1.3

Формат тренувальної картки

Вправа	Підхід 1	Підхід 2	Підхід 3	Примітки
Присідання	12х50 кг	10х55 кг	8х60 кг	важко
Жим лежачи	12х40 кг	10х45 кг	8х50 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 методики 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.

1.2. 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

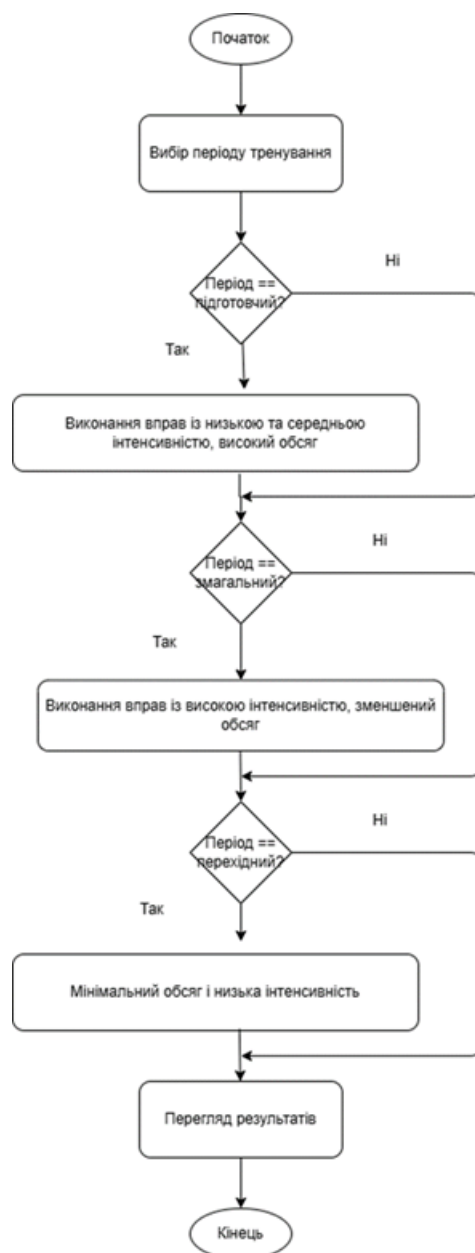


Рис. 1.1 Блок-схема алгоритму класичної періодизації

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).

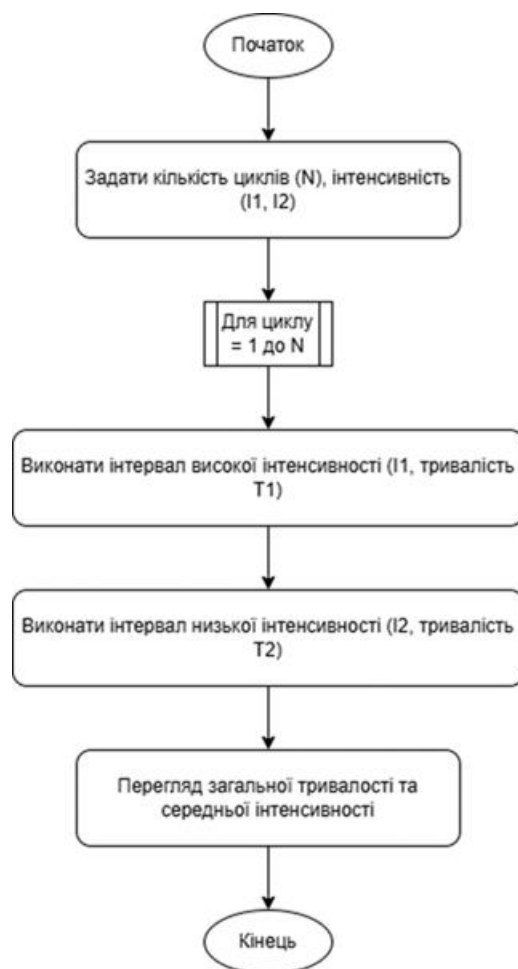


Рис. 1.2 Блок-схема алгоритму інтервального методу

Алгоритм інтервального тренування:

Початок

Задати кількість циклів (N), інтенсивність (I1, I2)

Для циклу = 1 до N:

Виконати інтервал високої інтенсивності (I1, тривалість T1)

Виконати інтервал низької інтенсивності (I2, тривалість T2)

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 А

День 2: Група м'язів В

День 3: Група м'язів С

...

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

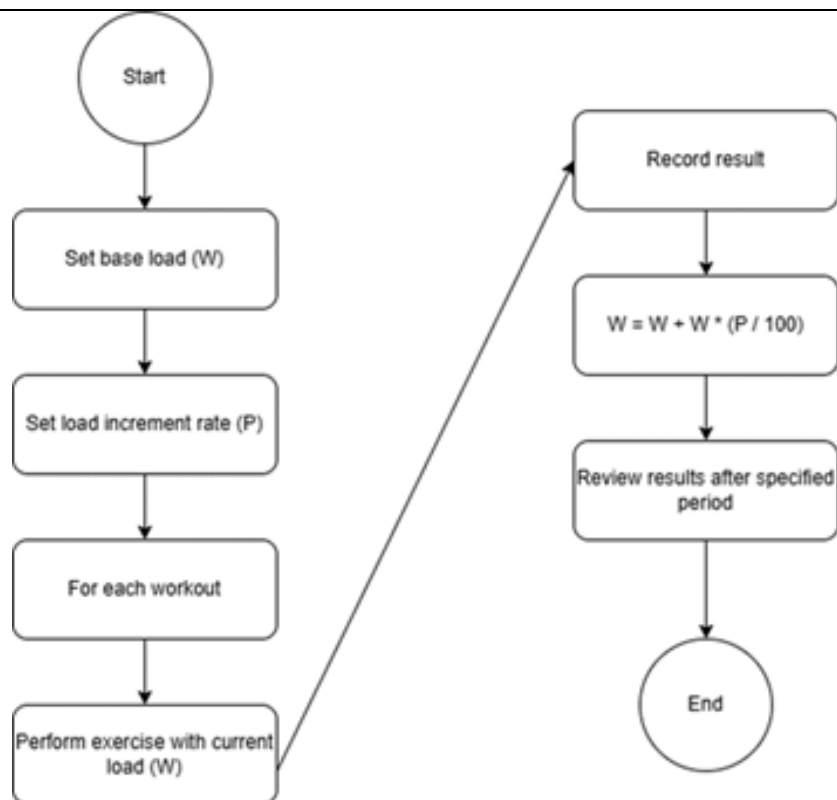


Рис. 1.4 Блок-схема алгоритму прогресивного навантаження

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/>) (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

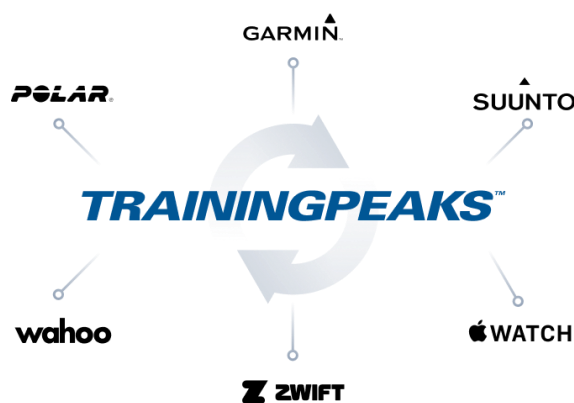


Рис. 1.6 Інтеграція TrainingPeaks з гаджетами



Рис. 1.7 Щоденник TrainingPeaks

Як бачимо з рис. 1.7 тренування розфарбовуються в залежності від відповідності результатів запланованим параметрам.

