МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____»_____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ЗАСТОСУНОК ПЕРСОНАЛІЗАЦІЇ ФІЗИЧНОЇ АКТИВНОСТІ НА ОСНОВІ КЛАСТЕРИЗАЦІЇ ТА ПРОГНОЗНИХ МОДЕЛЕЙ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач _____ Єгор УЛІНЕЦЬ

«___»_____ 2026 р.

Керівник канд. техн. наук, ст. викладач _____ Віктор ГОЖИЙ

«___»_____ 2026 р.

Миколаїв – 2026

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Улінця Єгора Олеговича
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Застосунок персоналізації фізичної активності на основі кластеризації та прогнозних моделей».

Керівник роботи: Гожий Віктор Олександрович, старший викладач кафедри інтелектуальних інформаційних систем, канд. техн. наук.

Затверджена наказом ЧНУ ім. Петра Могили від «25» грудня 2025 р. № 353.

2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: розробка десктопного програмного застосунку, призначеного для персоналізованого аналізу та планування фізичної активності користувача на основі методів машинного навчання. Початкові дані містять набори історичних даних про фізичну активність (кількість кроків, спалені калорії, хвилини активності, тривалість сну та кількість випитої води), персональні метрики користувача (вік, зріст, вага), а також результати аналізу існуючих програмних рішень у сфері цифрового здоров’я.

4. Перелік питань, що підлягають розробці: дослідження предметної області та аналіз наявних інформаційних систем відстеження фізичної активності; розробка архітектури десктопного застосунку та проєктування структури локальної бази даних для зберігання показників активності; вибір та математичне обґрунтування алгоритмів машинного навчання (кластеризації та регресії) для аналізу даних; створення користувацьких сценаріїв та проєктування графічного інтерфейсу системи; побудова UML-діаграм (прецедентів та класів) для моделювання роботи програмного засобу; програмна реалізація основного функціоналу застосунку: імпорт наборів даних, візуалізація статистики, автоматичне групування патернів активності та генерація персоналізованих рекомендацій; проведення тестування алгоритмів та оцінка ефективності роботи розробленої системи.

5. Перелік графічних матеріалів: презентація.

Керівник роботи _____ Віктор ГОЖИЙ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Єгор УЛІНЕЦЬ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «23» грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

кваліфікаційної роботи

Тема: «Застосунок персоналізації фізичної активності на основі кластеризації та прогнозних моделей».

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання кваліфікаційної роботи	21.12.2025	24.12.2025	Виконано
2	Дослідження предметної області та постановка задачі проєктування системи	25.12.2025	30.01.2026	Виконано
3	Огляд науково-технічної літератури та аналіз існуючих аналогів програмних засобів для персоналізації фізичної активності	31.01.2026	01.03.2026	Виконано
4	Обґрунтування вибору технологічного стеку та алгоритмів машинного навчання (кластеризації та прогнозування)	02.03.2026	01.04.2026	Виконано
5	Програмна реалізація десктопного застосунку: розробка бази даних, інтеграція ML-моделей та створення графічного інтерфейсу	02.04.2026	24.05.2026	Виконано
6	Перший попередній захист КР на засіданні комісії кафедри	25.05.2026	25.05.2026	Виконано
7	Корегування роботи та оптимізація програмного коду за результатами попереднього захисту	26.05.2026	04.06.2026	Виконано
8	Другий попередній захист КР на засіданні комісії кафедри	05.06.2026	05.06.2026	Виконано
9	Доробка текстової частини, тестування застосунку та остаточне оформлення пояснювальної записки	06.06.2026	14.06.2026	Виконано
10	Подання КР, її електронної копії та супровідних документів (відгуку, рецензії) до захисту	15.06.2026	19.06.2026	Виконано

Керівник роботи _____ Віктор ГОЖИЙ

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Єгор УЛІНЕЦЬ

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«29» січня 2026 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувача 401 групи ЧНУ ім. Петра Могили

Улінця Єгора Олеговича

На тему: «**ЗАСТОСУНОК ПЕРСОНАЛІЗАЦІЇ ФІЗИЧНОЇ АКТИВНОСТІ НА ОСНОВІ КЛАСТЕРИЗАЦІЇ ТА ПРОГНОЗНИХ МОДЕЛЕЙ»**

Актуальність роботи зумовлена необхідністю переходу від систем простого відстеження показників до інтелектуальних рішень. Більшість наявних фітнес-застосунків не здійснюють глибокого аналізу поведінки користувача, що знижує ефективність підтримки здорового способу життя.

Об’єктом роботи є процес аналізу та планування фізичної активності користувача з використанням інформаційних технологій.

Предметом роботи є програмні інструменти та алгоритми машинного навчання (кластеризації та прогнозування) для побудови системи персоналізації фізичних навантажень.

Метою роботи є розробка десктопного застосунку для покращення управління власною фізичною активністю за рахунок виявлення індивідуальних патернів поведінки та генерації персоналізованих рекомендацій.

Дана робота складається з чотирьох розділів. У першому розділі проведено аналіз предметної області, огляд існуючих систем трекінгу здоров’я та аналогів, сформовано постановку задачі на розробку інтелектуального застосунку. Другий розділ присвячений математичним моделям і методам машинного навчання (K-Means, Random Forest), що використані у роботі для кластеризації та предиктивної аналітики. У третьому розділі наведено результати проєктування архітектури бази даних, побудови UML-діаграм та моделювання графічного інтерфейсу користувача. В

четвертому – наведено практичну реалізацію програмного продукту, аналіз отриманих результатів прогнозування з динамічним розрахунком математичних похибок моделі, а також тестування стійкості застосунку до некоректного вводу. Загальний обсяг роботи – 84 сторінок. Кваліфікаційна робота містить 1 додаток, 34 рисунки, 7 таблиць і 37 джерел посилання.

Ключові слова: персоналізація фізичної активності, машинне навчання, кластеризація, K-Means, предиктивна аналітика, Random Forest, Python.

ABSTRACT

to the qualification work by the student of the group 401 Petro Mohyla Black Sea National University

Ulinets Yehor

«APPLICATION FOR PERSONALIZATION OF PHYSICAL ACTIVITY BASED ON CLUSTERING AND PREDICTIVE MODELS»

The relevance of the work is determined by the need to transition from simple tracking systems to intelligent solutions. Most existing fitness applications do not perform an in-depth analysis of user behavior, which reduces the effectiveness of maintaining a healthy lifestyle.

The objective of the work is the process of analyzing and planning user physical activity using information technologies.

The subject of the work includes software tools and machine learning algorithms (clustering and forecasting) for building a physical activity personalization system.

The objective of the work is to develop a desktop application to improve the management of personal physical activity by identifying individual behavioral patterns and generating personalized recommendations.

This work consists of four chapters. In the first chapter, an analysis of the subject area is carried out, a review of existing health tracking systems and analogues is provided, and the problem statement for the development of an intelligent application is formulated. The second chapter is devoted to mathematical models and machine learning methods (K-Means, Random Forest) used in the work for clustering and predictive analytics, as well as the theoretical justification of metrics for evaluating their accuracy. The third chapter presents the results of designing the database architecture using ORM technology, building UML diagrams, and modeling the graphical user interface. In the fourth chapter, the practical implementation of the software product, analysis of the obtained forecasting results with dynamic calculation of mathematical model errors, and application

robustness testing are presented. The total volume of the work is 84 pages. The qualification work contains 1 appendix, 34 figures, 7 tables, and 37 references.

Keywords: physical activity personalization, machine learning, clustering, K-Means, predictive analytics, Random Forest, Python.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – База даних

СКБД – Система керування базами даних

BMR – Базовий рівень метаболізму

GUI – Графічний інтерфейс користувача

ML – Машинне навчання

ORM – Об’єктно-реляційне відображення

SQL – Мова структурованих запитів

UML –Уніфікована мова моделювання

ВСТУП

Сьогодні спостерігається стрімке зростання уваги суспільства до збереження та покращення показників власного здоров’я. Такий тренд є рушієм розвитку різноманітних цифрових засобів, які покликані допомагати людям у досягненні персональних цілей щодо підтримки належної фізичної форми. Однак більшість наявних на ринку програмних рішень для фітнесу мають значну прогалину: вони переважно виконують функцію звичайного збору та фіксації показників, залишаючи поза увагою глибокий аналіз індивідуальних поведінкових патернів користувача.

Актуальність даної роботи обумовлена гострою потребою у розробці більш гнучких та інтелектуальних систем. Існує необхідність створення таких застосунків, які здатні не лише відображати поточний стан, а й адаптивно генерувати персоналізовані рекомендації на базі виявлених закономірностей у фізичних навантаженнях, ритмах сну та відпочинку. Впровадження алгоритмів машинного навчання для кластеризації даних дозволяє значно підвищити якість взаємодії користувача з програмою, сприяючи підтримці стабільної мотивації до регулярних тренувань.

Метою цієї роботи є підвищення загальної результативності управління фізичною активністю людини через створення спеціалізованої інформаційної системи. Розроблений застосунок покликаний забезпечити зручне середовище для накопичення даних, їх інтелектуального аналізу та формування прогнозних моделей, які слугуватимуть основою для планування майбутніх навантажень.

Декларація про використання ШІ. Відповідно до таксономії GAIDeT (2025), наведені нижче завдання були делеговані інструментам генеративного ШІ за повного людського нагляду:

- оцінювання здійсненності та ризиків;
- пошук і систематизація літератури;
- оптимізація коду;

- вичитування та редагування;
- резюмування тексту;
- оцінювання якості;
- рекомендації.

Використаний інструмент генеративного ШІ: Gemini 3.1 Pro.

Повну відповідальність за фінальний рукопис несе автор.

Інструменти генеративного ШІ не зазначаються як автори та не несуть відповідальності за кінцеві результати.

Декларацію подав: Улінець Єгор.

1 АНАЛІЗ ІНФОРМАЦІЙНИХ СИСТЕМ МОНІТОРИНГУ ЗДОРОВ'Я ТА ПОСТАНОВКА ЗАДАЧІ ІНТЕЛЕКТУАЛЬНОЇ ОБРОБКИ ДАНИХ

1.1 Опис предметної сфери

В умовах стрімкого розвитку інформаційних технологій та глобальної цифровізації суспільства спостерігається значне зростання інтересу населення до концепції здорового способу життя. Сучасні темпи життя, поширення малорухливих професій, незбалансоване харчування та високий рівень стресу ставлять перед людством нові виклики щодо збереження фізичного та ментального здоров'я. У відповідь на ці виклики виник і активно розвивається напрям цифрового здоров'я, який інтегрує медичні знання, інформаційні технології та портативні пристрої.

Ключовим елементом цифрового здоров'я є технології моніторингу та відстеження фізичної активності. Поява на масовому ринку доступних носимих пристроїв, таких як розумні годинники та фітнес-браслети, докорінно змінила підхід до персонального здоров'я. Сучасні пристрої оснащені складними сенсорними системами: акселерометрами, гіроскопами, оптичними датчиками серцевого ритму та датчиками температури шкіри. Ці сенсори дозволяють з високою точністю, у режимі реального часу і безперервно збирати широкий спектр біометричних даних та показників рухової активності.

Основними типами даних, що генеруються в процесі моніторингу, є:

– кінематичні дані: кількість пройдених кроків, подолана відстань, швидкість руху;

– фізіологічні показники: частота серцевих скорочень, рівень кисню в крові, варіабельність серцевого ритму;

– енергетичні показники: оцінка об'єму спалених калорій як у стані спокою, так і під час активності;

– показники сну: тривалість сну, розподіл на фази (швидкий, глибокий, легкий сон), кількість пробуджень.

Однак безперервний збір цих даних створює нову проблему у предметній сфері – перевантаження «сирими даними». Користувач щоденно генерує величезні масиви інформації. Більшість існуючих мобільних застосунків, які постачаються разом із трекерами, виконують лише функцію візуалізації – вони відображають абсолютні значення (наприклад, «пройдено 8000 кроків» або «сон тривав 6 годин») за допомогою базових графіків чи діаграм. Такий підхід належить до категорії описової аналітики, яка дає відповідь лише на питання, що саме відбулося.

Проблема полягає в тому, що пересічному користувачеві без спеціальної освіти важко інтерпретувати ці ізольовані цифри. Відсутність комплексного аналізу взаємозв'язків між різними показниками (наприклад, як нестача глибокого сну напередодні впливає на ефективність тренування сьогодні) призводить до того, що зібрані дані не трансформуються у корисні знання. Як наслідок, користувач часто втрачає мотивацію до використання трекера через кілька місяців після покупки, оскільки не отримує персоналізованих рекомендацій щодо того, як саме йому слід змінити свою поведінку.

Вирішенням цієї проблеми є перехід до предиктивного аналізу, що стає можливим завдяки впровадженню методів штучного інтелекту та машинного навчання в інформаційні системи. Машинне навчання дозволяє аналізувати багатовимірні набори даних користувача, знаходити приховані закономірності та групувати дні з подібною поведінкою за допомогою кластеризації. На основі виявлених кластерів система може прогнозувати майбутню активність і генерувати адаптивні рекомендації, які враховують індивідуальні особливості конкретного користувача, а не загальні стандарти.

Виявлення індивідуальних патернів поведінки є ключовим етапом у створенні дійсно персоналізованих систем. Кожна людина має власний унікальний ритм життя, який складно описати універсальними усередненими нормами. Наприклад, для одного користувача проходження п'яти тисяч кроків є ознакою малорухливого дня, тоді як для іншого це може бути нормальним рівнем активності в період відновлення після важкого силового тренування. Саме тому застосування

методів кластеризації до зібраних даних дозволяє системі автоматично виокремлювати специфічні групи днів та формувати індивідуальні профілі активності.

Завдяки кластеризації історичні масиви інформації розподіляються на логічні категорії: наприклад, активні будні, дні відпочинку, періоди підвищеного навантаження або дні з дефіцитом сну. Визначення того, до якого саме кластера належить поточний стан користувача, створює надійний фундамент для роботи прогнозних алгоритмів. Якщо програма здатна передбачити ймовірне зниження фізичного ресурсу людини на основі аналізу її попередніх днів, вона може завчасно скоригувати розклад майбутніх навантажень. Такий превентивний підхід значно знижує ризики перетренованості, зменшує ймовірність отримання травм та робить процес досягнення фітнес-цілей більш стабільним.

Окрім математичної та алгоритмічної складової, критично важливим аспектом є архітектура самої інформаційної системи. На сучасному ринку цифрового здоров’я абсолютно переважають мобільні додатки та хмарні вебзастосунки. Незважаючи на їхню мобільність та інтеграцію із соціальними мережами, вони мають суттєвий недолік, пов’язаний із безпекою та приватністю. Дані про серцевий ритм, якість сну та щоденні маршрути є надзвичайно чутливою особистою інформацією. Використання хмарних аналітичних сервісів вимагає обов’язкової передачі цієї інформації на віддалені сервери сторонніх компаній, що автоматично створює ризики витоку даних або їх використання у комерційних цілях без відома користувача.

З огляду на це, розробка десктопного застосунку для локального інтелектуального аналізу фізичної активності є доцільним, безпечним та обґрунтованим рішенням. Формат настільної програми дозволяє зберігати всі історичні бази даних безпосередньо на персональному комп’ютері користувача, гарантуючи максимальний рівень конфіденційності. Крім того, сучасні комп’ютери володіють достатніми обчислювальними потужностями для швидкого виконання операцій з очищення даних та тренування моделей машинного навчання без

необхідності постійного підключення до мережі Інтернет. Це робить десктопні рішення оптимальним середовищем для розгортання персональних аналітичних систем підтримки прийняття рішень.

Таким чином, предметна сфера моніторингу здоров'я вимагає переходу від простих портативних лічильників до потужних локальних систем інтелектуальної обробки інформації. Розуміння цих тенденцій, а також усвідомлення обмежень існуючих технологій створює основу для подальшого детального дослідження наявних програмних продуктів. Це, у свою чергу, дозволить чітко сформулювати функціональні вимоги до нової розроблюваної системи.

1.2 Огляд та аналіз наявних аналогів

Ринок програмного забезпечення у сфері цифрового здоров'я пропонує широкий вибір мобільних та вебзастосунків для моніторингу фізичної активності. Для того щоб сформувати вимоги до розроблюваної інформаційної системи, необхідно детально проаналізувати існуючі аналоги, виділити їхні ключові переваги та ідентифікувати технологічні недоліки. Для порівняльного аналізу обрано три найбільш популярні платформи: Strava [1], Google Fit [2] та Apple Health [3].

1. Strava [1]

Strava – це один із найпопулярніших у світі мобільних та вебзастосунків для відстеження фізичної активності, орієнтований переважно на бігунів та велосипедистів. Головною особливістю платформи є її соціальна складова: користувачі можуть змагатися на певних ділянках маршруту, ділитися результатами та порівнювати свої показники з іншими спортсменами.

Інтерфейс програми Strava (рис. 1.1) побудований за принципом соціальної стрічки новин, де основний акцент робиться на публікації виконаних тренувань, картах маршрутів та коментарях підписників. Навігація орієнтована на швидкий доступ до запису нового тренування та перегляду таблиць лідерів. Програма відстежує рух за допомогою GPS-модуля пристрою, фіксує швидкість, темп, набір

висоти та розраховує витрачені калорії. У платній версії (Strava Premium) доступний більш розширений аналіз навантажень на основі даних про частоту серцевих скорочень.

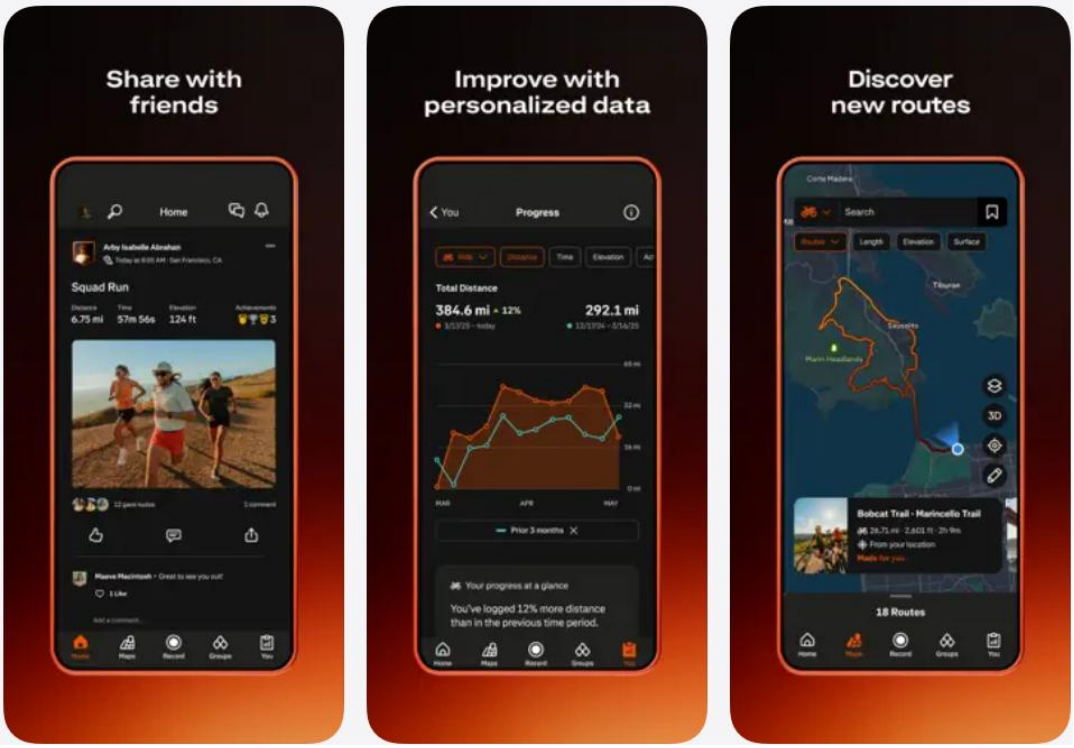

Рисунок 1.1 – Інтерфейс програми Strava [1]

До переваг системи можна віднести потужну соціальну мотивацію та зручний алгоритм побудови маршрутів. Проте суттєвим недоліком є те, що система орієнтована на фіксацію вже виконаних спортивних тренувань, а не на повсякденний моніторинг (сон і пасивна активність не враховуються). Strava не використовує алгоритми кластеризації для пошуку індивідуальних патернів поведінки та не надає предиктивних рекомендацій щодо майбутніх тренувань.

2. Google Fit [2]

Google Fit – це універсальний трекер активності, розроблений компанією Google переважно для екосистеми Android. Платформа позиціонується як центральний агрегатор даних про здоров’я, здатний збирати інформацію з датчиків

смартфона та синхронізуватися з іншими фітнес-додатками й розумними годинниками.

Застосунок Google Fit (рис. 1.2) використовує концепцію мінімалістичного дизайну. Головний екран складається з концентричних кілець прогресу («Кроки» та «Кардіобали»). Такий інтерфейс забезпечує швидке зчитування базової інформації, проте не дозволяє розгорнути складну багаторівневу аналітику на одному екрані. В основі концепції лежить гейміфікація: додаток автоматично розпізнає тип рухової активності і нараховує бали за досягнення встановлених цілей.

Рисунок 1.2 – Інтерфейс програми Google Fit [2]

Незважаючи на безкоштовний доступ та чудову інтеграцію зі сторонніми сервісами, аналітика Google Fit є базовою. Система обмежується підрахунком балів та констатацією факту виконання щоденної норми. Вона не будує прогнозних моделей і не здатна адаптувати рекомендації залежно від накопиченої втоми чи якості сну попередньої ночі. Крім того, всі зібрані дані зберігаються на хмарних серверах корпорації.

3. Apple Health [3]

Додаток Apple Health є вбудованим системним рішенням для користувачів операційної системи iOS. Це потужна інформаційна екосистема, яка збирає вичерпні дані з iPhone, Apple Watch та сторонніх медичних пристроїв. Платформа здатна агрегувати найширший спектр метрик: від стандартних показників активності до клінічних медичних записів.

Інтерфейс Apple Health (рис. 1.3) реалізований у вигляді дашборду з системою карток. Кожна картка відповідає за окрему метрику (сон, кроки, серцевий ритм). Незважаючи на величезну кількість зібраних даних, інтерфейс залишається «пласким» – він відображає лише історичні зведення та графіки, не пропонуючи користувачу екранів для роботи з прогнозами чи персоналізованим плануванням. Компанія робить великий акцент на конфіденційності: дані шифруються на пристрої і не передаються на сервери без явного дозволу.

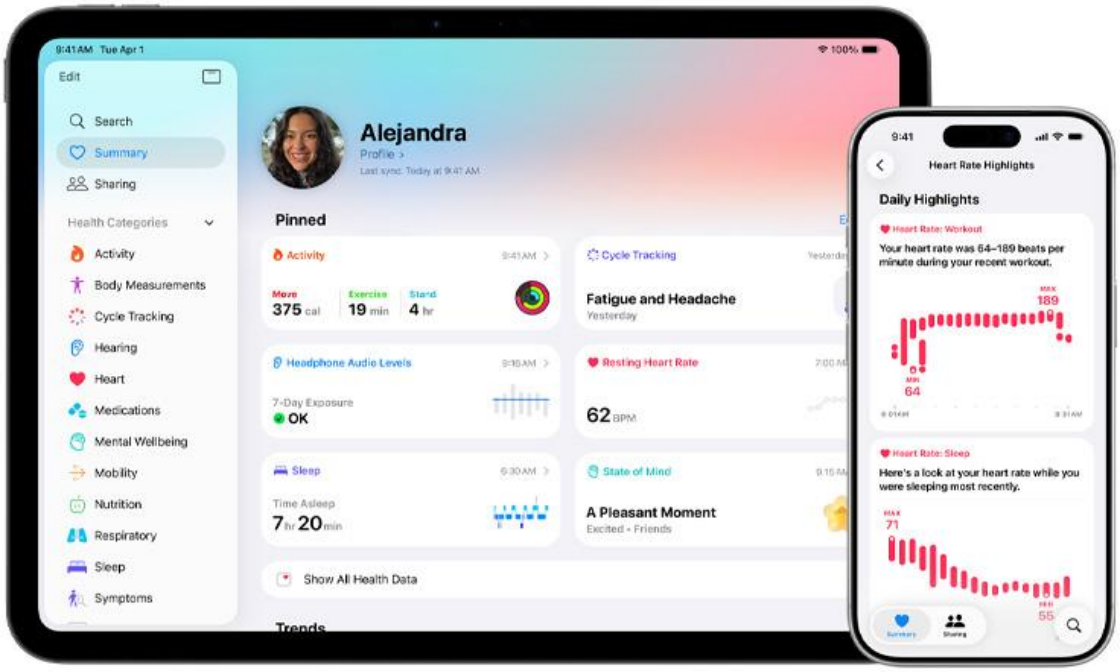

Рисунок 1.3 – Інтерфейс програми Apple Health [3]

Перевагами системи є найвищий рівень безпеки даних та безшовна інтеграція з апаратним забезпеченням Apple. Недоліком є те, що платформа є повністю закритою (ексклюзивною для Apple). Як і попередні аналоги, Apple Health чудово

візуалізує поточні тренди, проте не містить вбудованих інструментів прескриптивної аналітики. Система не генерує персоналізований розклад тренувань за допомогою машинного навчання.

Аналіз інтерфейсів підтверджує, що мобільний формат обмежує можливості для відображення складної аналітики, результатів кластеризації та прогнозних моделей, що ще раз підкреслює доцільність розробки саме десктопного застосунку.

Для систематизації отриманих результатів та обґрунтування функціоналу розроблюваної системи результати огляду зведено у таблицю порівняльного аналізу (табл. 1.1).

Таблиця 1.1 – Порівняльний аналіз програмних засобів для моніторингу активності

Критерій порівняння	**Strava**	**Google Fit**	**Apple Health**	**Розроблюваний застосунок**
Цільова платформа	Мобільні пристрої, Веб	Мобільні пристрої	iOS (ексклюзивно)	Десктоп (ПК)
Основний фокус	Спортивні тренування (біг, вело)	Базова щоденна активність	Комплексний медичний хаб	Персоналізація та аналітика патернів
Тип аналітики	Описова (статистика тренувань)	Описова (кардіобали)	Описова (історичні графіки)	Предиктивна (прогнозування активності)
Використання ML (кластеризації)	Відсутнє	Відсутнє	Відсутнє	Реалізовано (визначення типів днів)
Локальне зберігання даних	Ні (хмарне)	Ні (хмарне)	Так (на пристрої)	Так (локальна база на ПК)
Генерація адаптивних планів	Ні (тільки статичні плани)	Ні	Ні	Так (на основі кластерів)

Як видно з таблиці 1.1, жоден з проаналізованих популярних застосунків не використовує методи кластеризації для виявлення індивідуальних патернів і не надає прогнозних рекомендацій щодо планування фізичної активності, обмежуючись лише статистичною візуалізацією.

1.3 Постановка задачі

На основі проведеного аналізу предметної сфери та огляду існуючих рішень для моніторингу здоров’я, можна констатувати, що ринок потребує створення нових, більш інтелектуальних інструментів для обробки персональних даних фізичної активності.

Актуальність роботи зумовлена необхідністю якісного переходу від систем простого відстеження статистичних показників (описової аналітики) до систем інтелектуального управління здоров’ям. Більшість наявних фітнес-застосунків не здійснюють глибокого багатовимірного аналізу поведінки користувача, що знижує їх ефективність як інструментів для підтримки здорового способу життя. Впровадження методів машинного навчання, зокрема кластеризації, дозволяє автоматично виявляти індивідуальні патерни активності та формувати персоналізовані рекомендації, які враховують поточний стан організму, рівень втоми та якість сну.

Метою роботи є розробка десктопного інформаційного застосунку для управління персональною фізичною активністю. Застосунок має забезпечити користувачу можливість не лише зручно зберігати та візуалізувати зібрані біометричні дані, але й застосовувати до них алгоритми машинного навчання з метою кластеризації типів активності та генерації прогнозних планів на майбутні періоди.

Об’єктом роботи є процес аналізу, оцінки та планування фізичної активності користувача з використанням інформаційних технологій.

Предметом роботи є методи машинного навчання (алгоритми кластеризації та прогнозування) і програмні інструменти для побудови інтелектуальної десктопної системи персоналізації фізичних навантажень.

Для досягнення поставленої мети та вирішення проблеми, описаної у попередніх підрозділах, необхідно виконати такі основні завдання:

– обґрунтувати вибір та спроєктувати структуру локальної реляційної бази даних для безпечного зберігання метрик активності користувача (кількість кроків, спалені калорії, показники сну тощо);

– обрати технологічний стек та бібліотеки для реалізації методів обробки даних на мові програмування Python;

– програмно реалізувати модуль кластеризації для автоматичного групування історичних даних користувача у змістовні категорії (наприклад, «день високого навантаження», «день відновлення»);

– розробити модуль генерування рекомендацій, який на основі визначеного кластеру пропонуватиме персоналізований план на наступний день;

– спроєктувати зручний та інтуїтивно зрозумілий графічний інтерфейс десктопного застосунку з використанням інтерактивних діаграм та графіків для візуалізації аналітики.

Виходячи зі сформульованих завдань, розроблюваний програмний продукт повинен відповідати чітким вимогам до функціональності та надійності.

Функціональні вимоги до системи:

– можливість ручного введення щоденних показників через інтерфейс користувача;

– відображення загальної статистики та динаміки змін у вигляді лінійних графіків;

– запуск алгоритму машинного навчання (кластеризації) для сегментації даних та візуалізація результатів розподілу;

– формування текстових або графічних порад та прогнозів щодо майбутньої активності на основі знайдених патернів.

Нефункціональні вимоги до системи:

– застосунок має бути реалізований як десктопна програма для операційної системи Windows;

– система повинна працювати автономно, без необхідності постійного підключення до мережі Інтернет;

– усі зібрані дані користувача повинні зберігатися виключно у локальній базі даних на персональному комп’ютері для забезпечення високого рівня конфіденційності;

– час відгуку системи під час виконання операцій кластеризації набору даних обсягом до 1000 записів не повинен перевищувати 5 секунд.

Виконання зазначених завдань та дотримання сформульованих вимог дозволить створити ефективний інструмент підтримки прийняття рішень для індивідуального контролю здоров’я.

Висновки до розділу 1

У першому розділі було проведено комплексне дослідження предметної області моніторингу та аналізу фізичної активності людини. Встановлено, що перехід суспільства до використання носимих пристроїв (фітнес-трекерів) призвів до генерації великих масивів біометричних даних. Проте доведено, що існує проблема «сирих даних», коли користувачі не отримують користі від зібраних метрик через відсутність інструментів для їх інтелектуального аналізу. Обґрунтовано необхідність використання алгоритмів машинного навчання, зокрема кластеризації, для виявлення прихованих індивідуальних патернів поведінки та переходу від описової до предиктивної аналітики.

В ході огляду та порівняльного аналізу існуючих програмних рішень (Strava, Google Fit, Apple Health) було виявлено їхні ключові недоліки: орієнтованість виключно на констатацію фактів (статистику), відсутність алгоритмів прогнозування, а також ризики щодо конфіденційності даних при використанні хмарних архітектур. На основі цих висновків було чітко сформульовано

постановку задачі. Визначено мету, об’єкт, предмет роботи та складено перелік основних завдань для розробки. Сформовано конкретні функціональні та нефункціональні вимоги до майбутньої системи, виконання яких забезпечить створення безпечного, локального десктопного застосунку з інтегрованими модулями машинного навчання для персоналізації планів фізичної активності.

2 АЛГОРИТМІЧНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПЕРСОНАЛІЗАЦІЇ ФІЗИЧНОЇ АКТИВНОСТІ

2.1 Обґрунтування вибору методів машинного навчання та технологічного стеку

Для успішного функціонування інтелектуальної системи персоналізації фізичної активності було проведено аналіз та вибір оптимальних методів обробки даних і програмних інструментів. Було обрано наступний технологічний стек та алгоритми:

– K-Means: метод кластеризації (навчання без учителя), який використовується для сегментації користувачів на групи залежно від їхньої фізичної активності (кількість кроків, спалені калорії, пульс у спокої тощо). Алгоритм ефективно розподіляє дані на задану кількість кластерів, мінімізуючи дисперсію всередині кожної групи. Обчислювальна складність алгоритму дозволяє ефективно обробляти великі обсяги біометричних даних локально [4];

– Random Forest: ансамблевий алгоритм машинного навчання, який планується використовувати для прогнозування майбутніх показників активності користувача. Він генерує множину дерев рішень і усереднює їхні результати, що робить його дуже стійким до аномалій (викидів) у даних з фітнес-трекерів. Цей метод дозволяє ефективно працювати з нелінійними залежностями та детально описаний у працях з інтелектуального аналізу даних [5];

– Python та Scikit-learn: Python – високорівнева мова програмування, що є стандартом де-факто у сфері машинного навчання та аналізу даних. Вона має велику екосистему готових бібліотек, що дозволяє швидко реалізувати математичні моделі. У свою чергу, Scikit-learn є найпопулярнішою бібліотекою мовою Python для машинного навчання, що містить оптимізовані реалізації алгоритмів K-Means та Random Forest [6, 7];

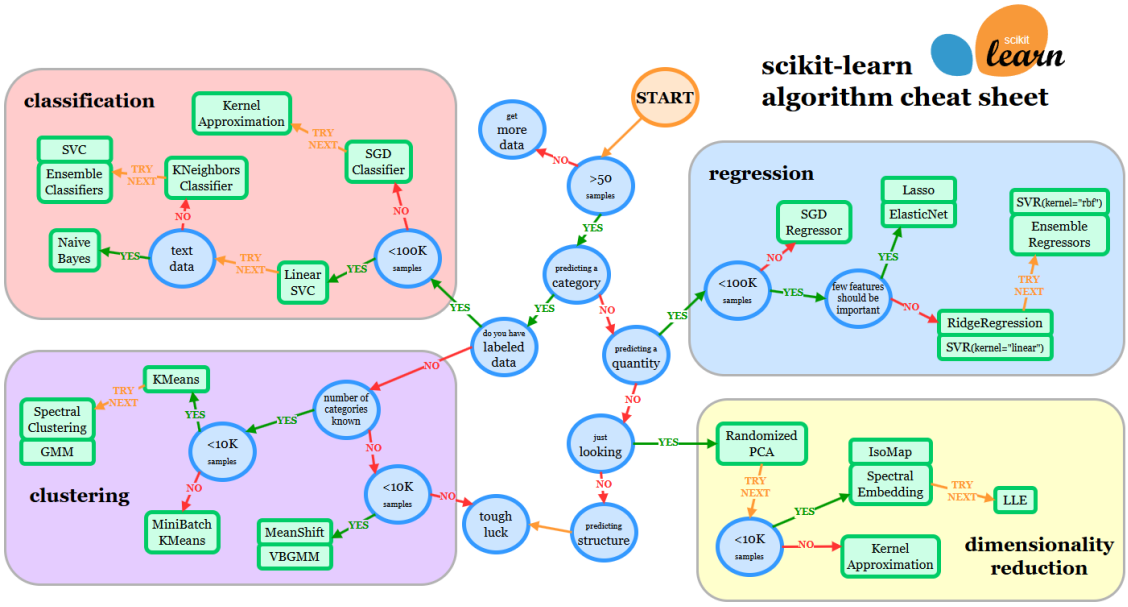

Рисунок 2.1 – Скріншот з офіційної сторінки Scikit-learn [8]

– SQLite та SQLAlchemy: SQLite – компактна реляційна система управління базами даних, яка зберігає всю інформацію безпосередньо у локальному файлі на комп’ютері користувача, не потребуючи розгортання окремого сервера. SQLAlchemy – це потужний інструментарій та об’єктно-реляційне відображення (ORM) для Python, що дозволяє працювати з базою даних через класи Python, уникаючи написання «сирих» SQL-запитів. Разом ці технології забезпечують швидку та безпечну роботу застосунку, як зазначено у їхніх специфікаціях [9, 10];

– CustomTkinter: сучасна бібліотека для створення графічних інтерфейсів (GUI) у Python. На відміну від standard Tkinter, вона дозволяє створювати адаптивні десктопні додатки із сучасним дизайном (заокруглені кути, підтримка темної та світлої тем), що робить програму візуально привабливою та зручною для користувача [11].

2.2 Математична модель алгоритму кластеризації K-Means

Алгоритм K-Means є одним із найбільш ефективних та поширених методів кластеризації в задачах машинного навчання без учителя [12]. Його основна мета полягає у розбитті множини з N спостережень $X = \{x_1, x_2, \dots, x_N\}$, де кожне

спостереження є d-вимірним вектором ознак, на заздалегідь задану кількість K кластерів $C = \{C_1, C_2, \dots, C_K\}$ таким чином, щоб мінімізувати варіацію всередині кожного кластера [13].

У контексті розроблюваної системи кожне спостереження x_i представляє собою набір щоденних біометричних показників конкретного користувача (наприклад, кількість пройдених кроків, спалені калорії, кількість випитої води тощо).

Математично задача алгоритму K-Means зводиться до оптимізації (мінімізації) цільової функції, яка називається внутрішньокластерною сумою квадратів WCSS (Within-Cluster Sum of Squares). Вона обчислюється за такою формулою:

$$WCSS = \sum_{j=1}^{K} \sum_{x_i \epsilon C_j} \left\| x_i - \mu_j \right\|^2, \quad (2.1)$$

де K – загальний кількість кластерів;

C_j – j-й кластер;

x_i – точка даних, що належить до кластера C_j;

μ_j – центроїд (середнє значення) точок у кластері C_j;

$\|x_i - \mu_i\|^2$ – квадрат евклідової відстані між точкою даних та центроїдом кластера [14].

Для знаходження локального мінімуму цієї цільової функції використовується стандартний ітеративний алгоритм Ллойда, який складається з наступних кроків [15].

Ініціалізація. Випадковим чином обираються K початкових центроїдів $\mu_1^{(0)}, \mu_2^{(0)}, \dots, \mu_K^{(0)}$ з простору даних. Для підвищення ефективності та уникнення поганих локальних мінімумів у системі використовується вдосконалений метод ініціалізації k-means++ [16].

Крок призначення. Кожне спостереження x_i призначається до того кластера, чий центроїд знаходиться найближче до нього (за мірою евклідової відстані). Формально, кластер $C_j^{(t)}$ на ітерації t формується як:

$$C_j^{(t)} = \left\{ x_i : \left\| x_i - \mu_l^{(t)} \right\|^2 \ \forall l, 1 \le l \le K \right\}. \tag{2.2}$$

Крок оновлення. Перераховуються координати центроїдів для кожного кластера на основі точок, які були призначені йому на попередньому кроці. Новий центроїд $\mu_j^{(t+1)}$ обчислюється як векторне середнє арифметичне:

$$\mu_j^{(t+1)} = \frac{1}{\left|C_j^{(t)}\right|} \sum_{x_i \epsilon C_j^{(t)}} x_i. \tag{2.3}$$

Умова зупинки. Кроки 2 і 3 повторюються до досягнення збіжності – тобто доки координати центроїдів не перестануть змінюватися, або поки не буде виконана максимально допустима кількість ітерацій [17].

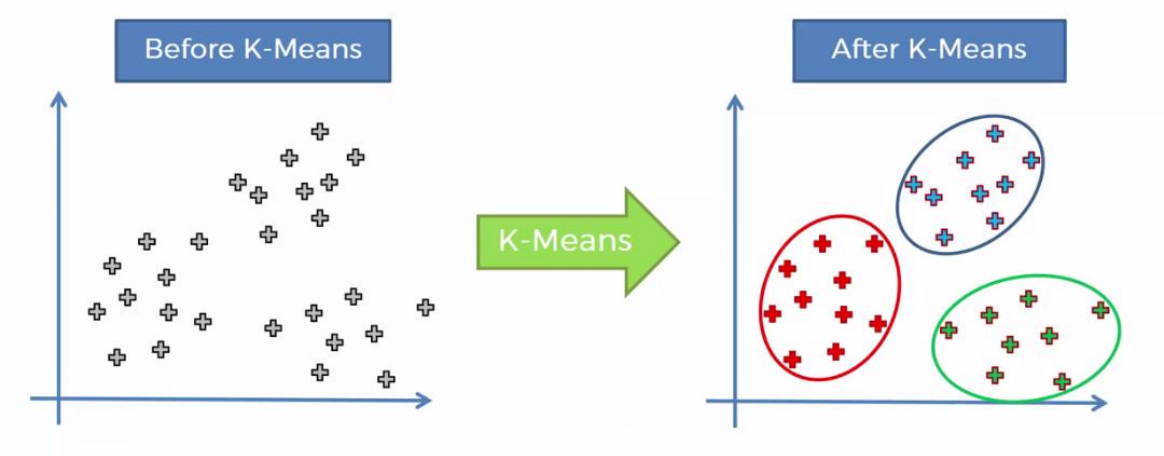

Рисунок 2.2 – Процес кластеризації алгоритмом K-Means [18]

Однією з головних проблем алгоритму K-Means є необхідність заздалегідь вказувати кількість кластерів K. Оскільки розроблювана система повинна бути автоматизованою, для визначення оптимальної кількості груп користувачів імплементовано «метод ліктя» [19]. Суть методу полягає у запуску алгоритму для різних значень K (наприклад, від 1 до 10) та побудові графіка залежності WCSS від кількості кластерів. Оптимальним вважається значення K, при якому додавання нового кластера більше не призводить до значного зменшення похибки, утворюючи на графіку характерний «згин» [20].