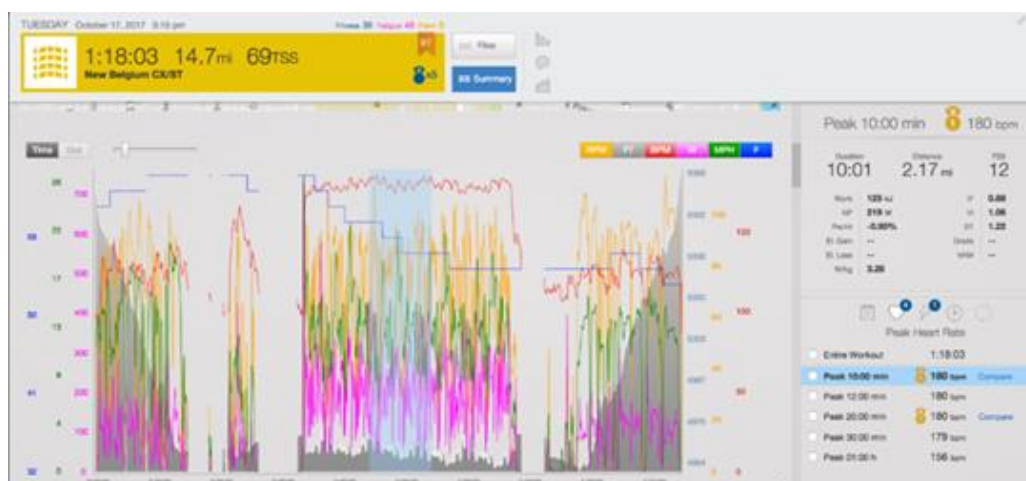


Рис. 1.8 Аналітичні графіки TrainingPeaks

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 градуйований темп (NGP). У велоспорті функція «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/>) — 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

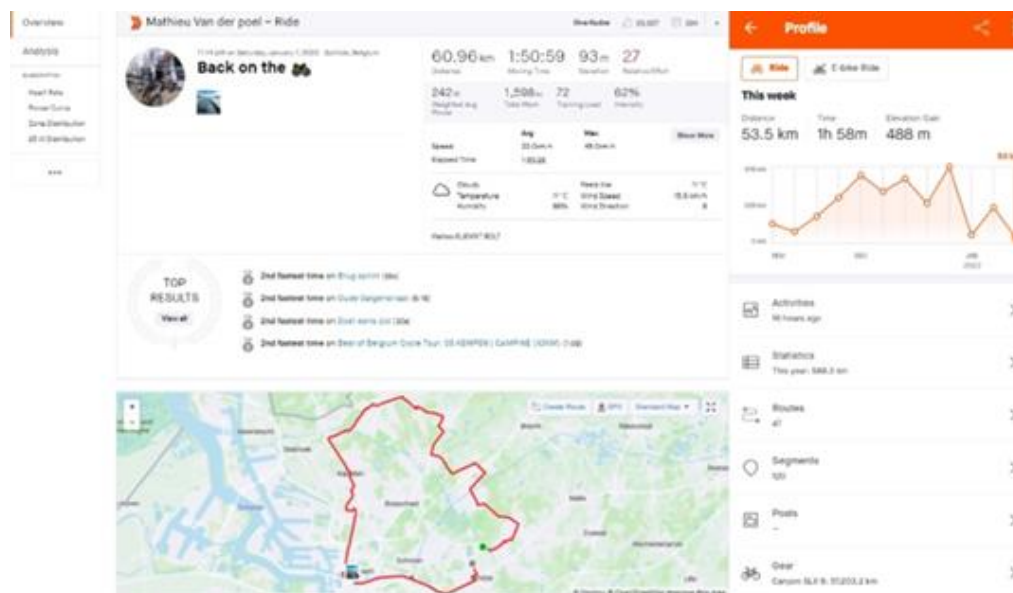


Рис. 1.10 Зовнішній вигляд веб сторінки з результатами тренування

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.

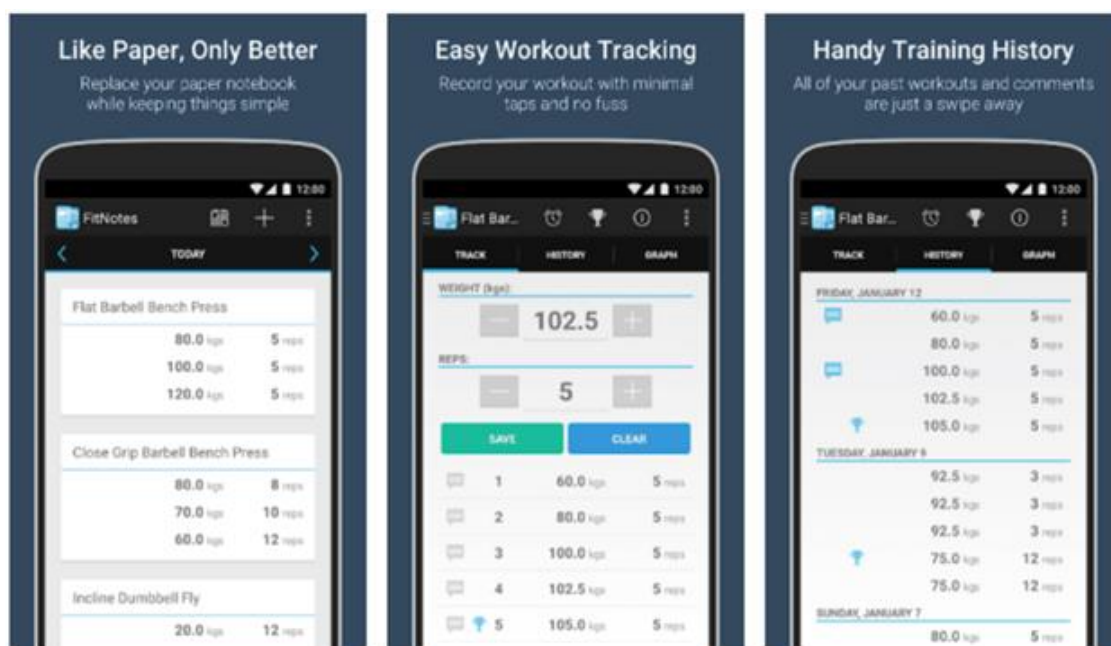


Рис. 1.11 FitNotes

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 платформ	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:

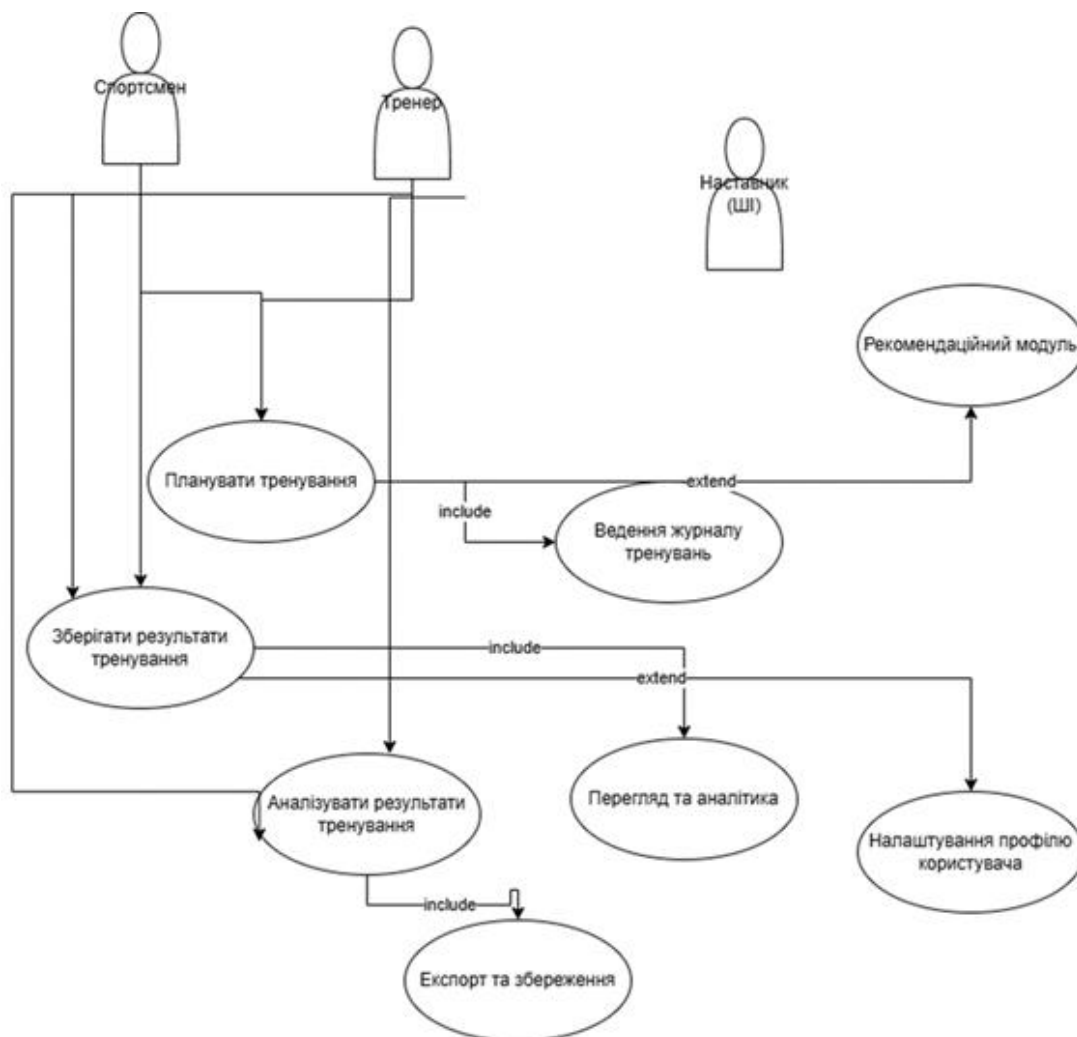


Рис. 1.12 Use Case діаграма застосунку

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 «include» та «extend».

На основі рис. 1.12 деталізуємо функціональні вимоги до застосунку:

1. 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:

«Планування»

«Журнал»

«Аналітика»

«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 ІІІ 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 ІІІ
- Exporter – модуль експорту
- StorageManager – робота з локальною БД

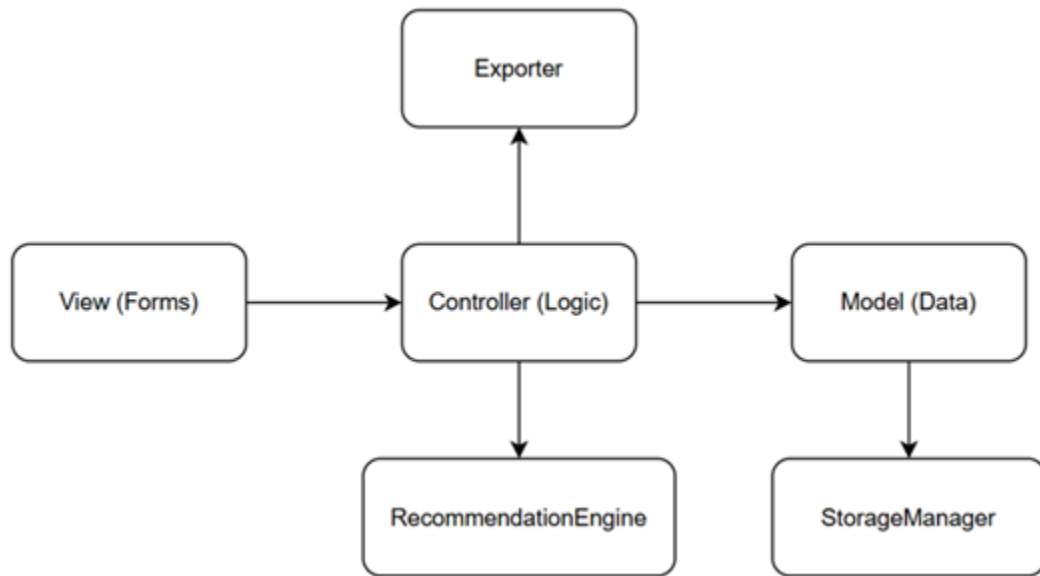


Рис. 2.1 Архітектура Windows Forms додатку (MVC)

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, “зберегти тренування”, “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 ІІІ, 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

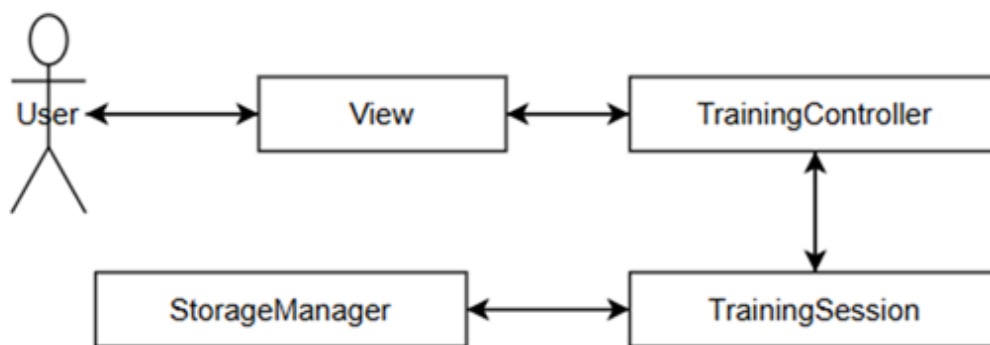


Рис. 2.2 Діаграма послідовності для архітектури застосунку

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.

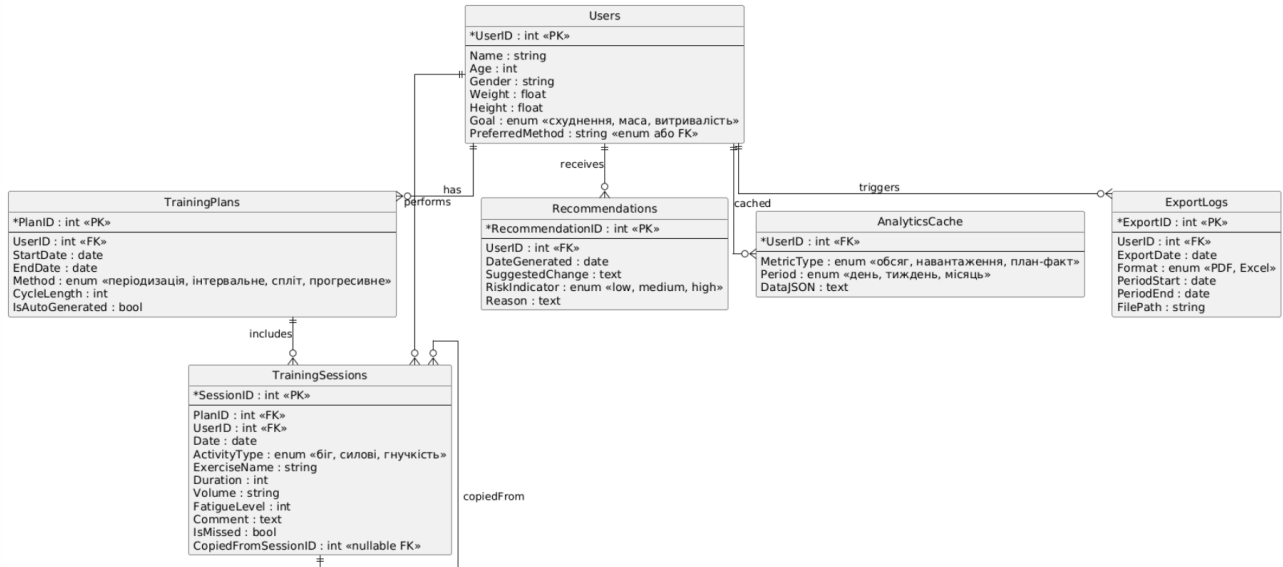


Рис. 2.3 ER діаграма бази даних

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

Таблиця	Зв'язок	Призначення
TrainingSessions	опціонально → TrainingSessions	Перенесення або копіювання тренування
Users → Recommendations	$1 \rightarrow \infty$	Рекомендації, згенеровані для користувача
Users → AnalyticsCache	$1 \rightarrow \infty$	Кеш для відображення метрик
Users → ExportLogs	$1 \rightarrow \infty$	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:

- $1 \rightarrow \infty$ до TrainingPlans (один користувач може мати багато планів).
- $1 \rightarrow \infty$ до TrainingSessions (один користувач має багато тренувань).
- $1 \rightarrow \infty$ до Recommendations, ExportLogs, AnalyticsCache.

Users
*UserID : int «PK»
Name : string
Age : int
Gender : string
Weight : float
Height : float
Goal : enum «схуднення, маса, витривалість»
PreferredMethod : string «enum або FK»

Рис. 2.4 Таблиця «Uers»

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.

TrainingPlans
*PlanID : int «PK»
UserID : int «FK»
StartDate : date
EndDate : date
Method : enum «періодизація, інтервальне, спліт, прогресивне»
CycleLength : int
IsAutoGenerated : bool

Рис. 2.5 Таблиця «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.

Recommendations
*RecommendationID : int «PK»
UserID : int «FK» DateGenerated : date SuggestedChange : text RiskIndicator : enum «low, medium, high» Reason : text

Рис. 2.6 Таблиця «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.

TrainingSessions
*SessionID : int «PK»
PlanID : int «FK»
UserID : int «FK»
Date : date
ActivityType : enum «біг, силові, гнучкість»
ExerciseName : string
Duration : int
Volume : string
FatigueLevel : int
Comment : text
IsMissed : bool
CopiedFromSessionID : int «nullable FK»

Рис. 2.7 Таблиця «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).

AnalyticsCache
*UserID : int «FK»
MetricType : enum «обсяг, навантаження, план-факт»
Period : enum «день, тиждень, місяць»
DataJSON : text

Рис. 2.8 Таблиця «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.

ExportLogs
*ExportID : int «PK»
UserID : int «FK»
ExportDate : date
Format : enum «PDF, Excel»
PeriodStart : date
PeriodEnd : date
FilePath : string

Рис. 2.9 Таблиця «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.

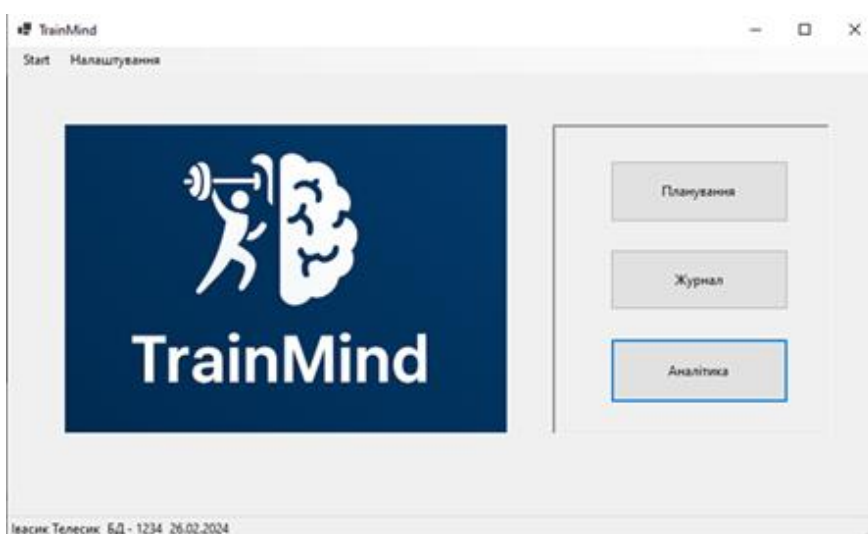


Рис.2.10 Інтерфейс головної форми MainForm

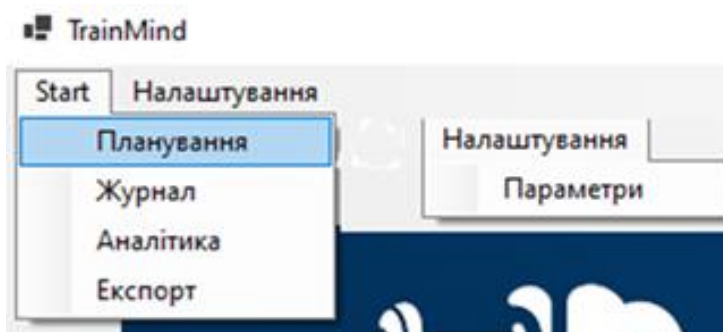


Рис. 2.11 Меню MainForm

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 «Згенерувати план автоматично»;
- кнопки "Зберегти", "Очистити", "Завантажити шаблон".

Для кожного дня плану відображається таблиця із попередньо заповненими сесіями або можливістю вручну додати тип активності.

PlanID	UserID	StartDate	EndDate
1	1	01.01.2023	31.03.2023
2	2	01.02.2023	30.04.2023

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.

TrainingJournalForm

User: John Doe

	Date	ActivityType	ExerciseName	Duration
▶	02.01.2023	Running	Morning Run	30
	04.01.2023	Weightlifting	Chest Press	45

Назва вправи:

Тип активності:

Тривалість:

Обсяг:

Рівень втоми:

Коментар:

Save Cancel

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.

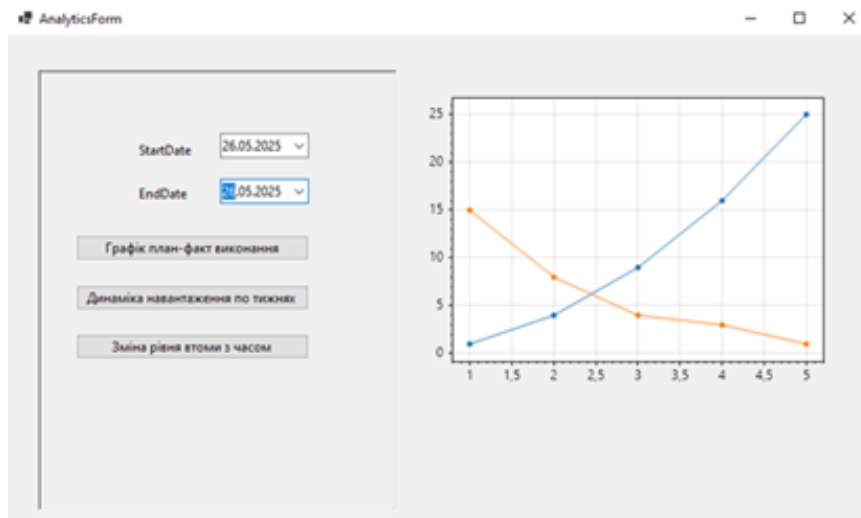


Рис. 2.14 – Графічний інтерфейс форми аналітики AnalyticsForm]

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.

	Format	PeriodStart
▶	PDF	01.01.2023
	Excel	01.02.2023

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 "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 враховані 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 проекту TrainMind. У результаті було реалізовано структуру, як зображено на рис. 3.1.