2.3 Алгоритми прогнозування фізичної активності

Після сегментації користувачів на відповідні кластери наступним важливим етапом роботи інтелектуальної системи є прогнозування майбутніх показників їхньої фізичної активності. Задача прогнозування (наприклад, оцінка кількості кроків або витрачених калорій на наступний день) з математичної точки зору належить до класу задач регресії у машинному навчанні [21].

Для реалізації цієї функції було обрано алгоритм Випадкового лісу. Цей метод є потужним ансамблевим алгоритмом машинного навчання, який будується на основі великої кількості незалежних дерев рішень [22]. Вибір саме цього алгоритму обґрунтований його високою точністю, стійкістю до перенавчання та здатністю ефективно працювати з нелінійними залежностями і викидами, що є дуже характерними для біометричних даних (наприклад, різка зміна активності користувача у вихідні дні). Математична основа алгоритму базується на концепції беггінгу. Нехай задано навчальну вибірку $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, де x_i – вектор вхідних ознак (історія активності за попередні дні, день тижня, тривалість сну тощо), а y_i – цільова змінна (прогнозована кількість кроків). Алгоритм генерує B нових навчальних підвибірок S_b шляхом випадкової вибірки з поверненням із початкового набору даних [24].

Для кожної підвибірки S_b будується окреме дерево рішень $f_b(x)$. Особливістю Випадкового лісу є те, що під час розгалуження кожного вузла дерева

алгоритм шукає оптимальне розбиття не серед усіх доступних ознак, а лише серед їх випадкової підмножини. Це забезпечує дескореляцію дерев між собою і значно підвищує загальну надійність моделі [25].

Процес розбиття у вузлах дерева для задачі регресії найчастіше здійснюється за критерієм мінімізації дисперсії або середньоквадратичної похибки MSE (Mean Squared Error). Для вузла m з кількістю спостережень N_m похибка обчислюється за формулою:

$$MSE = \frac{1}{N_m}\sum_{i=1}^{N_m}(y_i - \bar{y}_m)^2, \tag{2.4}$$

де y_i – фактичне значення цільової змінної для i-го спостереження у вузлі;

$\bar{y}_m$ – середнє значення цільової змінної у цьому ж вузлі [26].

Після того, як побудовано всі B дерев рішень, фінальний прогноз моделі Випадкового лісу для нового невідомого вхідного вектора ознак x (наприклад, даних користувача на сьогоднішній вечір) обчислюється як усереднене значення прогнозів усіх індивідуальних дерев ансамблю:

$$\hat{y} = \frac{1}{B}\sum_{b=1}^{B} f_b(x), \tag{2.5}$$

де $\hat{y}$ – фінальне прогнозоване значення активності;

B – загальна кількість дерев у лісі;

$f_b(x)$ – прогноз окремого b-го дерева [27].

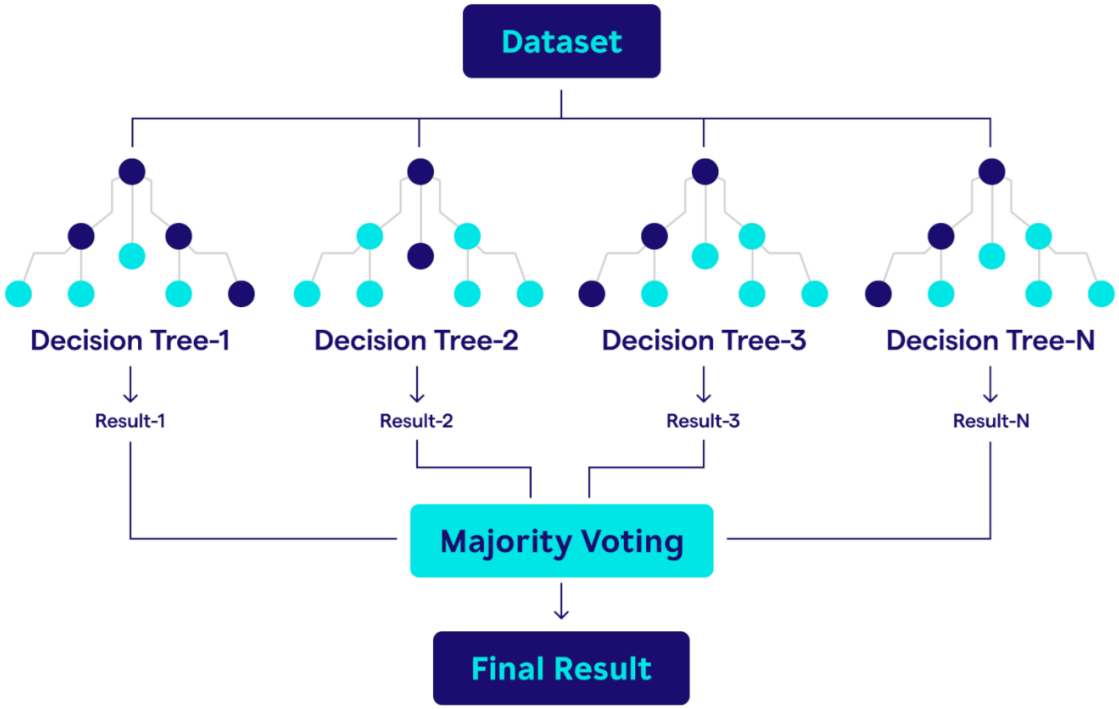

Рисунок 2.3 – Архітектура моделі Випадкового лісу [28]

Головною перевагою такого математичного підходу є те, що дисперсія похибки загального ансамблю зменшується пропорційно кількості дерев, що гарантує високу стабільність прогнозів системи навіть при наявності шумів чи пропущених значень у даних фітнес-трекера [29].

Для фінальної оцінки якості моделі та інтерпретації результатів розраховуються додаткові метрики.

Середня абсолютна похибка (MAE). Відображає середнє відхилення прогнозу у тих самих одиницях виміру, що й сам показник (кроки, хвилини тощо):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|. \quad (2.6)$$

Коефіцієнт детермінації (R^2). Оцінює, яку частку дисперсії залежної змінної пояснює побудована модель (чим ближче до 1, тим краще):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}. \quad (2.7)$$

Використання комплексу вищезазначених метрик дозволяє не лише мінімізувати похибку на етапі навчання алгоритму, але й надати користувачеві кінцевого програмного продукту зрозумілу та прозору оцінку якості згенерованих прогнозів (через показники MAE та R^2).

2.4 Аналіз та підготовка набору даних (FitBit Dataset)

Будь-яка інтелектуальна система, що базується на методах машинного навчання, вимагає якісного та репрезентативного набору даних для навчання, тестування та валідації математичних моделей [30]. Для розробки системи персоналізації фізичної активності було обрано відкритий набір даних «FitBit Fitness Tracker Data», опублікований на платформі Kaggle [31]. Цей набір містить персональні фітнес-дані від 30 користувачів смарт-пристроїв FitBit, які надали згоду на використання своєї щоденної статистики (кількість кроків, серцевий ритм, спалені калорії, фази сну тощо) для наукових та дослідницьких цілей [32].

Загальний масив даних складається з кількох взаємопов’язаних таблиць, проте базовим для задачі кластеризації та прогнозування є файл щоденної активності (dailyActivity_merged.csv). Основні атрибути цього набору даних та їхній опис наведено в Таблиці 2.1 [33].

Таблиця 2.1 – Структура базового набору даних фізичної активності

Назва атрибута (стовпця)	**Тип даних**	**Опис показника**
Id	Кількісний (Int)	Унікальний ідентифікатор користувача фітнес-трекера
ActivityDate	Дата (Date)	Дата фіксації показників у форматі ММ/ДД/РРРР
TotalSteps	Кількісний (Int)	Загальна кількість пройдених кроків за день

Кінець таблиці 2.1

TotalDistance	Дійсний (Float)	Пройдена дистанція за день (у кілометрах)
VeryActiveMinutes	Кількісний (Int)	Кількість хвилин інтенсивної фізичної активності
SedentaryMinutes	Кількісний (Int)	Час, проведений у малорухливому (сидячому) стані
Calories	Кількісний (Int)	Загальна кількість спалених калорій за добу

Перед подачею необроблених («сирих») даних на вхід алгоритмів машинного навчання, вони повинні пройти етап попередньої підготовки. Цей етап є критично важливим, оскільки наявність аномалій, пропущених значень або різних масштабів вимірювання може призвести до некоректної роботи моделей [34].

Процес підготовки даних у розроблюваній системі складається з трьох ключових етапів:

1) очищення даних. Перевірка масиву на наявність дублікатів та пропущених значень (NaN / Null). Рядки з критичними пропусками (наприклад, відсутність даних про калорії або кроки за день) видаляються, щоб не спотворювати результати кластеризації;

2) трансформація типів. Переведення атрибута ActivityDate з текстового (String) у формат дати та часу (Datetime) для можливості аналізу часових рядів та виявлення залежностей від дня тижня [35];

3) масштабування ознак. Оскільки алгоритм K-Means базується на обчисленні евклідової відстані між точками, ознаки з великими числовими значеннями (наприклад, кроки: 10 000) будуть домінувати над ознаками з малими значеннями (наприклад, дистанція: 7.5 км). Щоб уникнути цього, всі ознаки проходять процедуру стандартизації (Z-score normalization).

Формула стандартизації для кожного значення ознаки x виглядає наступним чином:

$$z = \frac{x - \mu}{\sigma}, \tag{2.8}$$

де z – нове стандартизоване значення;

x – початкове значення ознаки;

μ – математичне сподівання (середнє значення) ознаки по всій вибірці;

σ – середньоквадратичне відхилення ознаки [36].

Після стандартизації всі ознаки матимуть середнє значення, рівне нулю ($\mu = 0$), і дисперсію, рівну одиниці ($\sigma^2 = 1$), що гарантує рівномірний вплив кожної метрики на результат кластеризації.

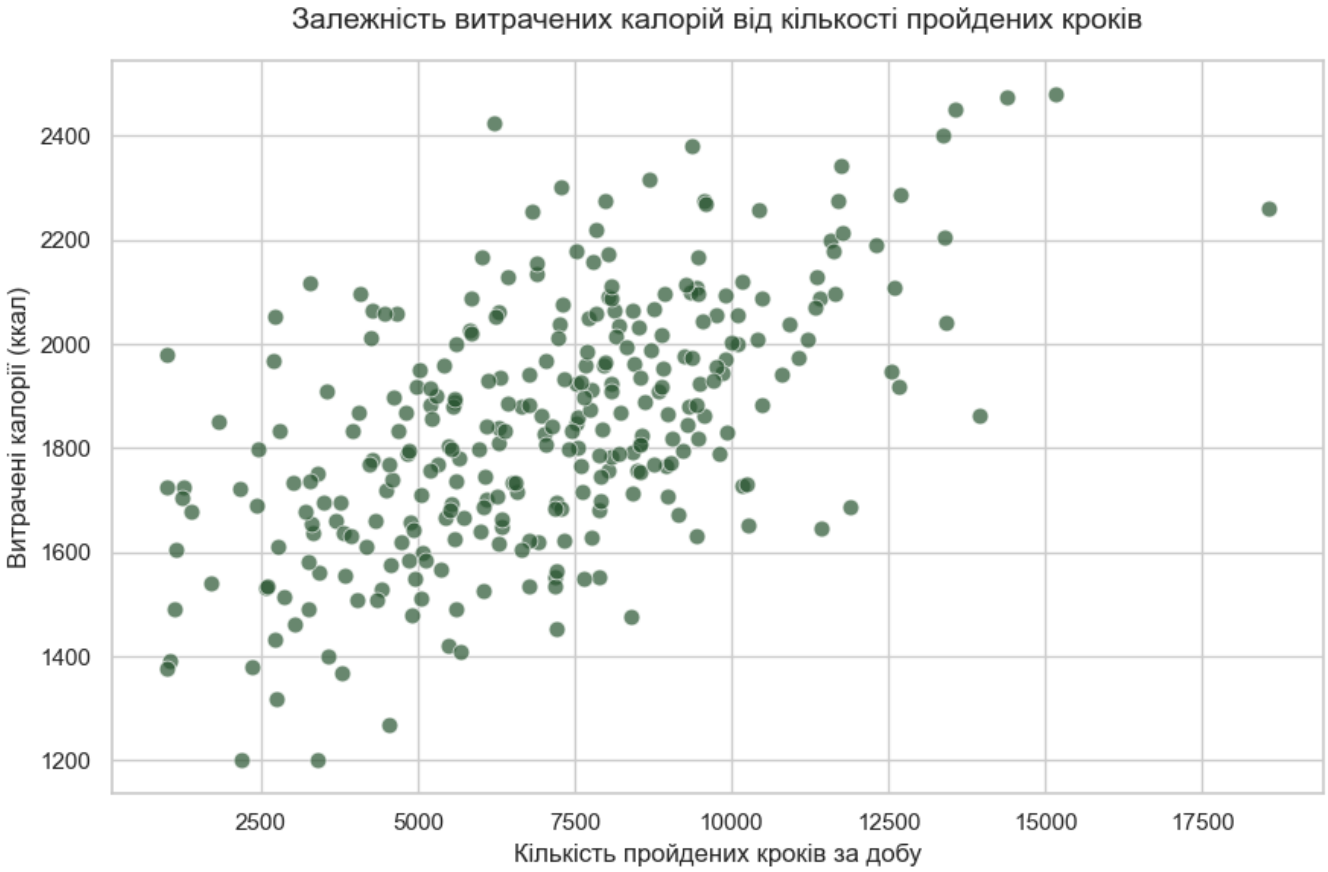

Рисунок 2.4 – Розвідувальний аналіз: залежність між кількістю пройдених кроків та витраченими калоріями

Крім того, проведений кореляційний аналіз дозволив виявити сильні лінійні залежності між певними ознаками. Наприклад, очікувано спостерігається високий коефіцієнт кореляції Пірсона між TotalSteps та Calories. Ці залежності в подальшому будуть використані алгоритмом Випадковий ліс для підвищення точності прогнозування [37].

Висновки до розділу 2

У другому розділі було проведено ґрунтовний аналіз методів та інформаційних технологій, необхідних для проєктування інтелектуальної системи персоналізації фізичної активності. Обґрунтовано вибір алгоритму кластеризації K-Means, який дозволяє ефективно сегментувати користувачів на групи за рівнем їхнього щоденного навантаження. Розглянута математична модель алгоритму та використання «методу ліктя» забезпечують автоматичне й точне формування узагальнених профілів користувачів на основі їхніх біометричних показників.

Для задачі прогнозування майбутньої фізичної активності обрано ансамблевий метод Випадкового лісу. Завдяки своїй архітектурі, побудованій на базі множини незалежних дерев рішень, цей алгоритм гарантує високу точність регресійного аналізу та стійкість до аномальних викидів, що часто зустрічаються у даних з фітнес-трекерів. Крім того, був детально проаналізований набір даних FitBit, визначено ключові атрибути та етапи їхньої попередньої підготовки, зокрема очищення та математичну стандартизацію значень.

Визначений технологічний стек, що включає мову програмування Python, бібліотеку машинного навчання Scikit-learn, локальну реляційну базу даних SQLite з ORM SQLAlchemy та сучасну бібліотеку для створення графічного інтерфейсу CustomTkinter, повністю задовольняє вимоги проєкту. Теоретична, алгоритмічна та технологічна база, сформована в цьому розділі, є надійним підґрунтям для переходу до етапу безпосередньої програмної реалізації десктопного застосунку.

3 ПРОЄКТУВАННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ ПЕРСОНАЛІЗАЦІЇ ФІЗИЧНОЇ АКТИВНОСТІ

3.1 Проєктування функціональної структури

Процес розробки будь-якого складного програмного забезпечення, зокрема інтелектуальної системи персоналізації фізичної активності, неможливий без попереднього формування чітких вимог та проєктування архітектури. На цьому етапі визначаються основні функції, які має виконувати система, обмеження, в межах яких вона повинна працювати, а також взаємодія між її окремими модулями.

Першим кроком було формування специфікації вимог до програмного забезпечення. Вимоги було розділено на дві великі категорії: функціональні та нефункціональні.

Функціональні вимоги визначають, що саме повинна робити система:

– зберігання даних користувача: система повинна дозволяти зберігати в профілі користувача його базові антропометричні дані (вік, вага, зріст, стать);

– збір та збереження даних: програма має надавати інтерфейс для ручного введення (кількість кроків, спалені калорії, кількість активних хвилин, години сну, кількість випитої води). Всі дані повинні надійно зберігатися в локальній базі даних;

– аналітична обробка (кластеризація): система повинна автоматично обробляти збережені дані за допомогою алгоритму машинного навчання K-Means та відносити користувача до певної групи активності (наприклад, «Низька активність», «Помірна активність», «Висока активність»);

– прогнозування: на основі історичних даних система повинна використовувати модель Випадкового лісу (Random Forest) для прогнозування фізичних показників користувача на наступний день;

– візуалізація: програма повинна генерувати зрозумілі графіки, які відображають динаміку фактичної активності користувача та прогнозованої активності;

– надання рекомендацій: на основі результатів кластеризації та прогнозування, система має генерувати персоналізовані текстові рекомендації щодо коригування фізичних навантажень.

Нефункціональні вимоги визначають атрибути якості та обмеження системи:

– продуктивність: алгоритми машинного навчання повинні виконувати кластеризацію масиву з 10 000 записів не довше ніж за 5 секунди на стандартному персональному комп’ютері;

– ергономічність (UI/UX): графічний інтерфейс має бути інтуїтивно зрозумілим, підтримувати сучасний дизайн (заокруглені елементи) та мати можливість перемикання між світлою і темною темами для зменшення навантаження на очі користувача;

– автономність: система повинна бути реалізована як десктопний застосунок і функціонувати повністю автономно, без необхідності постійного підключення до мережі Інтернет (всі розрахунки та зберігання даних відбуваються локально).

Для візуального моделювання вимог та архітектури системи було використано уніфіковану мову моделювання (UML – Unified Modeling Language). Головним інструментом для відображення взаємодії між користувачем та системою є діаграма прецедентів (Use Case Diagram).

На діаграмі прецедентів розроблюваної системи виділено одного головного актора (Actor) – «Користувач» (User). Цей актор ініціює всі основні процеси в програмі. Діаграма включає наступні ключові прецеденти:

– «Запустити систему»;

– «Ввести нові показники активності»;

– «Переглянути дашборд»;

– «Отримати предиктивну аналітику активності»;

– «Отримати план тренувань».

Важливою особливістю архітектури є наявність внутрішніх (системних) прецедентів, які запускаються автоматично. Наприклад, прецедент «Ввести нові

показники активності» обов’язково включає (відношення <<include>>) прецедент «Оновити кластер користувача», оскільки після кожної зміни даних система повинна перерахувати центроїди алгоритмом K-Means. Аналогічно, прецедент «Отримати персональні рекомендації» включає «Згенерувати прогноз (Random Forest)».