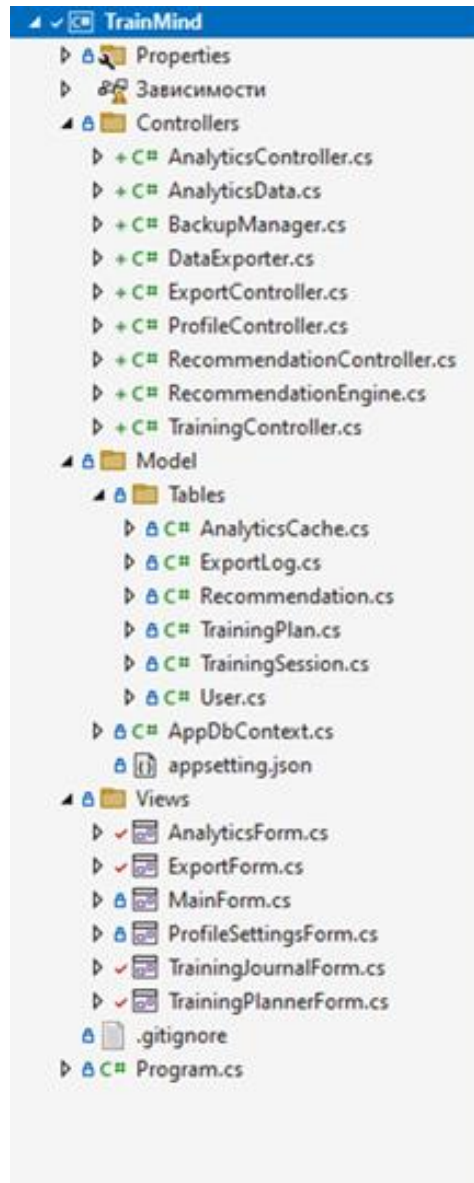


Рис. 3.1 Структура застосунку TrainMind

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 "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 "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("Будь ласка, оберіть хоча б одну методику.");
    return;
}

if (numericUpDown1.Value == 0)
{
    MessageBox.Show("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(", ", methods),
        IsAutoGenerated = false
    };

    using (var db = new AppDbContext())
    {
        db.TrainingPlans.Add(newPlan);
        db.SaveChanges();
    }
    Refresh();

    MessageBox.Show("План тренувань успішно збережено!");

```

Крім того, передбачено можливість видалення створених планів. Після вибору відповідного запису в таблиці та підтвердження дії, запис видалється 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 = $"{p.PlanID}"
            })
            .ToList();
        comboBox3.DataSource = plans;
        comboBox3.DisplayMember = "DisplayText"; // що бачить користувач
        comboBox3.ValueMember = "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:

1. 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.

2. 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.

```
end) public static AnalyticsData GetLoadByDate(int userId, DateTime start, DateTime
{
    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
            })
            .ToList();

        return new AnalyticsData
        {
            Dates = grouped.Select(g => g.Date).ToList(),
            Loads = grouped.Select(g => (float)g.Load).ToList()
        };
    }
}
```

3. 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($"Знайдено {data.Dates.Count} записів з
{start.ToShortDateString()} по {end.ToShortDateString()}.");
    else
    {
        double[] x = data.Dates.Select(d => d.ToOADate()).ToArray(); // Convert
DateTime to OLE Automation Date for plotting
        formsPlot1.Plot.Clear();
        formsPlot1.Plot.Axes.SetLimits(start.ToOADate(), end.ToOADate(), 0,
data.Loads.Max() * 1.1);
        formsPlot1.Plot.Add.Scatter(x, data.Loads.ToArray());
        formsPlot1.Plot.XLabel("Дата");
        formsPlot1.Plot.YLabel("Виконання");
        formsPlot1.Plot.Title($"Тренування з {start.ToShortDateString()} по
{end.ToShortDateString()}");
        formsPlot1.Refresh();
    }
}

```

Натискання кнопки button2 формує графік на основі обчислення добутку $\text{Volume} \times \text{Duration}$ 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, чію активність потрібно експортувати. Таблиця `dataGridView1` заповнюється журналом попередніх експортів з таблиці `ExportLogs.2`. Резервне копіювання 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 = "Training Sessions Report";

PdfPage page = document.AddPage();
XGraphics gfx = XGraphics.FromPdfPage(page);
GlobalFontSettings.UseWindowsFontsUnderWindows = true;
XFont font = new XFont("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($"Date: {session.Date:yyyy-MM-dd}", font, XBrushes.Black, new
XRect(40, y, page.Width, 20));
    y += 15;
    gfx.DrawString($"Activity: {session.ActivityType}", font, XBrushes.Black, new
XRect(40, y, page.Width, 20));
    y += 15;
    gfx.DrawString($"Exercise: {session.ExerciseName}", font, XBrushes.Black, new
XRect(40, y, page.Width, 20));
    y += 15;
    gfx.DrawString($"Duration: {session.Duration} min", font, XBrushes.Black, new
XRect(40, y, page.Width, 20));
    y += 15;
    gfx.DrawString($"Volume: {session.Volume}", font, XBrushes.Black, new
XRect(40, y, page.Width, 20));
    y += 15;
    gfx.DrawString($"Fatigue: {session.FatigueLevel}", font, XBrushes.Black, new
XRect(40, y, page.Width, 20));
    y += 15;
    gfx.DrawString($"Comment: {session.Comment}", font, XBrushes.Black, new
XRect(40, y, page.Width, 20));
    y += 30;
}

document.Save(filePath2);
Process.Start("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 «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 «Налаштування профілю», вводить свої дані (стать, вік, мета — наприклад, «витривалість»), створює план тренувань на місяць з методом «періодизація». Після цього він заповнює журнал, щодня додаючи активність.

Сценарій 2. Автоматичне оновлення графіків

Користувач додає два тренування у вигляді інтервальних сесій. У вкладці «Аналітика» автоматично відображається графік навантаження за тиждень. Після редагування одного з тренувань графік оновлюється миттєво.

Сценарій 3. Формування звіту для тренера

Користувач натискає «Експорт», 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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Горбань Г. В., Кандиба І. О., Фісун М. Т. ПЕРВИННА ОБРОБКА ДАНИХ РЕЗУЛЬТАТІВ СПОРТИВНИХ ТРЕНУВАНЬ СТУДЕНТІВ НА ВЕСЛУВАЛЬНИХ ТРЕНАЖЕРАХ CONCERT2 ДЛЯ ПОДАЛЬШОГО АНАЛІЗУ ЗА ДОПОМОГОЮ БІБЛІОТЕКИ PANDAS. *Таврійський науковий вісник. Серія: Технічні науки*. 2023. № 2. С. 33–44. URL: <https://doi.org/10.32782/tnv-tech.2023.2.4> (дата звернення: 29.05.2025).
2. Application of Automatic/Controlled Processing Theory to Training Tactical Command and Control Skills: II. Evaluation of a Task Analytic Methodology / F. T. Eggemeier et al. *Proceedings of the Human Factors Society Annual Meeting*. 1988. Vol. 32, no. 18. P. 1232–1236. URL: <https://doi.org/10.1177/154193128803201809> (date of access: 29.05.2025).
3. Bartlett R. Introduction to Sports Biomechanics. Routledge, 2014. URL: <https://doi.org/10.4324/9781315889504> (date of access: 29.05.2025).
4. Bartlett R. Principles of Throwing. *Biomechanics in Sport*. Oxford, UK. P. 365–380. URL: <https://doi.org/10.1002/9780470693797.ch18> (date of access: 29.05.2025).
5. Berkoff S. Health Education Programing. *Journal of Health, Physical Education, Recreation*. 1966. Vol. 37, no. 2. P. 63–64. URL: <https://doi.org/10.1080/00221473.1966.10618545> (date of access: 29.05.2025).
6. Bompa T., Buzzichelli C. Periodization: Theory and Methodology of Training. Human Kinetics, 2018. 392 p.
7. Bompa T. O. Periodizacion/ Periodization: Teoria y metodologia del entrenamiento/ Theory and Methodology of Training. Hispano Europea Editorial, 2003. 429 p.

8. Features of planning multi- year sports training in sports swimming / M. Kruk et al. *Scientific Journal of National Pedagogical Dragomanov University. Series 15. Scientific and pedagogical problems of physical culture (physical culture and sports)*. 2023. No. 2(160). P. 124–129. URL: [https://doi.org/10.31392/npu-nc.series15.2023.02\(160\).26](https://doi.org/10.31392/npu-nc.series15.2023.02(160).26) (date of access: 29.05.2025).
9. Jensen A. R. Industry in Programing. *PsycCRITIQUES*. 1962. Vol. 7, no. 1. URL: <https://doi.org/10.1037/006739> (date of access: 29.05.2025).
10. Kostiuk T. O. THE EXPERIENCE OF TRAINING EXPERTS IN PUBLIC ADMINISTRATION AND DECISION-MAKING FIELDS. *PUBLIC AND MUNICIPAL ADMINISTRATION: THEORY, METHODOLOGY, PRACTICE*. 2020. P. 96–113. URL: <https://doi.org/10.30525/978-9934-588-46-4.06> (date of access: 29.05.2025).
11. Martins A. S., Carneiro F. S., Oliveira A. L. d. Estudo comparativo em atletas amadores do triathlon de curta e longa distância na organização, planejamento e desempenho de suas periodizações na plataforma trainingpeaks. *Educação Física, Esporte, Cultura & Sociedade*. 2022. P. 151–164. URL: <https://doi.org/10.51360/zh4.20221-01-p.151-164> (date of access: 29.05.2025).
12. Planning the sports training sessions with the bat algorithm / I. Fister et al. *Neurocomputing*. 2015. Vol. 149. P. 993–1002. URL: <https://doi.org/10.1016/j.neucom.2014.07.034> (date of access: 29.05.2025).
13. Rushall B. S., Pyke F. S. Planning a Sporting Career. *Training for Sports and Fitness*. London, 1990. P. 349–365. URL: https://doi.org/10.1007/978-1-349-15135-6_20 (date of access: 29.05.2025).
14. Rushall B. S., Pyke F. S. Planning a Training Year. *Training for Sports and Fitness*. London, 1990. P. 328–348. URL: https://doi.org/10.1007/978-1-349-15135-6_19 (date of access: 29.05.2025).

- 15.Skyrda T. S., Lazorenko N. L., Zhudro O. V. SECTION #4. VIRTUAL EDUCATIONAL ENVIRONMENT IN THE SYSTEM OF PROFESSIONAL FOREIGN LANGUAGE TRAINING 4.2 EDUCATION OF PATRIOTISM IN INTERNATIONAL STUDENTS THROUGH GAMIFICATION. *CURRENT THEORY AND PRACTICE ASPECTS OF LINGUISTICS, SOCIOLINGUISTICS AND METHODOLOGY OF FOREIGN LANGUAGES AT UNIVERSITIES IN MODERN GLOBAL HIGHER EDUCATIONAL SPACE*. 2022. URL: <https://doi.org/10.31435/rsglobal/052-8> (date of access: 29.05.2025).
- 16.Sports Council for Wales. Policy Planning Section. New education and training structures: Implications for sport and leisure. Cardiff : Sports Council for Wales, 1992.
- 17.Verkhoshansky Y., Charniga A. Fundamentals of Special Strength Training in Sport. Primedia eLaunch LLC, 2019.
- 18.Young W. RESISTANCE TRAINING: The planning of resistance training for power sports. *National Strength & Conditioning Association Journal*. 1991. Vol. 13, no. 4. P. 26. URL: [https://doi.org/10.1519/0744-0049\(1991\)013%3C0026:tporf%3E2.3.co;2](https://doi.org/10.1519/0744-0049(1991)013%3C0026:tporf%3E2.3.co;2) (date of access: 29.05.2025).

ДОДАТКИ

Додаток А

Код структури бази даних TrainMind

```

using System.Reflection.Emit;
using System.Text;
using System.Threading.Tasks;
using TrainMind.Model.Classes;
using Microsoft.EntityFrameworkCore;

namespace TrainMind.Model
{
    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; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            // Users
            modelBuilder.Entity<User>()
                .HasKey(u => u.UserID);

            // TrainingPlans
            modelBuilder.Entity<TrainingPlan>()
                .HasKey(tp => tp.PlanID);

            modelBuilder.Entity<TrainingPlan>()
                .HasOne(tp => tp.User)
                .WithMany(u => u.TrainingPlans)
                .HasForeignKey(tp => tp.UserID)
                .OnDelete(DeleteBehavior.Cascade);

            // TrainingSessions
            modelBuilder.Entity<TrainingSession>()
                .HasKey(ts => ts.SessionID);
        }
    }
}

```

```

modelBuilder.Entity<TrainingSession>()
    .HasOne(ts => ts.User)
    .WithMany(u => u.TrainingSessions)
    .HasForeignKey(ts => ts.UserID)
    .OnDelete(DeleteBehavior.Cascade);

modelBuilder.Entity<TrainingSession>()
    .HasOne(ts => ts.TrainingPlan)
    .WithMany(tp => tp.TrainingSessions)
    .HasForeignKey(ts => ts.PlanID)
    .OnDelete(DeleteBehavior.Cascade);

modelBuilder.Entity<TrainingSession>()
    .HasOne(ts => ts.CopiedFromSession)
    .WithMany()
    .HasForeignKey(ts => ts.CopiedFromSessionID)
    .OnDelete(DeleteBehavior.Restrict);

// Recommendations
modelBuilder.Entity<Recommendation>()
    .HasKey(r => r.RecommendationID);

modelBuilder.Entity<Recommendation>()
    .HasOne(r => r.User)
    .WithMany(u => u.Recommendations)
    .HasForeignKey(r => r.UserID)
    .OnDelete(DeleteBehavior.Cascade);

// AnalyticsCache
modelBuilder.Entity<AnalyticsCache>()
    .HasKey(a => a.AnalyticsCacheID);

modelBuilder.Entity<AnalyticsCache>()
    .HasOne(a => a.User)
    .WithMany(u => u.AnalyticsCaches)
    .HasForeignKey(a => a.UserID)
    .OnDelete(DeleteBehavior.Cascade);

// ExportLogs
modelBuilder.Entity<ExportLog>()
    .HasKey(e => e.ExportID);

modelBuilder.Entity<ExportLog>()
    .HasOne(e => e.User)

```

```

        .WithMany(u => u.ExportLogs)
        .HasForeignKey(e => e.UserID)
        .onDelete(DeleteBehavior.Cascade);
    }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlite("Data Source=TrainMind.db");
    }
}

internal class AnalyticsCache
{
    public int AnalyticsCacheID { get; set; }
    public int UserID { get; set; }
    public string MetricType { get; set; } = string.Empty;
    public string Period { get; set; } = string.Empty;
    public string DataJSON { get; set; } = string.Empty;
    public User User { get; set; }
}

internal class ExportLog
{
    public int ExportID { get; set; }
    public int UserID { get; set; }
    public DateTime ExportDate { get; set; }
    public string Format { get; set; } = string.Empty;
    public DateTime PeriodStart { get; set; }
    public DateTime PeriodEnd { get; set; }
    public string FilePath { get; set; }

    public User User { get; set; }
}

internal class Recommendation
{
    public int RecommendationID { get; set; }
    public int UserID { get; set; }
    public DateTime DateGenerated { get; set; }
    public string SuggestedChange { get; set; } = string.Empty;
    public string RiskIndicator { get; set; } = string.Empty;
}

```

```

    public string Reason { get; set; } = string.Empty;
    public User User { get; set; }
}