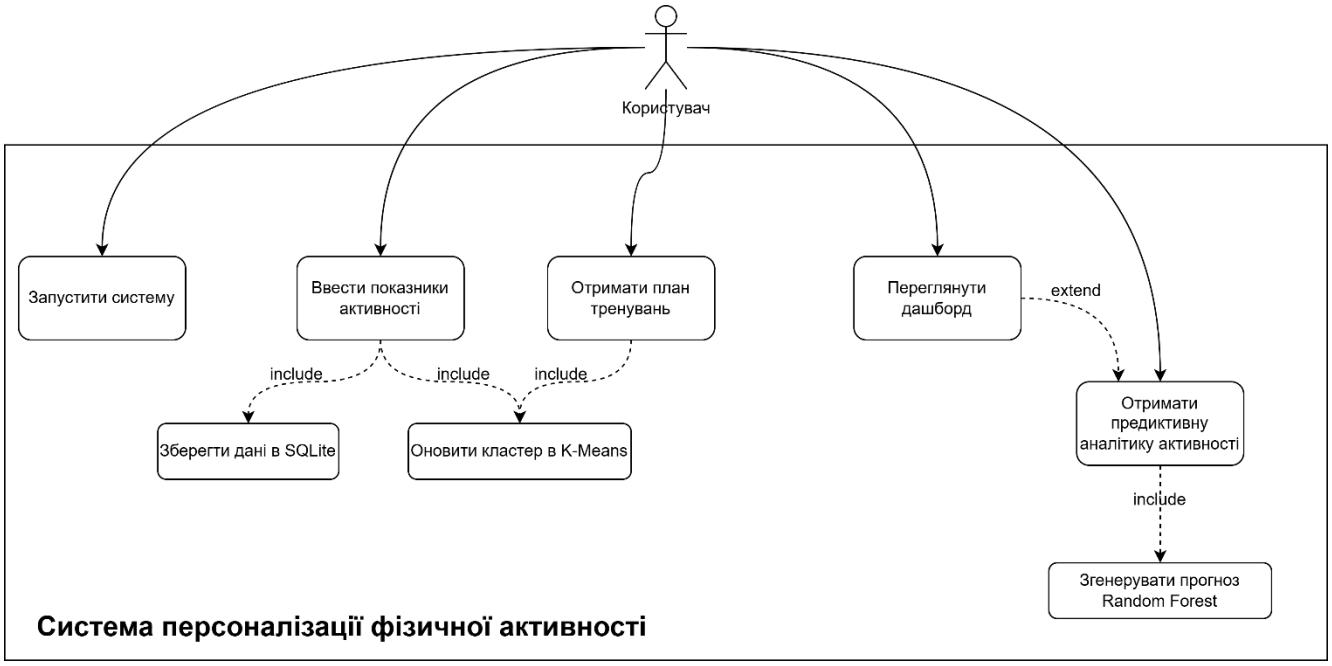

Рисунок 3.1 – Діаграма прецедентів розроблюваної системи

Як видно з рис. 3.1, архітектура розроблюваної системи персоналізації фізичної активності спроєктована таким чином, щоб максимально спростити взаємодію для кінцевого користувача. Уніфікована мова моделювання (UML) дозволяє наочно продемонструвати межі системи, які на схемі позначені великим зовнішнім прямокутником. Усе, що знаходиться всередині цього прямокутника, є програмними функціями, а те, що ззовні – зовнішніми сутностями.

Основним і єдиним зовнішнім актором (Actor) у цій моделі є «Користувач». З точки зору системи, користувач – це фізична особа, яка взаємодіє з програмою через графічний інтерфейс для моніторингу свого здоров’я. Актор ініціює чотири базові прецеденти (Use Cases), які є точками входу в систему:

1) прецедент «запустити систему». Оскільки система розрахована на зберігання біометричних даних (вага, зріст, вік, стать), кожен користувач повинен мати власний профіль. Цей прецедент забезпечує ідентифікацію користувача та завантаження його персональної статистики з локальної бази даних SQLite при кожному запуску застосунку;

2) прецедент «Ввести показники активності». Це найважливіший сценарій для збору даних. Користувач щоденно взаємодіє з цим прецедентом, вводячи кількість пройдених кроків, спалені калорії та хвилини інтенсивного навантаження. Цей прецедент є критично необхідним, оскільки саме ці введені дані формують навчальну вибірку (dataset) для подальшої роботи алгоритмів машинного навчання. Без регулярного введення даних система не зможе адаптуватися під зміну фізичного стану користувача;

3) прецедент «Переглянути дашборд». Цей функціонал відповідає за візуалізацію даних. Користувач може ініціювати цей прецедент, щоб побачити динаміку своїх показників у вигляді теплового календаря. Важливо зазначити, що цей прецедент виконує лише функцію читання (Read-only) і жодним чином не модифікує дані в системі;

4) прецедент «Отримати предиктивну аналітику активності». Це ключовий аналітичний прецедент системи. Користувач звертається до нього, коли потребує експертної оцінки своїх результатів у динаміці за останні 7 днів. Даний прецедент обов’язково включає прихований процес «Згенерувати прогноз Random Forest», який на основі історичних даних формує лінію очікуваної поведінки (план) та порівнює її з фактичними результатами, надаючи користувачу текстові рекомендації щодо виконання чи відставання від графіка;

5) прецедент «Отримати план тренувань». Цей прецедент забезпечує користувача персоналізованим планом активності на тиждень вперед. Використовуючи результати кластеризації (K-Means), система ідентифікує поточний рівень користувача (базовий, інтенсивний або відновлення) для кожного

дня тижня і генерує конкретні кількісні цілі (кроки, вода, сон, активні хвилини) та якісні рекомендації щодо режиму тренувань.

Для забезпечення глибини архітектурного проєктування, на діаграмі (рис. 3.1) також відображено системні (приховані) прецеденти за допомогою UML-стереотипів <<include>> (обов'язкове включення) та <<extend>> (опціональне розширення). Це деталізує логіку роботи програми:

– використання відношення <<include>> (Включення). Цей стереотип показує, що виконання одного прецеденту неможливе без автоматичного виклику іншого. Як показано на схемі, коли користувач виконує дію «Ввести показники активності», система автоматично генерує дві приховані події. По-перше, викликається прецедент «Зберегти дані в SQLite», який гарантує цілісність інформації за допомогою ORM SQLAlchemy та фіксує транзакцію. По-друге, миттєво викликається прецедент «Оновити кластер в K-Means». Оскільки користувач додав нові дані про свою активність, система повинна перерахувати центроїди кластерів і, за необхідності, перевести користувача до іншої групи (наприклад, з «Помірної активності» у «Високу»). Прецедент «Отримати план тренувань» обов'язково включає прихований виклик прецеденту «Оновити кластер в K-Means». Це означає, що перед видачею плану система автоматично перераховує центроїди кластерів на основі найсвіжіших даних користувача і, за необхідності, переводить його до іншої групи інтенсивності. Аналогічно, дія «Отримати предиктивну аналітику активності» обов'язково включає прихований виклик функції «Згенерувати прогноз Random Forest», оскільки поради формуються не з поточних даних, а з передбачень математичної моделі на майбутнє.

– Використання відношення <<extend>> (Розширення). На відміну від включення, розширення спрацьовує лише за певних умов або за бажанням користувача. На діаграмі цей зв'язок встановлено між «Переглянути дашборд» та «Отримати предиктивну аналітику активності». Це означає, що перегляд статистики є абсолютно самостійним процесом. Користувач може просто

подивитися на графік і закрити програму. Однак, знаходячись на екрані дашборду, він може скористатися додатковою можливістю (розширити базовий функціонал) і натиснути кнопку для отримання рекомендацій. Тобто прецедент рекомендацій лише доповнює прецедент перегляду графіків.

Підсумовуючи аналіз діаграми прецедентів, можна стверджувати, що обрана архітектура відповідає принципу інкапсуляції: кінцевий користувач взаємодіє лише з простими та зрозумілими інтерфейсами, тоді як усі складні процеси запису в базу даних, стандартизації матриць ознак та перенавчання моделей штучного інтелекту відбуваються автоматично на фоновому рівні системи.

3.2 Проєктування фізичної та логічної структури бази даних

Важливим етапом розробки програмного продукту є проєктування моделі даних, яка здатна ефективно зберігати інформацію про користувачів, їхні антропометричні показники, історію фізичної активності (time-series data) та прогнози предиктивних моделей. Враховуючи, що застосунок розробляється як автономне десктопне рішення, було прийнято рішення відмовитися від складних клієнт-серверних баз даних (наприклад, PostgreSQL або MySQL) на користь вбудованої системи керування базами даних (СКБД) – SQLite.

Вибір SQLite обґрунтований її високою продуктивністю при локальній роботі та тим, що вся база даних зберігається у вигляді єдиного файлу (app_database.db). Це значно спрощує розгортання програми на комп’ютерах кінцевих користувачів.

Взаємодія програмного коду на мові Python із базою даних реалізована за допомогою технології ORM (Object-Relational Mapping), зокрема через використання бібліотеки SQLAlchemy. Такий підхід дозволяє працювати з таблицями бази даних як з об’єктами (класами) мови програмування, що мінімізує кількість «сирих» SQL-запитів, підвищує безпеку даних та полегшує подальшу підтримку коду.

Для візуального відображення структури бази даних, її сутностей, атрибутів та зв’язків між ними було побудовано UML-діаграму класів (рис. 3.2). Використання нотації UML є доцільним, оскільки в архітектурі ORM кожна таблиця бази даних безпосередньо репрезентується відповідним класом Python.

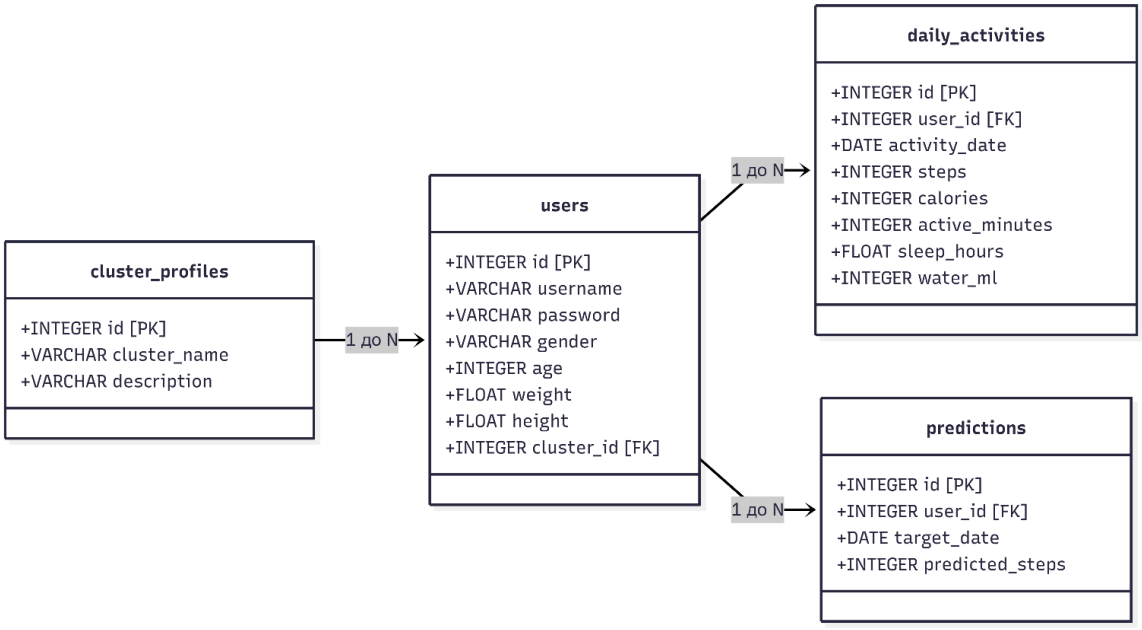

Рисунок 3.2 – UML-діаграма класів (модель бази даних застосунку)

Розроблена база даних знаходиться у третій нормальній формі (3NF), що гарантує відсутність дублювання даних та аномалій оновлення. Архітектура бази даних складається з чотирьох взаємопов’язаних таблиць.

Центральною сутністю системи є таблиця users (Користувачі). Вона зберігає як облікові дані для безпечної авторизації у системі, так і ключові антропометричні параметри, які використовуються для розрахунку базового рівня метаболізму (BMR) та кластеризації. Структура таблиці users наведена у таблиці 3.1.

Таблиця 3.1 – Структура таблиці User

Поле	**Тип даних**	**Обмеження**	**Опис**
id	INTEGER	Primary Key	Унікальний ідентифікатор користувача (Auto Increment)

Кінець таблиці 3.1

username	VARCHAR(50)	Not Null, Unique	Логін (ім’я) користувача в системі
password	VARCHAR(255)	Not Null	Хеш-сума пароля для безпечної авторизації
gender	VARCHAR(20)		Стать користувача (необхідна для коректного розрахунку витрати калорій)
age	INTEGER	Not Null	Повний вік користувача у роках
weight	FLOAT	Not Null	Маса тіла користувача у кілограмах
height	FLOAT	Not Null	Зріст користувача у сантиметрах
cluster_id| INTEGER	Foreign Key	Посилання на ідентифікатор сегмента (кластера), до якого належить користувач	

Поле cluster_id виступає зовнішнім ключем до таблиці cluster_profiles, реалізуючи зв’язок «один-до-багатьох» (1:N), оскільки до одного кластера може належати велика кількість користувачів. Таблиця cluster_profiles (Профілі кластерів) виконує роль статичного довідника, що містить результати роботи алгоритму машинного навчання K-Means. Її структура описана у таблиці 3.2.

Таблиця 3.2 — Структура таблиці cluster_profiles

Поле	**Тип даних**	**Обмеження**	**Опис**
id	INTEGER	Primary Key	Ідентифікатор кластера (наприклад: 0, 1, 2)
cluster_name	VARCHAR(100)	Not Null	Текстова назва сегмента (наприклад: «Базовий», «Спортсмени»)
description	VARCHAR	Not Null	Детальний текстовий опис поведінкових характеристик профілю

Для зберігання щоденної статистики користувача розроблено транзакційну таблицю daily_activities (Щоденна активність). Ця таблиця є критично важливою, оскільки накопичені у ній дані формують часовий ряд (time-series), який слугує датасетом (X-matrix) для навчання алгоритму прогнозування Random Forest. Зв’язок із користувачем реалізовано через зовнішній ключ user_id відношенням 1:N. Структура наведена у таблиці 3.3.

Таблиця 3.3 – Структура таблиці daily_activities

Поле	**Тип даних**	**Обмеження**	**Опис**
id	INTEGER	Primary Key	Унікальний ідентифікатор запису журналу
user_id	INTEGER	Foreign Key, Not Null	Ідентифікатор користувача (власника запису)

Кінець таблиці 3.3

activity_date	DATE	Not Null	Календарна дата, за яку внесено показники (формат DD-MM-YYYY)
steps	INTEGER	Not Null	Кількість кроків, пройдених за вказану добу
calories	INTEGER	Not Null	Кількість кілокалорій, витрачених під час активності
active_minutes| INTEGER	Not Null	Сумарна тривалість інтенсивного навантаження у хвилинах	
sleep_hours	FLOAT		Тривалість нічного сну (в годинах) для аналізу відновлення
water_ml	INTEGER		Об’єм спожитої води (в мілілітрах) для контролю гідратації

Оскільки програмний продукт використовує елементи предиктивної аналітики, до бази даних було додано таблицю predictions (Прогнози). Вона призначена для кешування результатів розрахунків моделі RandomForestRegressor. Це дозволяє швидко виводити спрогнозовані показники (наприклад, очікувану кількість кроків) у графічному інтерфейсі та в майбутньому аналізувати показник середньоквадратичної помилки (MSE) між прогнозом і фактом. Структуру наведено у таблиці 3.4.

Таблиця 3.4 – Структура таблиці predictions

Поле	**Тип даних**	**Обмеження**	**Опис**
id	INTEGER	Primary Key	Унікальний ідентифікатор запису прогнозу
user_id	INTEGER	Foreign Key, Not Null	Ідентифікатор користувача, для якого створено прогноз
target_date	DATE	Not Null	Майбутня дата, на яку розраховано показники
predicted_steps| INTEGER	Not Null	Прогнозована (очікувана) кількість кроків	

Останньою таблицею в архітектурі є Prediction, яка призначена для збереження результатів роботи алгоритмів машинного навчання. Зберігання прогнозів дозволяє системі порівнювати їх з реальними результатами, що користувач введе наступного дня, і тим самим розраховувати похибку моделі. Структура наведена в Таблиці 3.4.

Таблиця 3.4 – Фізична структура таблиці Prediction

Назва поля	**Тип даних**	**Обмеження**	**Опис призначення поля**
id	INTEGER	Primary Key, Auto Increment	Унікальний ідентифікатор прогнозу
user_id	INTEGER	Foreign Key, Not Null	Прив’язка прогнозу до профілю користувача

Кінець таблиці 3.4

target_date	DATE	Not Null	Дата, на яку було згенеровано прогноз алгоритмом
predicted_steps| INTEGER	Not Null	Згенерована алгоритмом цільова кількість кроків	

Всі зв’язки (Foreign Keys) між таблицями імплементовані безпосередньо на рівні рушія SQLite (через увімкнення прагми PRAGMA foreign_keys = ON). Це гарантує строгу каскадну цілісність даних (Referential Integrity): система не дозволить створити запис журналу активності або прогноз, якщо відповідний користувач (user_id) не існує або був видалений. Крім того, завдяки суворій типізації колонок (INTEGER, FLOAT) нівелюється ризик збереження некоректних даних.

3.3 Проєктування інтерфейсу користувача

Після завершення етапу проєктування архітектури та бази даних, наступним критично важливим кроком є перевірка функціональності та зручності графічного інтерфейсу (GUI). Інтерфейс виступає єдиною точкою взаємодії (точкою контакту) між складною математичною логікою машинного навчання та кінцевим користувачем. З огляду на те, що цільовою аудиторією застосунку є люди з різним рівнем технічної підготовки, головною вимогою до дизайну була максимальна простота, інтуїтивна зрозумілість та відсутність візуального перевантаження.

Для демонстрації роботи розробленої системи було проведено запуск десктопного застосунку. При ініціалізації головного файлу програми користувача зустрічає базове вікно, яке структурно поділене на дві ключові області: статичну навігаційну панель (Sidebar) зліва та динамічну робочу область (Main Frame) справа.

Розглянемо детальніше навігаційну панель, яка виконує роль головного диспетчера екранів (рис. 3.3).

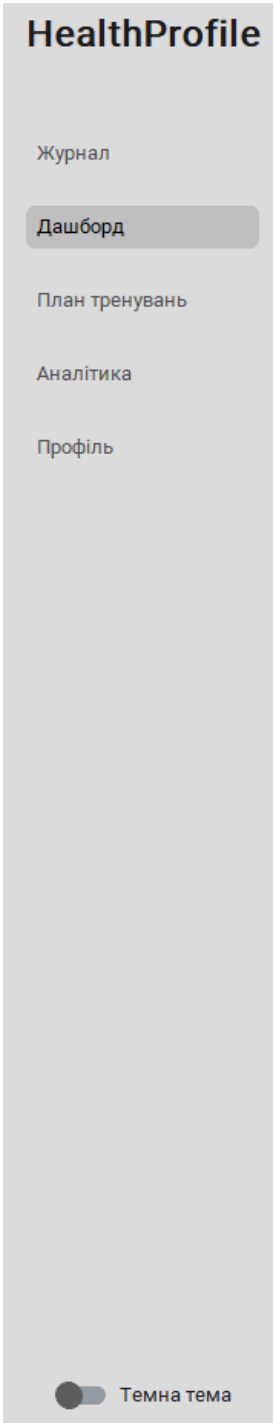

Рисунок 3.3 – Навігаційна панель системи «HealthProfile»

Як детально показано на Рисунку 3.3, ліва панель має світло-сірий фон, що створює плавний контраст із білою робочою областю. У самому верху панелі

розміщено текстовий логотип «HealthProfile», виконаний напівжирним шрифтом. Цей елемент не лише виконує функцію брендування програмного продукту, але й слугує візуальним якорем для користувача.

Основний блок навігації складається з п’яти інтерактивних кнопок:

– «Журнал»: кнопка, що відповідає за виклик форми додавання нових біометричних записів у базу даних;

– «Дашборд»: базова кнопка, що повертає користувача на головний аналітичний екран;

– «План тренувань»: кнопка, що відповідає за виклик рекомендованого плану тренувань на основі кластеризації K-Means;

– «Аналітика»: кнопка переходу до модуля предиктивної аналітки (демонстрація роботи алгоритму Random Forest);

– «Профіль»: розділ, присвячений редагуванню та зберіганню антропометричним даним користувача.

Важливою деталлю UX-дизайну (User Experience) є реалізація ефекту наведення (hover effect). При наведенні курсора миші на будь-яку з неактивних кнопок, її фон плавно змінюється на синій, інформуючи користувача про можливість взаємодії. Активна ж кнопка (екран якої наразі відкритий) має постійну сіру заливку, чітко показуючи поточне місцезнаходження в системі. Вигляд ефекту наведення представлений на рис. 3.4.

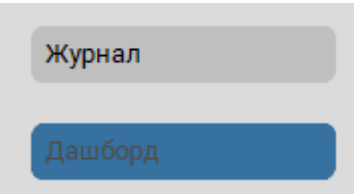

Рисунок 3.4 – Ефект наведення

У нижньому лівому куті навігаційної панелі розміщено елемент управління «Темна тема» (перемикач типу Toggle Switch). Використання сучасного фреймворку CustomTkinter дозволило реалізувати зміну кольорової палітри

застосунку (з білої на темно-сіру) «на льоту». Це надзвичайно важлива функція для додатків категорії Health & Fitness, оскільки вона дозволяє знизити навантаження на очі користувача при використанні програми у вечірній або нічний час. На рис. 3.5 представлений перемикач кольорової палітри.

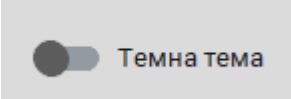

Рисунок 3.5 – Вимкнений перемикач теми

При натисканні на першу кнопку навігаційного меню, система генерує та відображає у правій частині вікна головний екран аналітики – «Дашборд активності» (рис. 3.6).

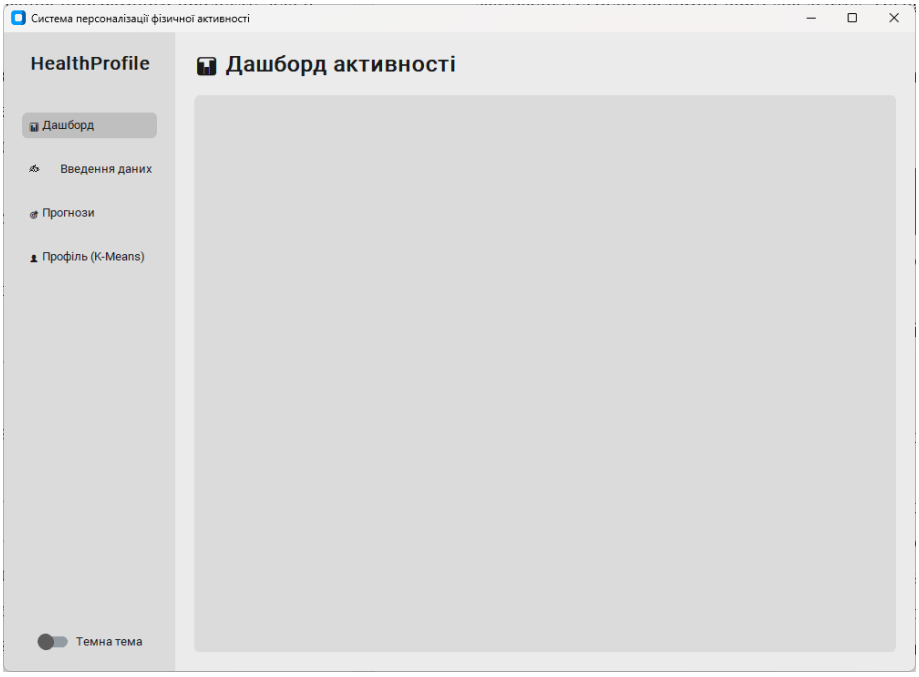

Рисунок 3.6 – Робоча область головного екрану «Дашборд активності»

На рис. 3.6 зображено базову структуру екрана візуалізації. Верхня частина робочої області зарезервована під великий заголовок (Header), який динамічно змінюється залежно від обраного розділу. Це гарантує, що користувач завжди розуміє контекст поточного вікна.

Нижче заголовка розташований великий фрейм (контейнер) із заокругленими кутами. На етапі ініціалізації програми цей контейнер є порожнім (як показано на скріншоті), проте під час повноцінної роботи системи він динамічно заповнюється графіками бібліотеки Matplotlib та інформаційними картками KPI (Key Performance Indicators). Вільний простір (Negative space) навколо контейнера додає інтерфейсу «повітря», роблячи його сучасним та відповідним стандартам Material Design.

Для повноцінної роботи алгоритмів машинного навчання системі необхідні дані. Для збору цих даних розроблено спеціальний екран «Введення щоденних показників», перехід на який здійснюється через другу кнопку меню (рис. 3.7).

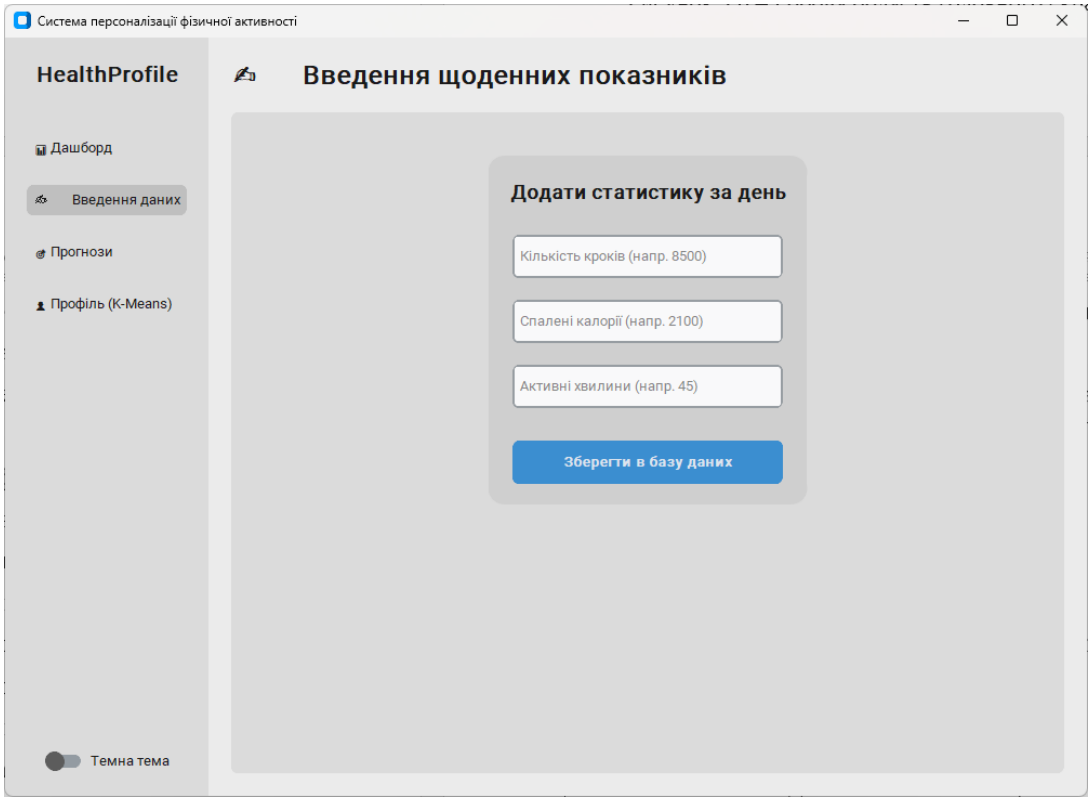

Рисунок 3.7 – Інтерфейс екрана введення біометричних даних

Як видно з Рисунка 3.7, форма збору щоденної статистики реалізована у вигляді централізованої картки (Card UI pattern), розташованої рівно по центру робочої області. Такий підхід фокусує 100% уваги користувача виключно на процесі введення інформації, мінімізуючи фактори відволікання.

Картка містить заголовок «Додати статистику за день» та три поля введення тексту (віджети типу Entry):

– поле «Кількість кроків». Призначене для введення цілочисельного значення кроків;

– поле «Спалені калорії». Фіксує енергетичні витрати організму у кілокалоріях;

– поле «Активні хвилини». Записує час інтенсивного кардіонавантаження.

Продуманою деталлю дизайну є використання плейсхолдерів – сірого тексту-підказки безпосередньо всередині полів (наприклад, «напр. 8500»). Цей текст зникає в момент, коли користувач починає введення. Плейсхолдери слугують інструкцією та демонструють очікуваний формат даних (тільки числа), запобігаючи введенню некоректних символів (наприклад, літер або спеціальних знаків).

Під полями введення розміщено масивну кнопку підтвердження «Зберегти в базу даних», виділену акцентним синім кольором (Primary Call-to-Action). Синій колір зазвичай асоціюється в користувачів із надійністю та підтвердженням дій.

При натисканні на цю кнопку алгоритм системи не просто переносить текст у базу даних, а виконує жорстку програмну валідацію. Система перевіряє:

– чи не є поля порожніми;

– чи можна конвертувати введений текст у цілий тип даних (Integer);

– чи не є введені значення від’ємними (адже не можна пройти -100 кроків).

Тільки після успішного проходження всіх цих перевірок дані формуються у SQL-транзакцію, записуються в локальний файл бази даних app_database.db і стають доступними для подальшої обробки модулем кластеризації K-Means. У разі помилки користувачеві виводиться відповідне попередження. Подібний механізм захисту від «дурня» (Foolproof) є критично необхідним для забезпечення чистоти датасету машинного навчання.

Таким чином, демонстрація роботи системи підтверджує, що розроблений графічний інтерфейс є повністю працездатним, логічно структурованим та здатним

забезпечити безперебійний збір і візуалізацію даних для персоналізації фізичної активності.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи було здійснено детальний етап проєктування архітектури розроблюваної інформаційної системи персоналізації фізичної активності (HealthProfile). На основі проведеного аналізу вимог до програмного продукту, було сформовано функціональну структуру системи. За допомогою інструментарію уніфікованої мови моделювання (UML), зокрема діаграми прецедентів (Use Case), чітко визначено роль єдиного актора – «Користувача» - та базові сценарії його взаємодії з програмою, включаючи процеси авторизації, введення статистики, візуалізації прогресу та запиту рекомендацій через включені та розширені прецеденти.

Критично важливим етапом стало проєктування логічної та фізичної структури збереження даних. Розроблена концептуальна ER-модель бази даних реляційного типу дозволила структурувати інформацію у чотири ключові сутності: таблиці користувачів, щоденної активності, профілів кластерів та збережених прогнозів. Нормалізація бази даних до третьої нормальної форми (3NF) та імплементація технології об’єктно-реляційного відображення (ORM) у поєднанні з СКБД SQLite гарантують цілісність, відсутність дублювання даних та високу швидкість агрегації датасетів для їх подальшого використання модулями машинного навчання.

Також було спроєктовано та описано графічний інтерфейс користувача (GUI). Побудований на базі фреймворку CustomTkinter, інтерфейс реалізує принципи мінімалістичного дизайну, забезпечуючи інтуїтивно зрозумілу навігацію за допомогою бокової панелі та динамічного дашборду. Впровадження механізмів програмної валідації на формах введення даних гарантує збір коректної інформації, що є необхідною умовою для якісної роботи інтелектуальних аналітичних модулів системи.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ЗАСТОСУНКУ

4.1 Вибір середовища розробки та архітектура проєкту

Практична реалізація інформаційної системи персоналізації фізичної активності здійснювалася мовою програмування Python. У якості інтегрованого середовища розробки (IDE) було обрано PyCharm, що надає зручні інструменти для управління віртуальними середовищами та рефакторингу коду.

Для побудови графічного інтерфейсу користувача (GUI) використано сучасну бібліотеку customtkinter, яка базується на стандартному tkinter, але дозволяє створювати сучасні віджети з підтримкою темної та світлої тем оформлення, що відповідають принципам Material Design.

Робота з даними та алгоритмами машинного навчання реалізована за допомогою бібліотек pandas, scikit-learn та numpy. Візуалізація аналітичних даних забезпечується бібліотекою matplotlib.

Архітектура програмного продукту побудована за модульним принципом, що забезпечує гнучкість системи. Дерево проєкту складається з наступних основних директорій:

– data – директорія зберігання локальної бази даних app_database.db та датасетів (наприклад, fitbit_dataset.csv);

– database – пакет, що відповідає за взаємодію з БД (містить файли налаштування db_config.py та моделі сутностей models.py);

– gui – пакет графічного інтерфейсу, що містить класи головного вікна та всіх робочих фреймів (dashboard_frame.py, input_frame.py, plan_frame.py, prediction_frame.py, profile_frame.py);

– ml_models – пакет інкапсуляції логіки машинного навчання (clustering.py, forecasting.py);

– main.py – головний файл точки входу, що ініціалізує систему.

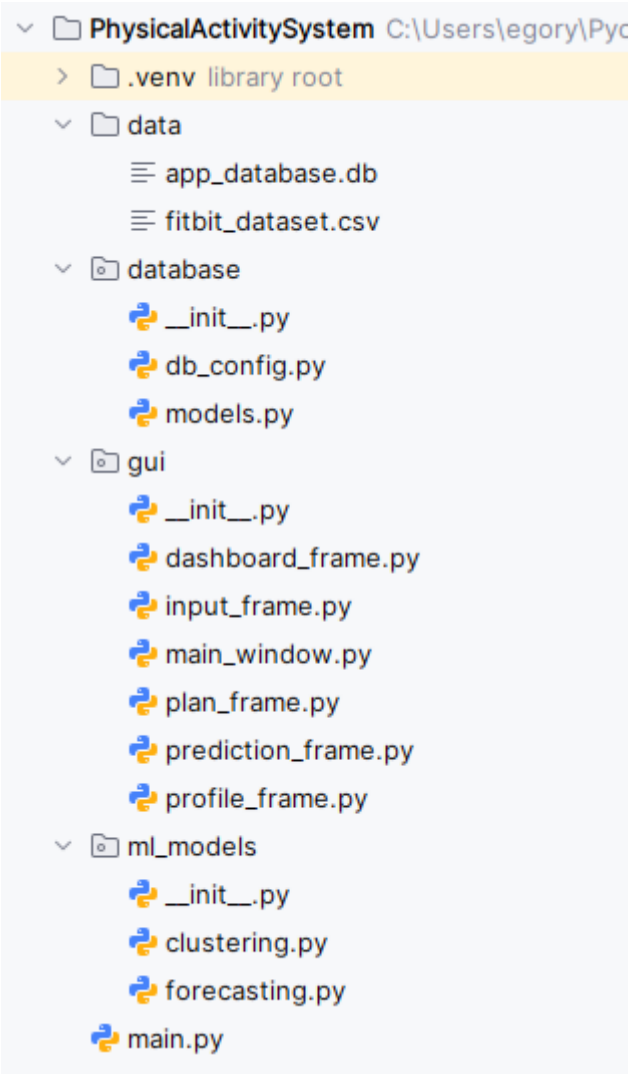

Рисунок 4.1 – Структура проєкту у PyCharm

4.2 Реалізація бази даних системи

Збереження історії фізичної активності та антропометричних даних користувача реалізовано за допомогою вбудованої реляційної СКБД SQLite. Для взаємодії з базою даних у Python використано технологію об’єктно-реляційного відображення (ORM) бібліотеки SQLAlchemy. Це дозволило уникнути написання сирих SQL-запитів та оперувати даними як об'єктами класів.

У файлі database/models.py реалізовано класи User (користувач системи, що містить базові налаштування віку, ваги, зросту) та DailyActivity (щоденна статистика активності). Реалізація класів представлена на рис. 4.2.

```python
class User(Base):  11 usages
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True, autoincrement=True)
    username = Column(String(50), nullable=False, unique=True)

    password = Column(String(255), nullable=True)
    gender = Column(String(20), nullable=True, default="Чоловіча")

    age = Column(Integer, nullable=False, default=25)
    weight = Column(Float, nullable=False, default=70.0)
    height = Column(Float, nullable=False, default=175.0)
    cluster_id = Column(Integer, ForeignKey('cluster_profiles.id'), nullable=True)

    cluster = relationship( argument: 'ClusterProfile', back_populates='users')
    activities = relationship( argument: 'DailyActivity', back_populates='user')
    predictions = relationship( argument: 'Prediction', back_populates='user')


class DailyActivity(Base):  17 usages
    __tablename__ = 'daily_activities'
    id = Column(Integer, primary_key=True, autoincrement=True)
    user_id = Column(Integer, ForeignKey('users.id'), nullable=False)
    activity_date = Column(Date, nullable=False)
    steps = Column(Integer, nullable=False)
    calories = Column(Integer, nullable=False)
    active_minutes = Column(Integer, nullable=False)

    sleep_hours = Column(Float, nullable=True, default=7.0)
    water_ml = Column(Integer, nullable=True, default=2000)

    user = relationship( argument: 'User', back_populates='activities')
```

Рисунок 4.2 – Реалізація моделей таблиць бази даних за допомогою SQLAlchemy

4.3 Розробка графічного інтерфейсу користувача

Головне вікно застосунку App (файл main_window.py) поділено на дві основні логічні зони: ліву бічну панель навігації (Sidebar) та праву динамічну область (Main Frame), куди завантажуються відповідні модулі при натисканні на кнопки меню.

Головним аналітичним центром системи є вкладка «Дашборд» (dashboard_frame.py). У верхній частині екрана розташовано картки ключових показників ефективності (KPI), які відображають останні кроки, середні показники та тривалість сну.

Особливістю даного екрану є розроблений інтерактивний тепловий календар. Дні у календарі підсвічуються різними відтінками зеленого кольору в залежності від рівня активності (кількості кроків) користувача у цей день. При натисканні на конкретну дату в календарі, у правій панелі динамічно завантажується детальна статистика за обраний день: калорії, активність, сон та об’єм спожитої води. На рис. 4.3 представлено загальний вигляд дашборду активності із закритим календарем.

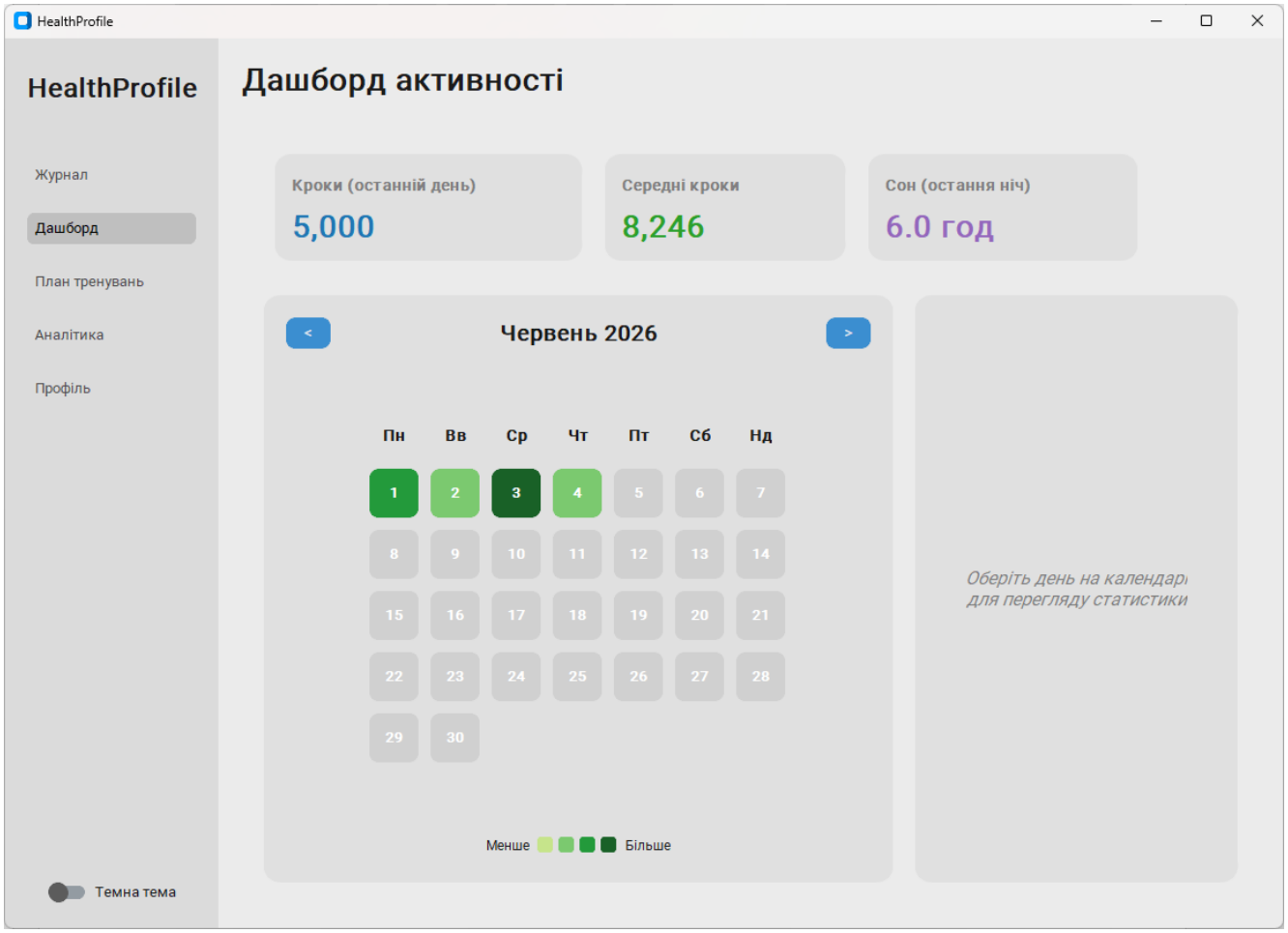

Рисунок 4.3 – Інтерфейс дашборду активності з тепловим календарем

На рис. 4.4 представлений тепловий календар в розгорнутому вигляді з показниками (кроки, калорії, активність, сон, вода).

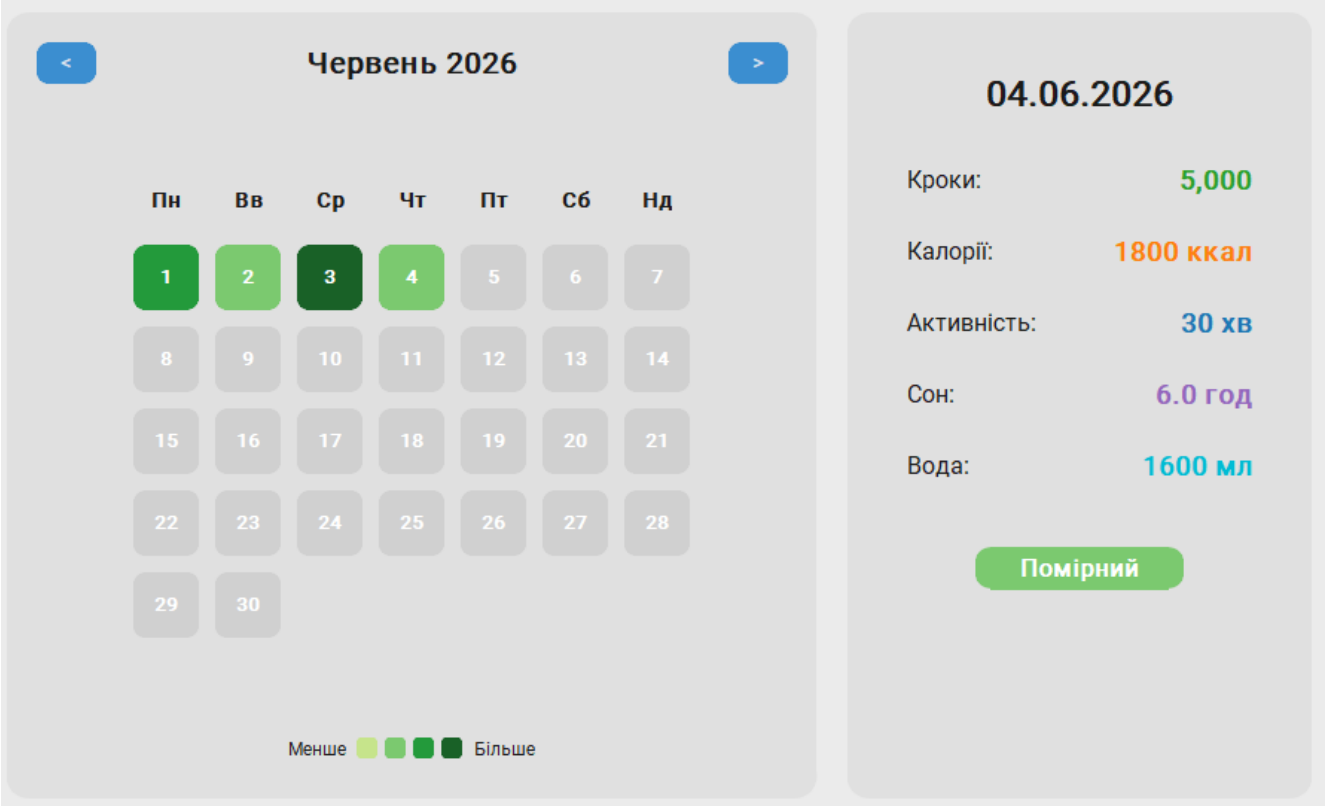

Рисунок 4.4 – Розгорнутий тепловий календар з показниками за обраний день

Для внесення нових даних розроблено вкладку «Журнал» (input_frame.py). Інтерфейс модуля реалізований у вигляді центрованої картки. Для зручності вибору дати було інтегровано віджет з бібліотеки tkcalendar. Під час розробки даного модуля було вирішено проблему втрати фокусу вікна операційної системи Windows шляхом використання методів примусового повернення фокусу головному вікну після закриття спливаючого календаря. На рис. 4.5 наведено загальний вигляд введення даних у журнал.

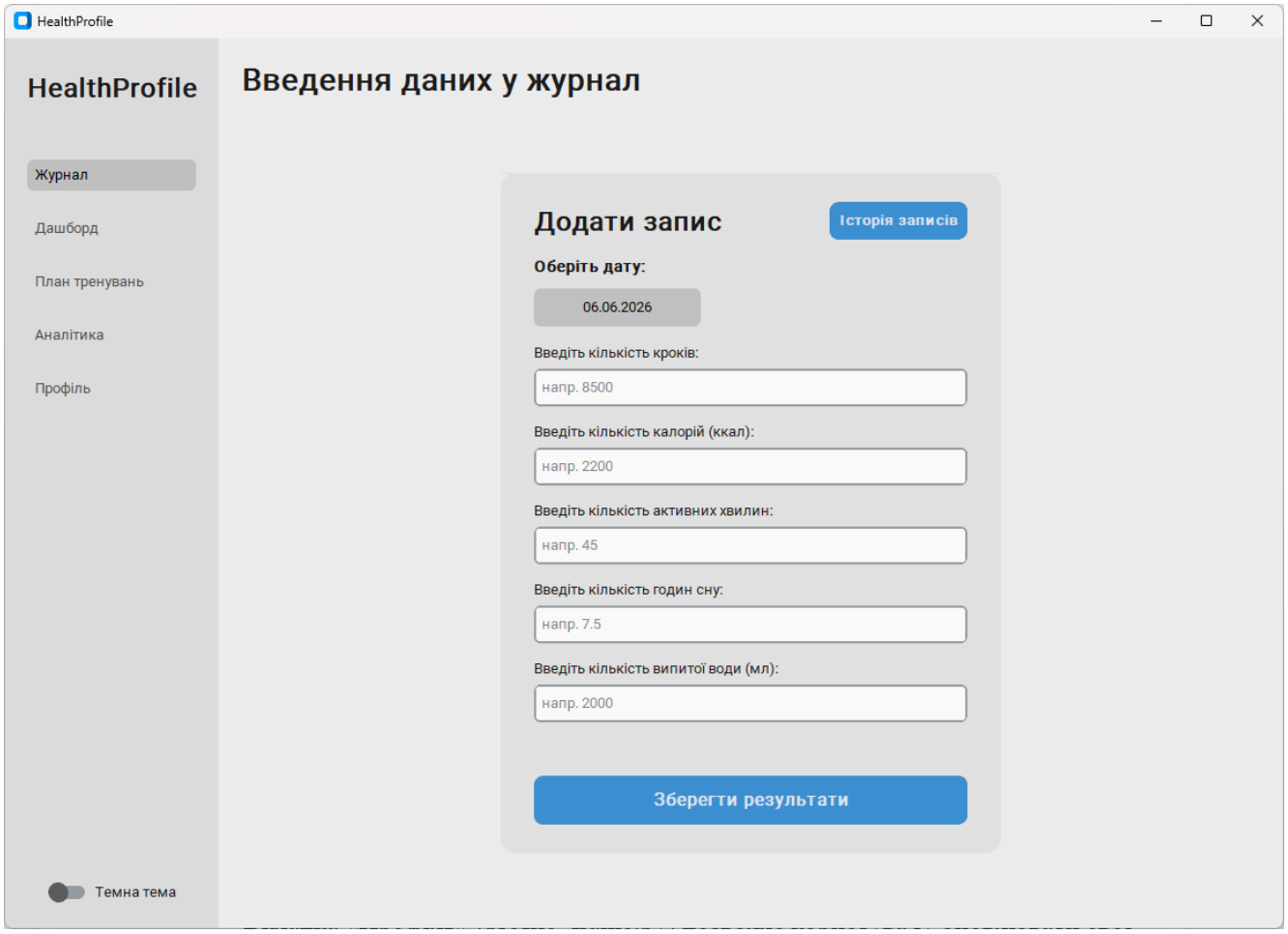

Рисунок 4.5 – Модуль введення щоденної статистики користувача

На рис. 4.6 представлено вигляд віджету вибору дати.

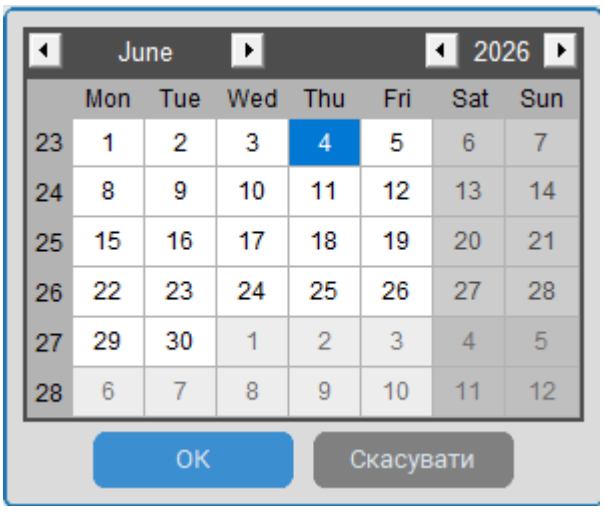

Рисунок 4.6 – Віджет вибору дати

Також в журналі є історія записів користувача, яка забезпечує зручний перегляд усіх даних які буле внесені за всі дні (рис. 4.7).

< Назад

Історія записів

Дата	Кроки	Ккал	Хв.	Сон (год)	Вода (мл)
2026-06-04	5000	1800	30	6.0	1600
2026-06-03	12800	2850	95	7.0	2600
2026-06-02	4500	1900	20	8.0	1800
2026-06-01	8300	2350	50	7.5	2200
2026-05-30	3000	1750	10	9.0	1500
2026-05-29	11500	2750	85	8.0	2600
2026-05-28	8900	2450	60	7.5	2300
2026-05-26	7600	2250	45	7.0	2100
2026-05-25	8100	2350	50	7.5	2200
2026-05-24	5000	2000	30	8.0	1900
2026-05-23	13000	2900	100	8.5	2800
2026-05-21	7200	200	45	7.0	2100
2026-05-20	8800	2450	55	7.5	2200
2026-05-19	4800	1950	25	7.0	1800
2026-05-18	7000	2150	40	7.5	2000
2026-05-16	15500	3300	120	8.0	3200
2026-05-15	9200	2500	60	7.5	2400
2026-05-14	6500	2050	35	6.5	1900
2026-05-12	11000	2700	80	7.0	2600

Рисунок 4.7 – Історія записів

Вкладка «Профіль» (profile_frame.py) дозволяє користувачу оновлювати свої антропометричні параметри (вік, зріст, вагу, стать), які безпосередньо впливають на розрахунок ML-моделей. При збереженні змін система автоматично перераховує індекс маси тіла (IMT), базовий метаболізм (BMR) та добову норму води, виводячи їх на відповідних інформаційних панелях (рис.4.8).

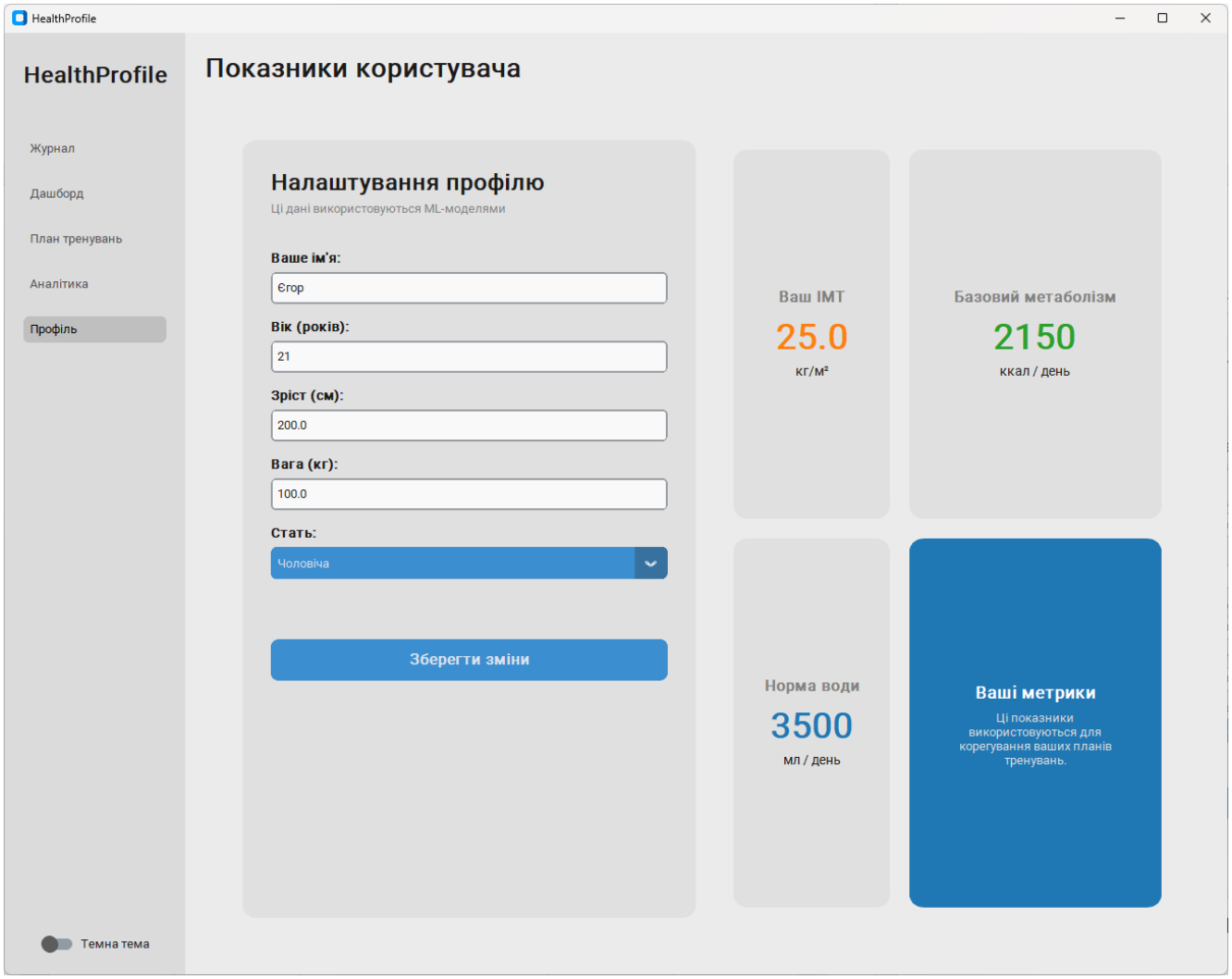

Рисунок 4.8 – Інтерфейс налаштування профілю та метрик здоров’я

4.4 Практична реалізація алгоритмів машинного навчання

Для виконання завдань персоналізації та прогнозування розроблено два незалежні ML-модулі, що вирішують задачі кластеризації та регресії. Взаємодія з моделями здійснюється на льоту, аналізуючи дані, завантажені з SQLite.

Реалізація кластеризації K-Means (План тренувань)

У модулі «План тренувань» (plan_frame.py) реалізовано підсистему генерації індивідуального плану на наступні 7 днів за допомогою алгоритму машинного навчання K-Means.

Система спочатку агрегує історичні дані користувача, групуючи їх за днями тижня, та знаходить середню кількість кроків, витрачених калорій та сну для кожного дня. Після цього дані передаються у модель K-Means, яка розбиває їх на 3 кластери. На основі центрів кластерів система класифікує кожен день як:

– інтенсивний: найактивніший день користувача. Система автоматично збільшує цілі по кроках на 10% та підвищує норму води для відновлення;

– базовий: стандартний рівень активності для підтримки форми;

– відновлення: найменш активні дні (кластер з найнижчим центром). Система генерує рекомендацію сфокусуватися на сні (мінімум 7.5 годин) та легкому стретчингу. Реалізація представлена на рис. 4.9.

```
X = avg_by_day[['steps']].values
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
labels = kmeans.fit_predict(X)

centers = kmeans.cluster_centers_.flatten()
sorted_idx = np.argsort(centers)
recovery_c, base_c, intensive_c = sorted_idx[0], sorted_idx[1], sorted_idx[2]
```

Рисунок 4.9 – Реалізація кластеризації активності алгоритмом K-Means

На основі розрахованих даних система генерує графічний інтерфейс у вигляді карток на кожен день із вказанням конкретних цілей (Кроки, Хвилини, Вода, Сон) та текстовими рекомендаціями (рис. 4.10).

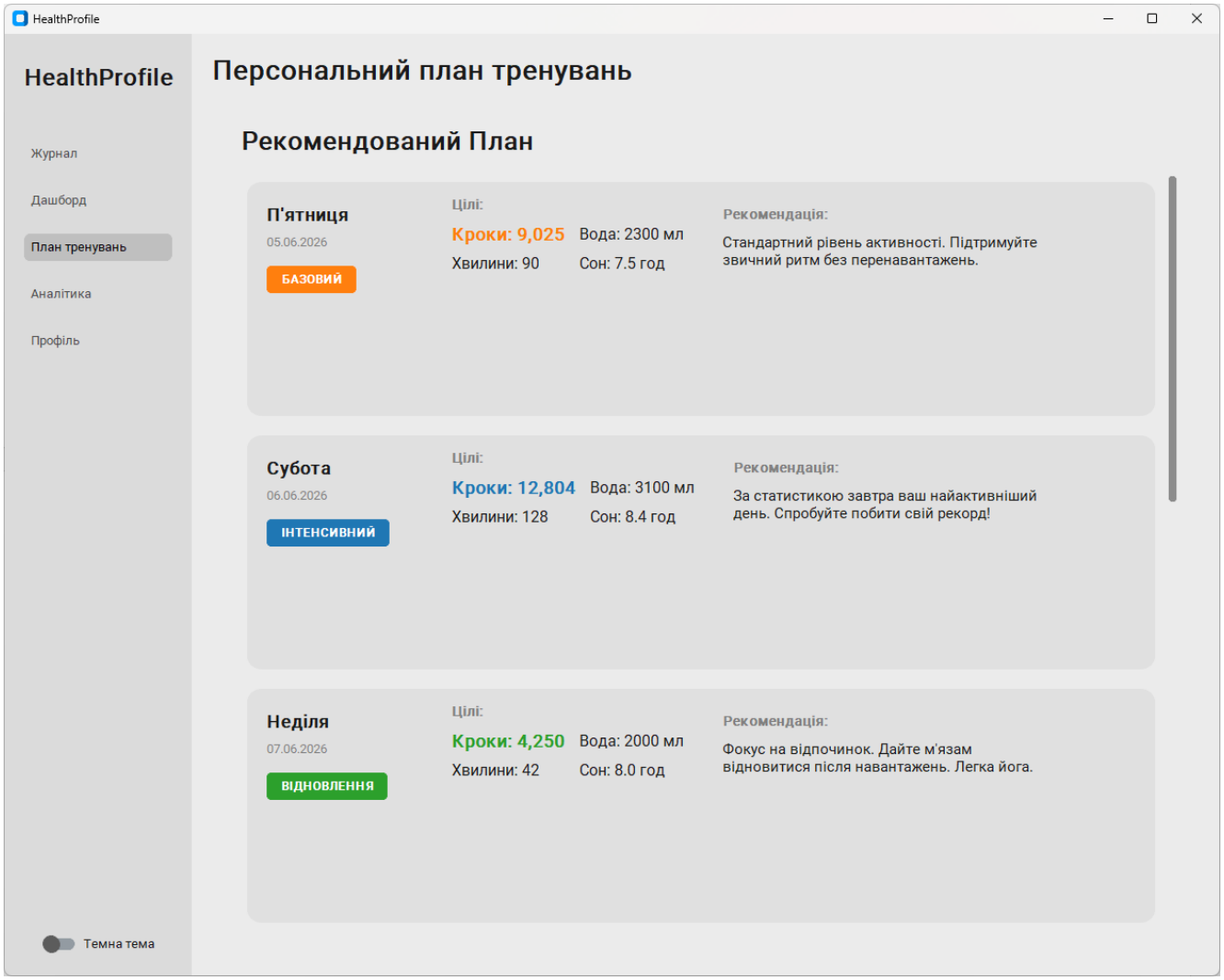

Рисунок 4.10 – Згенерований персональний план тренувань на основі кластеризації

Реалізація предиктивної аналітики (Аналітика)

Для оцінки виконання плану та мотивації користувача розроблено вкладку «Аналітика» (prediction_frame.py). У цьому модулі застосовано алгоритм RandomForestRegressor (Випадковий ліс) для побудови прогнозів активності.

На відміну від статичних розрахунків, алгоритм аналізує історичні патерни. Модель тренується на атрибутах (день тижня, день місяця) та цільовій змінній (наприклад, кількість кроків). Після тренування модель формує прогноз на останні 7 днів, створюючи масив очікуваних даних (План). Реалізація представлена на рис. 4.11.

```
X = df[['weekday', 'day']].values
y = df['value'].values
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X, y)

last_7_days = df.tail(7)
dates = last_7_days['date'].apply(lambda d: d.strftime('%d.%m'))
actual_values = last_7_days['value'].values

predicted_values = rf_model.predict(last_7_days[['weekday', 'day']].values)
```

Рисунок 4.11 – Навчання моделі Random Forest та отримання прогнозів

За допомогою бібліотеки matplotlib, яка інтегрована безпосередньо у вікно customtkinter через клас FigureCanvasTkAgg, система будує порівняльний графік. Зеленою лінією позначається фактична активність користувача, а помаранчевою пунктирною – очікуваний прогноз моделі. Для оптимізації пам’яті комп’ютера розроблено механізм своєчасного закриття об’єктів графіків plt.close('all'), що запобігає виникненню попереджень про переповнення RuntimeWarning.

Крім графічного відображення, система розраховує відсоткове відхилення факту від плану та генерує персоналізований текстовий висновок (наприклад, повідомляє про «Відмінну стабільність» у разі дотримання індивідуального графіка). На рис. 4.12 показано аналітичну сторінку застосунку.

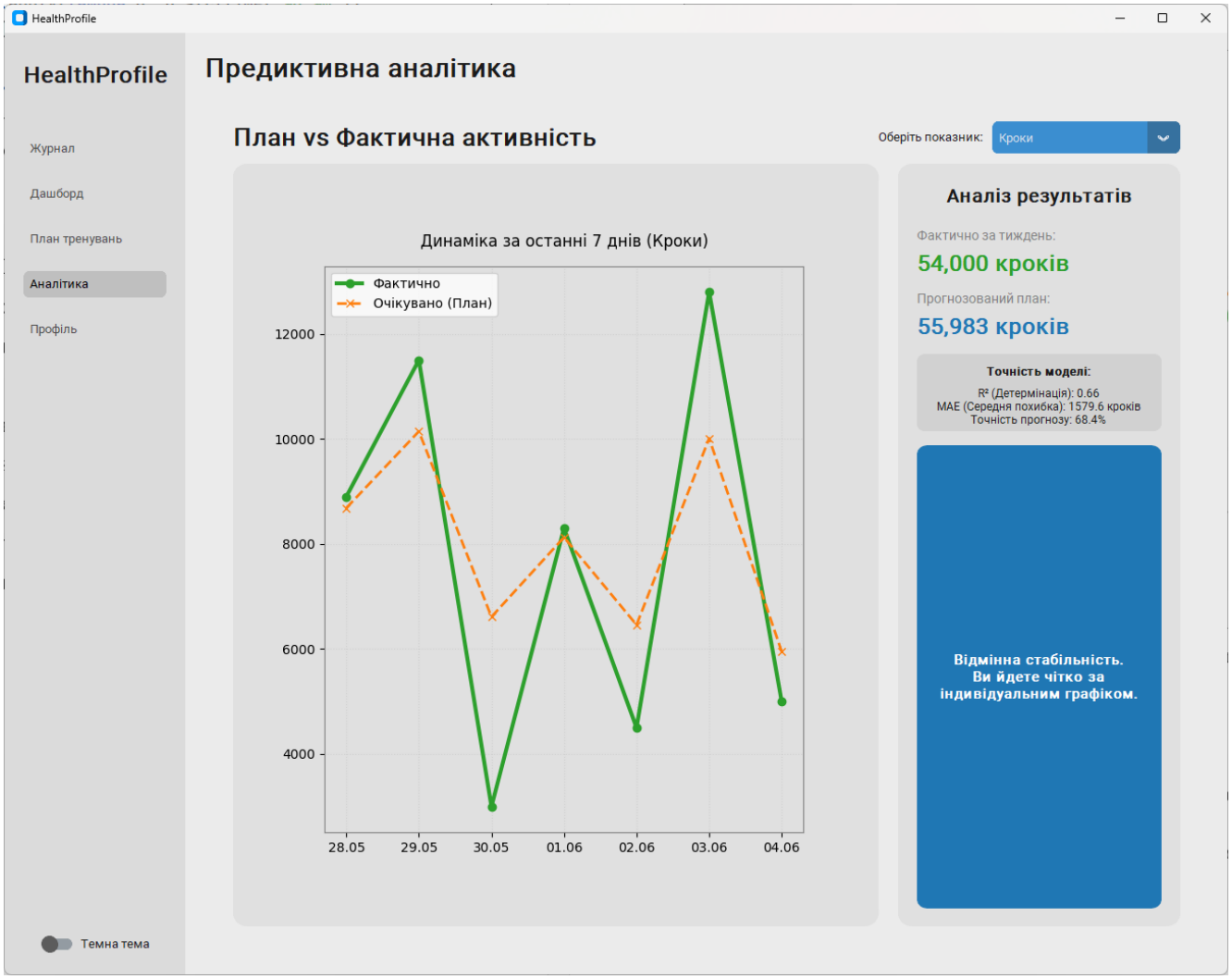

Рисунок 4.12 – Візуалізація роботи алгоритму Random Forest та порівняльний аналіз активності

4.5 Оцінка точності предиктивної моделі машинного навчання

Після практичної реалізації моделі машинного навчання RandomForestRegressor, важливим етапом стала розробка графічного інтерфейсу для візуалізації прогнозів та оцінки їхньої якості. Для цього у застосунку було створено окремий модуль предиктивної аналітики (prediction_frame.py), який дозволяє користувачу порівнювати фактичні показники активності з очікуваними (спрогнозованими) за останні 7 днів.

Окрім візуального порівняння, ключовою вимогою до системи є надання кількісної оцінки надійності моделі. В інтерфейсі застосунку реалізовано динамічний розрахунок математичних метрик точності, теоретичне обґрунтування та формули яких були наведені у підрозділі 2.3. При зміні досліджуваного показника система автоматично перераховує похибку моделі та виводить її у спеціальному інформаційному блоці. На рис. 4.13 показано графік динаміки та точність моделі відносно показника «Кроки».

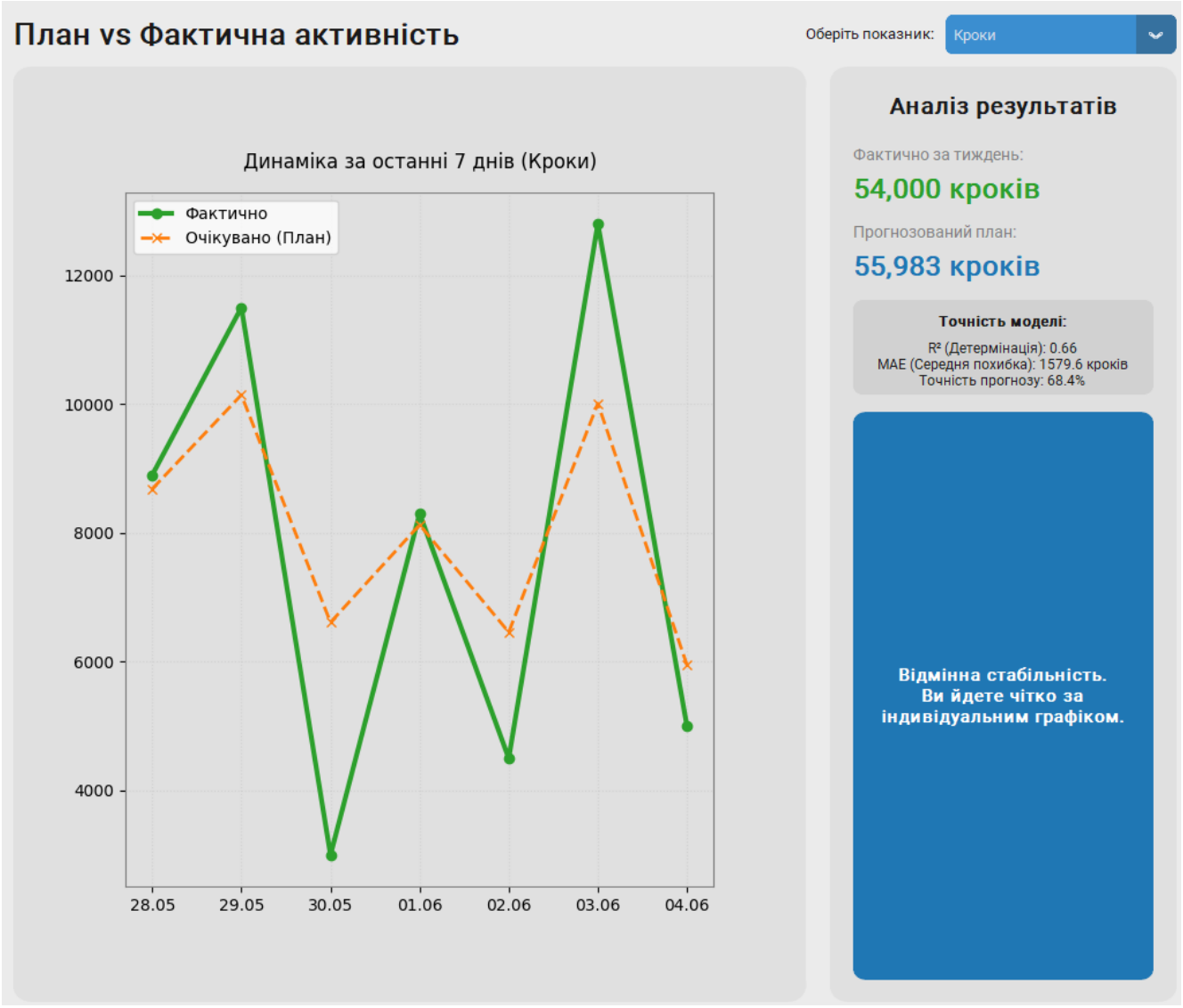

Рисунок 4.13 – Точність моделі відносно показника «Кроки»

На рис. 4.14 представлено графік динаміки та точність моделі відносно показника «Калорії».

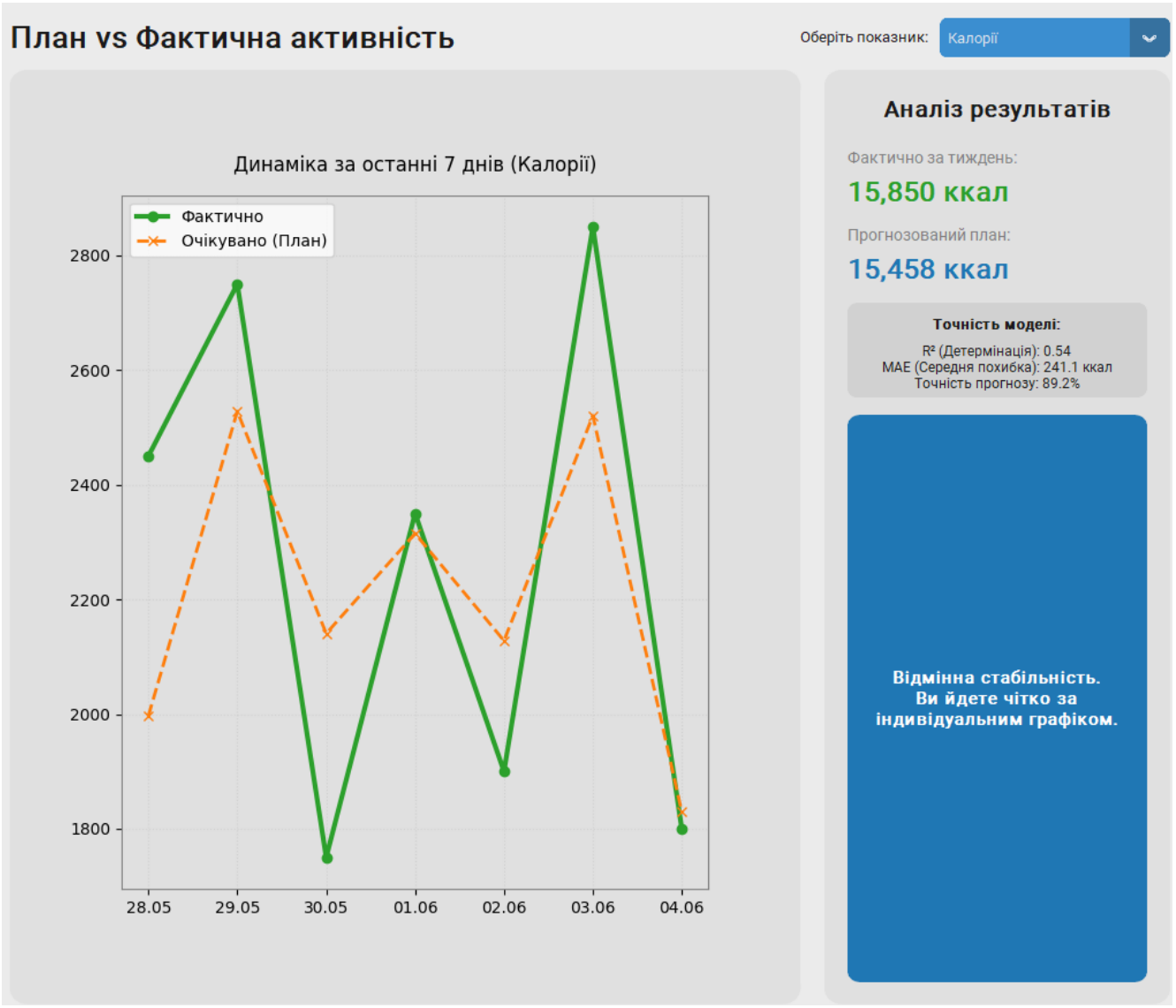

Рисунок 4.14 – Точність моделі відносно показника «Калорії»

На рис. 4.15 представлено графік динаміки та точність моделі відносно показника «Активні хвилини».

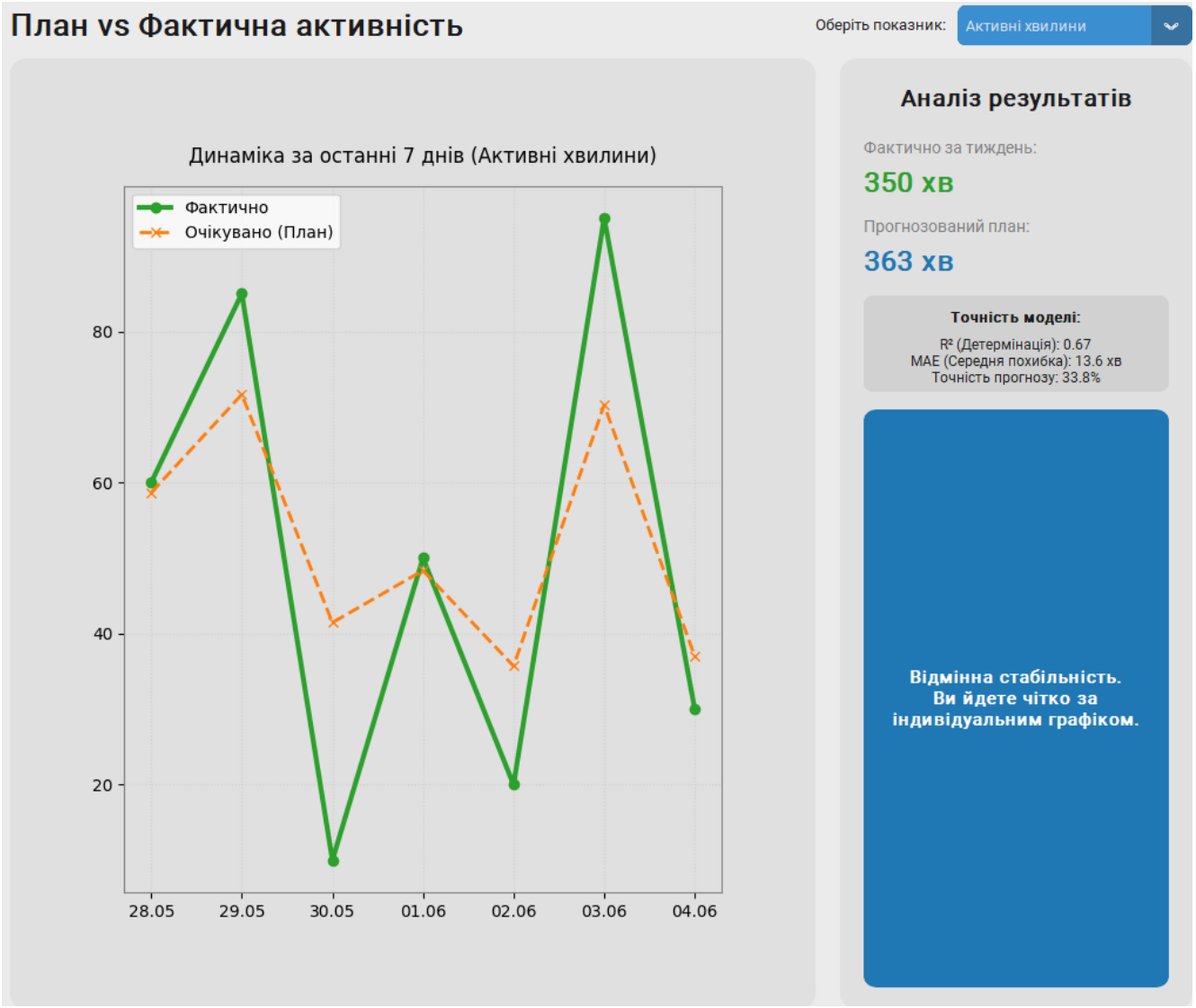

Рисунок 4.15 – Точність моделі відносно показника «Активні хвилини»

На рис. 4.16 зображено графік динаміки та точність моделі відносно показника «Сон».

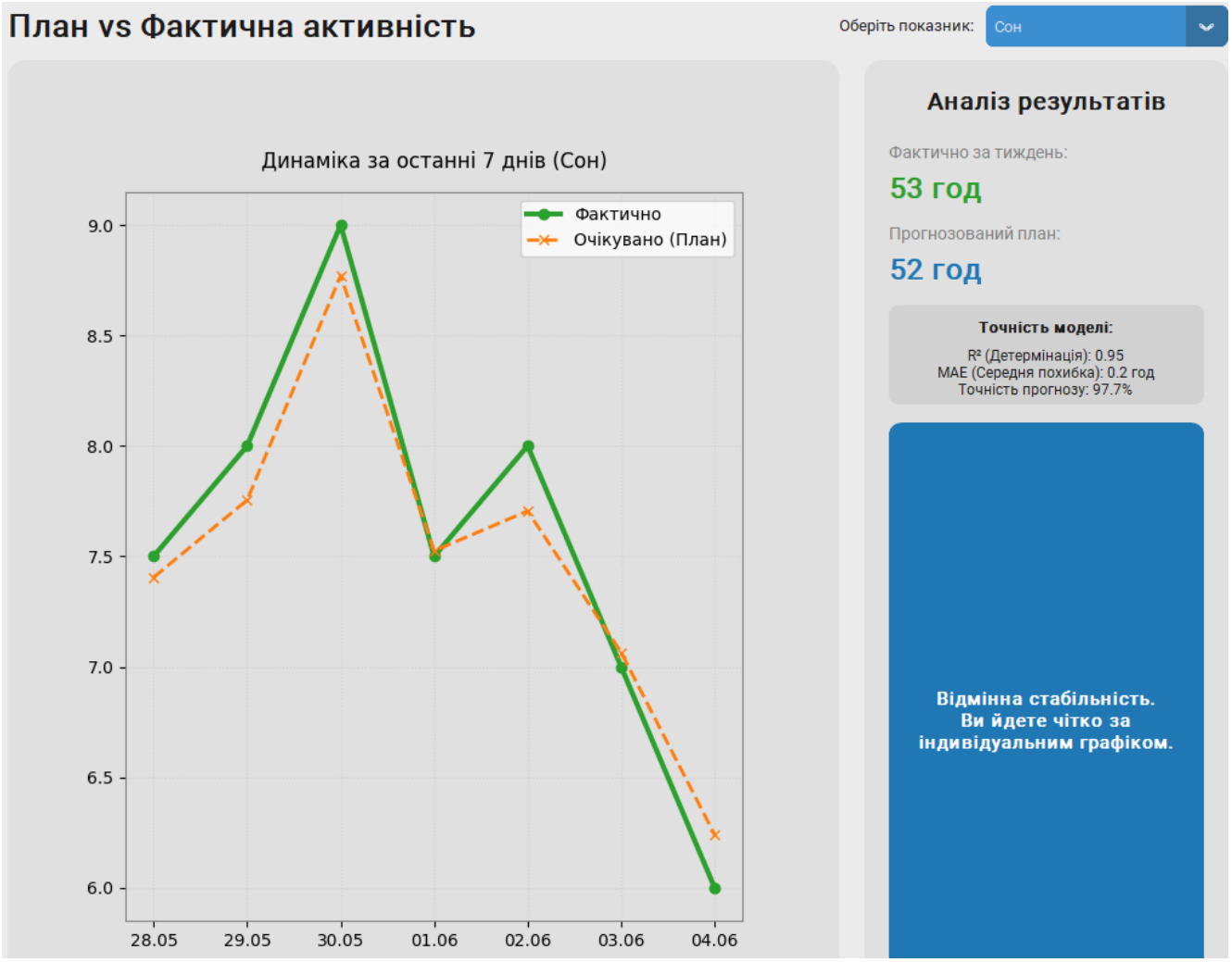

Рисунок 4.16 – Точність моделі відносно показника «Сон»

На рис. 4.17 зображено графік динаміки та точність моделі відносно показника «Активні хвилини».

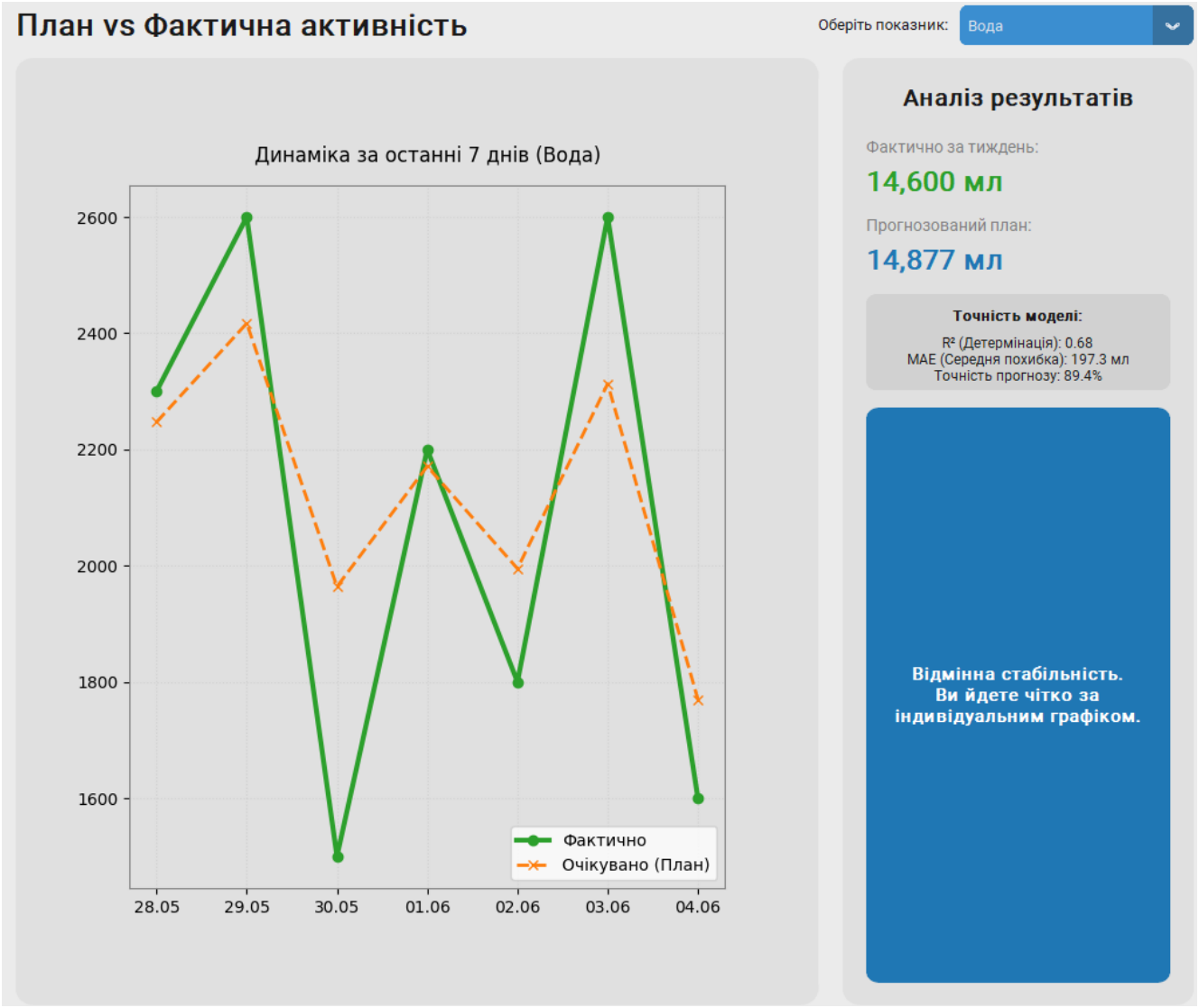

Рисунок 4.17 – Точність моделі відносно показника «Вода»

Тестування програми показало, що ефективність алгоритму Random Forest безпосередньо залежить від дисперсії вхідних даних.

Аналіз отриманих результатів виявив наступні закономірності роботи моделі:

Висока точність (Точність понад 90%, $R^2 > 0.9$). Спостерігається на фізіологічно стабільних показниках, таких як «Сон». Оскільки користувач дотримується відносно стабільного режиму, модель знаходить чіткі закономірності та робить максимально точні прогнози (MAE становить лише 0.2 год).

Середня точність (R^2 в межах 0.5 – 0.7). Притаманна високо мінливими показникам, таким як «Кроки» або «Калорії». Ці дані залежать від хаотичних зовнішніх факторів (погода, вихідні, незаплановані прогулянки), що спричиняє різкі коливання графіка. Для таких даних наявна похибка (зниження R^2) є математично очікуваною та допустимою.

Впровадження даного модуля дозволяє прозоро оцінити, наскільки надійними є згенеровані індивідуальні прогнози.

4.6 Тестування та перевірка працездатності системи

На завершальному етапі розробки було проведено функціональне тестування програмного продукту, зокрема методами позитивного та негативного тестування.

Захист від некоректного вводу. У формах введення даних реалізовано жорстку типізацію та валідацію. У разі введення строкових значень замість числових, або при спробі збереження від'ємних чи нереалістичних показників, система перехоплює виняток і виводить червоне інформаційне повідомлення без аварійного завершення програми.

На рис. 4.18 зображено повідомлення про попередження некоректно введеної кількості кроків.

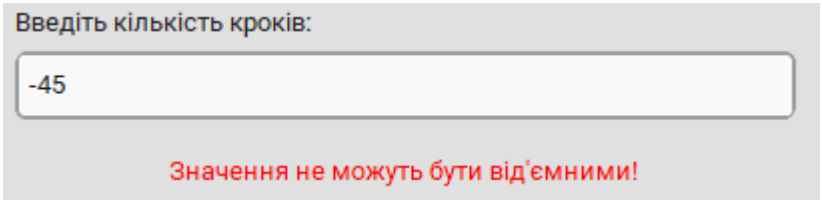

Рисунок 4.18 – Повідомлення про некоректність введення кількості кроків

Обробка крайових випадків у ML-модулях. Машинне навчання вимагає достатньої кількості даних для виявлення закономірностей. Якщо новий користувач завантажує модулі «План тренувань» або «Аналітика», не маючи мінімум 7 записів у базі даних, система програмно блокує виконання алгоритмів K-Means та Random Forest і виводить користувачеві повідомлення з проханням

зібрати більше статистики. Це запобігає виникненню системних помилок через недостатню розмірність матриці ознак. На рис. 4.19 зображено прохання про додавання записів для коректної роботи K-Means.

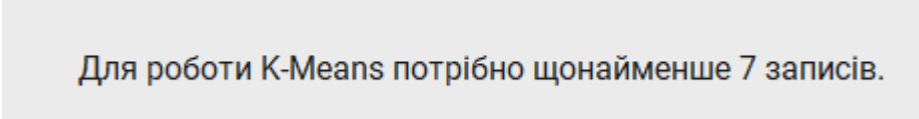

Рисунок 4.19 – Прохання про додавання більше записів для коректної роботи K-Means

На рис. 4.20 зображено прохання про додавання записів для коректної роботи Random Forest.

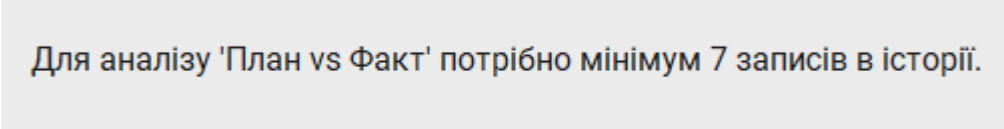

Рисунок 4.20 - Прохання про додавання більше записів для коректної роботи Random Forest

Проведене тестування доводить, що система є стабільною, алгоритми машинного навчання коректно обробляють введені дані, а інтерфейс користувача інтуїтивно зрозумілий та захищений від типових помилок введення.

Висновки до розділу 4

У четвертому розділі було описано процес практичної реалізації та тестування десктопного застосунку «HealthProfile» мовою Python. Побудовано архітектуру проєкту, налаштовано базу даних SQLite за допомогою технології SQLAlchemy та створено зручний графічний інтерфейс користувача з використанням бібліотеки customtkinter.

Основну увагу було приділено інтеграції модулів машинного навчання та оцінці їхньої ефективності. Алгоритм K-Means успішно застосовано для

сегментації історичних даних користувача та автоматичної генерації персоналізованих планів тренувань. Для алгоритму RandomForestRegressor реалізовано модуль предиктивної аналітики, який не лише візуалізує динаміку показників, але й забезпечує динамічний розрахунок метрик точності (R^2, середня абсолютна похибка MAE, точність у відсотках).

Проведена оцінка предиктивної моделі довела її адекватність: алгоритм забезпечує високу точність прогнозування (понад 90%) для стабільних фізіологічних показників (наприклад, тривалість сну), тоді як для високо мінливих даних (кроки, витрачені калорії) точність знаходиться на математично очікуваному середньому рівні, що пояснюється впливом хаотичних зовнішніх факторів.

Проведене функціональне тестування підтвердило надійність та відмовостійкість програмного продукту. У системі реалізовано багаторівневий захист від некоректного введення даних (перехоплення строкових, від’ємних чи нереалістичних значень) та програмну обробку крайових випадків для ML-модулів (зокрема, блокування роботи алгоритмів K-Means та Random Forest за умови відсутності мінімально необхідних 7 записів у базі даних).

Розроблена система функціонує стабільно, не допускає аварійних завершень через помилки користувача, повністю відповідає поставленим завданням дипломної роботи і є готовою до практичного використання з метою трекінгу та персоналізації фізичної активності.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було успішно вирішено актуальну науково-практичну задачу – розроблено програмну систему персоналізації фізичної активності користувача на основі алгоритмів машинного навчання.

На етапі системного аналізу предметної області було досліджено сучасний ринок фітнес-застосунків. Встановлено, що більшість існуючих рішень (наприклад, Google Fit, Apple Health) виконують здебільшого функцію пасивного збору даних (трекінгу) і не надають користувачам глибокої інтелектуальної аналітики їхніх результатів. Це обґрунтувало необхідність створення інформаційної системи, яка здатна не лише зберігати дані, але й адаптувати тренувальні плани під індивідуальні особливості та історію активності людини.

У процесі теоретичного дослідження було обрано та математично обґрунтовано використання двох алгоритмів машинного навчання: алгоритму кластеризації K-Means та алгоритму ансамблевого навчання Random Forest (Випадковий ліс). Комбінація цих методів дозволила вирішити одразу дві задачі: стратегічне планування (віднесення користувача до певної групи активності) та тактичне прогнозування (розрахунок очікуваних показників на кожен день).

Під час проєктування системи розроблено UML-діаграму прецедентів та класів, які чітко визначили межі системи, логіку взаємодії користувача з графічним інтерфейсом та архітектуру бази даних. Дані користувачів структуровано у реляційній базі даних SQLite із забезпеченням нормалізації таблиць. Взаємодія з базою даних реалізована через технологію ORM SQLAlchemy, що дозволило моделювати таблиці як класи та гарантувати каскадну цілісність інформації.

Практична реалізація інформаційної системи «HealthProfile» виконана мовою програмування Python. Застосовано бібліотеку customtkinter для створення сучасного графічного інтерфейсу користувача (GUI), що відповідає стандартам Material Design. Система містить модуль збору та відображення даних («Журнал» та «Дашборд»), що дозволяє вести облік кроків, калорій, сну та водного балансу з використанням інтерактивного теплового календаря. Модуль персонального

планування, який за допомогою алгоритму K-Means аналізує тижневу активність, визначає центри кластерів та генерує індивідуальні рекомендації (базовий день, інтенсивне тренування або день відновлення). Модуль предиктивної аналітики, де натренована модель RandomForestRegressor порівнює фактичну поточну активність із спрогнозованим планом, візуалізуючи результати у вигляді лінійних графіків за допомогою бібліотеки matplotlib та надаючи текстовий фідбек.

Ключовим досягненням предиктивного модуля є впровадження динамічного розрахунку математичних метрик точності (коефіцієнт детермінації R^2, середня абсолютна похибка MAE, відсоток точності). Оцінка результатів довела, що модель забезпечує високу точність прогнозування (понад 90%) для стабільних фізіологічних показників (сон), та математично очікуваний середній рівень точності для високо мінливих даних (кроки, калорії).

Проведене функціональне тестування підтвердило стабільність та відмовостійкість розробленої системи. Механізми валідації успішно блокують спроби введення некоректних або нереалістичних даних користувачем. Реалізовано програмну обробку крайових випадків: система блокує виконання алгоритмів K-Means та Random Forest за умови наявності менше 7 записів активності, чим захищає математичні моделі від збоїв через недостатній обсяг вибірки.

Отримані результати дозволяють стверджувати, що мета роботи досягнута повною мірою. Розроблена система може бути використана як персональний асистент для людей, що прагнуть оптимізувати свої фізичні навантаження, покращити розуміння власного тіла та підвищити ефективність тренувань завдяки впровадженню технологій штучного інтелекту. Подальший розвиток проєкту може полягати в інтеграції системи з носимними пристроями (смарт-годинниками) для автоматичного збору метрик через API та реалізації мобільної версії застосунку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Strava: Run, Bike, Hike [Електронний ресурс]. URL: https://apps.apple.com/us/app/strava-run-bike-hike/id426826309 (Last accessed: 24.05.2026).
2. Google Fit: Activity Tracking [Електронний ресурс]. URL: https://play.google.com/store/apps/details?id=com.google.android.apps.fitness (Last accessed: 24.05.2026).
3. Apple Health App [Електронний ресурс]. URL: https://www.apple.com/health (Last accessed: 24.05.2026).
4. Scikit-learn: Machine Learning in Python. K-Means Clustering [Електронний ресурс]. URL: https://scikit-learn.org/stable/modules/clustering.html#k-means (Last accessed: 24.05.2026).
5. Random Forest Regression in Python [Електронний ресурс]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html (Last accessed: 24.05.2026).
6. Python 3.10.0 Documentation [Електронний ресурс]. URL: https://docs.python.org/3.10 (Last accessed: 24.05.2026).
7. Scikit-learn: Official Documentation [Електронний ресурс]. URL: https://scikit-learn.org/stable (Last accessed: 24.05.2026).
8. Scikit-learn [Електронний ресурс]. URL: https://fahadsultan.com/csc272/beyond/sklearn.html (Last accessed: 24.05.2026).
9. SQLite Official Documentation [Електронний ресурс]. URL: https://www.sqlite.org/docs.html (Last accessed: 24.05.2026).
10. SQLAlchemy 2.0 Documentation [Електронний ресурс]. URL: https://docs.sqlalchemy.org/en/20 (Last accessed: 24.05.2026).
11. CustomTkinter UI-Library [Електронний ресурс]. URL: https://customtkinter.tomschimansky.com (Last accessed: 24.05.2026).

12. M. D. Lytras and A. C. Şerban, "E-Government Insights to Smart Cities Research: European Union (EU) Study and the Role of Regulations," in *IEEE Access*, vol. 8, pp. 65313-65326, 2020, doi: 10.1109/ACCESS.2020.2982737.

13. K-means Clustering: Algorithm, Applications, Evaluation Methods [Електронний ресурс]. URL: https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1 (Last accessed: 24.05.2026).

14. Understanding the Mathematics behind K-Means Clustering [Електронний ресурс]. URL: https://www.geeksforgeeks.org/k-means-clustering-introduction (Last accessed: 24.05.2026).

15. Lloyd's algorithm for k-means clustering [Електронний ресурс]. URL: https://en.wikipedia.org/wiki/Lloyd%27s_algorithm (Last accessed: 24.05.2026).

16. k-means++: The Advantages of Careful Seeding [Електронний ресурс]. URL: https://en.wikipedia.org/wiki/K-means%2B%2B (Last accessed: 24.05.2026).

17. Clustering Algorithms: A Comprehensive Guide [Електронний ресурс]. URL: https://developers.google.com/machine-learning/clustering (Last accessed: 24.05.2026).

18. K-Means Clustering [Електронний ресурс]. URL: https://github.com/DanBlocks/K-Means_Clustering/blob/master/README.md (Last accessed: 24.05.2026).

19. Elbow Method for optimal value of k in KMeans [Електронний ресурс]. URL: https://www.geeksforgeeks.org/machine-learning/elbow-method-for-optimal-value-of-k-in-kmeans (Last accessed: 24.05.2026).

20. The Elbow Method in K-Means Clustering [Електронний ресурс]. URL: https://www.scikit-yb.org/en/latest/api/cluster/elbow.html (Last accessed: 24.05.2026).

21. Regression Algorithms in Machine Learning [Електронний ресурс]. URL: https://www.ibm.com/topics/linear-regression (Last accessed: 24.05.2026).

22. What Is Random Forest? [Електронний ресурс]. URL: https://www.ibm.com/think/topics/random-forest (Last accessed: 24.05.2026).

23. What are the Advantages and Disadvantages of Random Forest? [Електронний ресурс]. URL: https://www.geeksforgeeks.org/machine-learning/what-are-the-advantages-and-disadvantages-of-random-forest (Last accessed: 24.05.2026).

24. Bootstrap Aggregating (Bagging) in Machine Learning [Електронний ресурс]. URL: https://en.wikipedia.org/wiki/Bootstrap_aggregating (Last accessed: 24.05.2026).

25. Decision Trees for Regression [Електронний ресурс]. URL: https://scikit-learn.org/stable/modules/tree.html#regression (Last accessed: 24.05.2026).

26. Mean Squared Error (MSE) Evaluation Metric [Електронний ресурс]. URL: https://scikit-learn.org/stable/modules/model_evaluation.html#mean-squared-error (Last accessed: 24.05.2026).

27. Ensemble Methods in scikit-learn [Електронний ресурс]. URL: https://scikit-learn.org/stable/modules/ensemble.html (Last accessed: 24.05.2026).

28. Random Forests in Machine Learning: What They Are and How They Work [Електронний ресурс]. URL: https://www.grammarly.com/blog/ai/what-is-random-forest (Last accessed: 24.05.2026).

29. Variance Reduction in Random Forests [Електронний ресурс]. URL: https://machinelearningmastery.com/bagging-and-random-forest-for-imbalanced-classification (Last accessed: 24.05.2026).

30. Data Preparation for Machine Learning [Електронний ресурс]. URL: https://www.actian.com/data-preparation-for-machine-learning (Last accessed: 24.05.2026).

31. FitBit Fitness Tracker Data (Kaggle Dataset) [Електронний ресурс]. URL: https://www.kaggle.com/datasets/arashnic/fitbit (Last accessed: 24.05.2026).

32. Data Collection Privacy for Fitness Trackers [Електронний ресурс]. URL: https://www.fitbit.com/global/us/legal/privacy-policy (Last accessed: 24.05.2026).

33. Pandas DataFrame Documentation [Електронний ресурс]. URL: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html (Last accessed: 24.05.2026).

34. Data Preprocessing for Machine Learning in Python [Електронний ресурс]. URL: https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python (Last accessed: 24.05.2026).

35. Time Series Data