```

```

internal class TrainingPlan
{
    public int PlanID { get; set; }
    public int UserID { get; set; }
    public DateTime StartDate { get; set; }
    public DateTime EndDate { get; set; }
    public string Method { get; set; } = string.Empty;
    public int CycleLength { get; set; }
    public bool IsAutoGenerated { get; set; }

    public User User { get; set; }
    public ICollection<TrainingSession> TrainingSessions { get; set; }
}

```

```

internal class TrainingSession
{
    public int SessionID { get; set; }
    public int PlanID { get; set; }
    public int UserID { get; set; }
    public DateTime Date { get; set; }
    public string ActivityType { get; set; }
    public string ExerciseName { get; set; }
    public int Duration { get; set; }
    public float Volume { get; set; }
    public int FatigueLevel { get; set; }
    public string Comment { get; set; }
    public bool IsMissed { get; set; }
    public int? CopiedFromSessionID { get; set; }

    public User User { get; set; }
    public TrainingPlan TrainingPlan { get; set; }
    public TrainingSession CopiedFromSession { get; set; }
}

```

```

internal class User
{
    public int UserID { get; set; }
    public string Name { get; set; }
    public int Age { get; set; }
}

```

```
public string Gender { get; set; }
public float Weight { get; set; }
public float Height { get; set; }
public string Goal { get; set; }
public string PreferredMethod { get; set; }

public ICollection<TrainingPlan> TrainingPlans { get; set; }
public ICollection<TrainingSession> TrainingSessions { get; set; }
public ICollection<Recommendation> Recommendations { get; set; }
public ICollection<AnalyticsCache> AnalyticsCaches { get; set; }
public ICollection<ExportLog> ExportLogs { get; set; }
}
```

Код TrainingPlannerForm

```
namespace TrainMind.Views
{
    public partial class TrainingPlannerForm : Form
    {
        public TrainingPlannerForm()
        {
            InitializeComponent();
            using (var context = new AppDbContext())
            {
                // Load users into the combo box

                comboBox1.DataSource = context.Users.ToList();
                comboBox1.DisplayMember = "Name"; // Assuming User class has a Name property
                comboBox1.SelectedIndex = 0; // Select the first user by default

            }
            Refresh(); // Load training plans into the DataGridView
        }

        public void Refresh()
        {
            using (var context = new AppDbContext())
            {
                var tp = context.TrainingPlans.ToList();
                dataGridView1.DataSource = tp;
            }
        }

        private void checkBox6_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox8.Checked)
            {
                groupBox1.Enabled = false;
                checkBox1.Checked = false;
                checkBox2.Checked = false;
                checkBox3.Checked = false;
            }
        }
    }
}
```

```

        checkBox4.Checked = false;
    }
    else
    {
        groupBox1.Enabled = true;
    }
}

private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
{
    int days = (dateTimePicker2.Value.Date - dateTimePicker1.Value.Date).Days;
    numericUpDown1.Value = days >= 0 ? days : 0;

}

private void numericUpDown1_ValueChanged(object sender, EventArgs e)
{
    dateTimePicker2.Value =
dateTimePicker1.Value.AddDays((double)numericUpDown1.Value);
}

private void button1_Click(object sender, EventArgs e)
{

    // 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 III
    }

    if (methods.Count == 0)

```

```

    {
        MessageBox.Show("Будь ласка, оберіть хоча б одну методику.");
        return;
    }
    if (numericUpDown1.Value == 0)
    {
        MessageBox.Show("Будь ласка, оберіть період тренування.");
        return;
    }

    // MessageBox.Show(comboBox1.SelectedIndex.ToString());
    // Створення нового плану
    var newPlan = new TrainingPlan
    {
        UserID = comboBox1.SelectedIndex + 1, // або інший спосіб отримати поточного
користувача

        StartDate = startDate,
        EndDate = endDate,
        CycleLength = cycleLength,
        Method = string.Join(", ", methods),
        IsAutoGenerated = false
    };

    using (var db = new AppDbContext())
    {
        db.TrainingPlans.Add(newPlan);
        db.SaveChanges();
    }
    Refresh();

    MessageBox.Show("План тренувань успішно збережено!");
    // this.Close(); // або оновити форму
}

private void button2_Click(object sender, EventArgs e)
{
    if (dataGridView1.CurrentRow == null)
    {
        MessageBox.Show("Будь ласка, виберіть план для видалення.");
        return;
    }

```

```

// Отримання ID вибраного плану
int selectedPlanId = Convert.ToInt32(dataGridView1.CurrentRow.Cells["PlanID"].Value);

var confirmResult = MessageBox.Show("Ви впевнені, що хочете видалити цей план?",
    "Підтвердження видалення",
    MessageBoxButtons.YesNo);

if (confirmResult == DialogResult.Yes)
{
    using (var db = new AppDbContext())
    {
        var plan = db.TrainingPlans.FirstOrDefault(p => p.PlanID == selectedPlanId);
        if (plan != null)
        {
            db.TrainingPlans.Remove(plan);
            db.SaveChanges();
            MessageBox.Show("План успішно видалено.");
            Refresh(); // метод для оновлення DataGridView
        }
        else
        {
            MessageBox.Show("План не знайдено.");
        }
    }
}
}
}

```

Код TrainingPlannerForm

```

namespace TrainMind.Views
{
    public partial class TrainingJournalForm : Form
    {
        public TrainingJournalForm()
        {
            InitializeComponent();
            var context = new AppDbContext();
            comboBox1.DataSource = context.Users.ToList();
            comboBox1.DisplayMember = "Name"; // Assuming User class has a Name property
            // Load training sessions into the DataGridView
            LoadSessions();

        }
        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
                    .Select(p => new
                    {
                        p.PlanID,
                        DisplayText = $"{p.PlanID}"
                    })
                    .ToList();

                comboBox3.DataSource = plans;
                comboBox3.DisplayMember = "DisplayText"; // що бачить користувач
                comboBox3.ValueMember = "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();
}
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    LoadSessions(); // Reload sessions when the selected user changes
    /*
    var context = new AppDbContext();

    // Load training sessions into the DataGridView
    var trainingSessions = context.TrainingSessions
        .Where(x => x.UserID == comboBox1.SelectedIndex + 1)
        .ToList();
    dataGridView1.DataSource = trainingSessions.Select(x => new

```

```

        {
            x.Date,
            x.ActivityType,
            x.ExerciseName,
            x.Duration,
            x.Volume,
            x.FatigueLevel,
            x.Comment,
            x.IsMissed
        }).ToList();*/
    }
    private void TrainingJournalForm_Load(object sender, EventArgs e)
    {
        // This method can be used to initialize any additional components or settings when the form
loads
    }
    private void button1_Click(object sender, EventArgs e)
    {
        if (comboBox1.SelectedItem == null)
        {
            MessageBox.Show("Будь ласка, виберіть користувача.");
            return;
        }
        int userId = ((User)comboBox1.SelectedItem).UserID;
        var newSession = new TrainingSession
        {
            UserID = userId,
            PlanID = (int)comboBox3.SelectedValue,
            Date = DateTime.Now, // або додати DateTimePicker
            ExerciseName = textBox1.Text,
            ActivityType = comboBox2.Text,
            Duration = (int)numericUpDown1.Value,
            Volume = (float)numericUpDown2.Value,
            FatigueLevel = (int)numericUpDown3.Value,
            Comment = textBox5.Text,
            IsMissed = false
        };
        using (var db = new AppDbContext())
        {
            db.TrainingSessions.Add(newSession);
            db.SaveChanges();
        }
        MessageBox.Show("Тренування додано.");
        LoadSessions(); // оновлення DataGridView
    }

```

```

private void button2_Click(object sender, EventArgs e)
{
    if (dataGridView1.CurrentRow == null)
    {
        MessageBox.Show("Будь ласка, виберіть запис для видалення.");
        return;
    }

    // MessageBox.Show(dataGridView1.CurrentRow.Cells["SessionID"].Value);

    int sessionId = Convert.ToInt32(dataGridView1.CurrentRow.Cells["SessionID"].Value);

    var confirm = MessageBox.Show("Ви впевнені, що хочете видалити це тренування?",
    "Підтвердження", MessageBoxButtons.YesNo);
    if (confirm == DialogResult.Yes)
    {
        using (var db = new AppDbContext())
        {
            var session = db.TrainingSessions.FirstOrDefault(s => s.SessionID == sessionId);
            if (session != null)
            {
                db.TrainingSessions.Remove(session);
                db.SaveChanges();
                MessageBox.Show("Запис видалено.");
                LoadSessions(); // оновлення DataGridView
            }
        }
    }
}

private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
{
    label9.Text = comboBox2.SelectedIndex.ToString();
}

private void button3_Click(object sender, EventArgs e)
{
    this.Close(); // Закриваємо форму журналу тренувань
}
}

```

Код AnalyticsController

```
namespace TrainMind.Controllers
{
    internal class AnalyticsController
    {
        private readonly AppDbContext _context;

        public AnalyticsController(AppDbContext context)
        {
            _context = context;
        }

        public List<TrainingSession> GetTrainingSessions(int userId)
        {
            return _context.TrainingSessions.Where(ts => ts.UserID == userId).ToList();
        }

        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
                    })
                    .ToList();

                return new AnalyticsData
                {
                    Dates = grouped.Select(g => g.Date).ToList(),
                    Loads = grouped.Select(g => (float)g.Load).ToList()
                };
            }
        }
    }
}
```

```

    }

    public static AnalyticsData GetLoadByDate2(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.Volume * s.Duration) // приклад формули навантаження
                })
                .ToList();

            return new AnalyticsData
            {
                Dates = grouped.Select(g => g.Date).ToList(),
                Loads = grouped.Select(g => (float)g.Load).ToList()
            };
        }
    }
}

```

```

    public static AnalyticsData GetLoadByDate3(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.FatigueLevel) // приклад формули навантаження
                })

```

```

        .ToList();

    return new AnalyticsData
    {
        Dates = grouped.Select(g => g.Date).ToList(),
        Loads = grouped.Select(g => (float)g.Load).ToList()
    };
}

public Dictionary<string, float> GetTrainingVolumeByDay(int userId)
{
    var sessions = GetTrainingSessions(userId);
    return sessions.GroupBy(s => s.Date.Date)
        .ToDictionary(
            g => g.Key.ToString("yyyy-MM-dd"),
            g => g.Sum(s => s.Volume));
}

public Dictionary<string, float> GetTrainingVolumeByWeek(int userId)
{
    var sessions = GetTrainingSessions(userId);
    return sessions.GroupBy(s => CultureInfo.CurrentCulture.Calendar.GetWeekOfYear(s.Date,
        CalendarWeekRule.FirstDay, DayOfWeek.Monday))
        .ToDictionary(g => g.Key.ToString(), g => g.Sum(s => s.Volume));
}

public Dictionary<string, float> GetTrainingVolumeByMonth(int userId)
{
    var sessions = GetTrainingSessions(userId);
    return sessions.GroupBy(s => s.Date.Month)
        .ToDictionary(g => g.Key.ToString(), g => g.Sum(s => s.Volume));
}
}

```

Код ExportForm

```
namespace TrainMind.Views
{
    public partial class ExportForm : Form
    {
        public ExportForm()
        {
            InitializeComponent();
            comboBox1.SelectedIndex = 0; // Default to the first option
            var _context = new AppDbContext();
            comboBox2.DataSource = _context.Users.ToList();
            comboBox2.DisplayMember = "Name";
            using (var context = new AppDbContext())
            {
                // Load users into the combo box
                var tp = context.ExportLogs.ToList();
                dataGridView1.DataSource = tp;
            }
        }
    }

    private void ExportForm_Load(object sender, EventArgs e)
    {
        // Initialize the combo box with export options

        comboBox1.Enabled = true;
    }

    private void button2_Click(object sender, EventArgs e)
    {

        ExportController.BackupData(comboBox1.SelectedIndex + 1);
    }
}
```


Код DataExporter

```

internal class DataExporter
{
    public void ExportToPdf(List<TrainingSession> sessions)
    {
        // Placeholder logic for exporting to PDF
        using (var writer = new StreamWriter("training_sessions.pdf"))
        {
            writer.WriteLine("Training Sessions Report");
            foreach (var session in sessions)
            {
                writer.WriteLine($"{session.Date}: {session.ActivityType} - {session.Volume}");
            }
        }
    }
    public void ExportToExcel(List<TrainingSession> sessions)
    {
        // Placeholder logic for exporting to Excel
        using (var writer = new StreamWriter("training_sessions.xlsx"))
        {
            writer.WriteLine("Date          ActivityType    Volume");
            foreach (var session in sessions)
            {
                writer.WriteLine($"{session.Date}    {session.ActivityType} {session.Volume}");
            }
        }
    }
    public void CreateBackup(List<TrainingSession> sessions)
    {
        // Placeholder logic for creating backup
        using (var writer = new StreamWriter("training_sessions_backup.txt"))
        {
            writer.WriteLine("Training Sessions Backup");
            foreach (var session in sessions)
            {
                writer.WriteLine($"{session.Date}: {session.ActivityType} - {session.Volume}");
            }
        }
    }
}

```

Код

```

namespace TrainMind.Views
{
    public partial class ExportForm : Form
    {
        public ExportForm()
        {
            InitializeComponent();
            comboBox1.SelectedIndex = 0; // Default to the first option
            var _context = new AppDbContext();
            comboBox2.DataSource = _context.Users.ToList();
            comboBox2.DisplayMember = "Name";
            using (var context = new AppDbContext())
            {
                // Load users into the combo box
                var tp = context.ExportLogs.ToList();
                dataGridView1.DataSource = tp;
            }
        }
    }

    private void ExportForm_Load(object sender, EventArgs e)
    {
        // Initialize the combo box with export options

        comboBox1.Enabled = true;
    }

    private void button2_Click(object sender, EventArgs e)
    {
        ExportController.BackupData(comboBox1.SelectedIndex + 1);
    }

    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);
var db = new AppDbContext();
var sessions2 = db.TrainingSessions
    .Where(x => x.UserID == comboBox2.SelectedIndex + 1 && x.Date >= start && x.Date
<= end)
    .ToList();
if (data.Dates.Count == 0)
    MessageBox.Show($"Знайдено {data.Dates.Count} записів з
{start.ToShortDateString()} по {end.ToShortDateString()}.");
else
{
    // Open the file dialog to select the export path
    using (SaveFileDialog saveFileDialog = new SaveFileDialog())
    {
        string filePath2 = "";
        saveFileDialog.Filter = "PDF files (*.pdf)|*.pdf|Excel files (*.xlsx)|*.xlsx|JSON files
(*.json)|*.json";
        saveFileDialog.Title = "Save Export File";
        if (saveFileDialog.ShowDialog() == DialogResult.OK)
        {
            filePath2 = saveFileDialog.FileName;
        }
        switch (comboBox1.SelectedIndex)
        {
            case 0:
                PdfDocument document = new PdfDocument();
                document.Info.Title = "Training Sessions Report";
                PdfPage page = document.AddPage();
                XGraphics gfx = XGraphics.FromPdfPage(page);
                GlobalFontSettings.UseWindowsFontsUnderWindows = true;
                XFont font = new XFont("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($"Date: {session.Date:yyyy-MM-dd}", font, XBrushes.Black,
new XRect(40, y, page.Width, 20));
                    y += 15;
                }
            }
        }
    }

```

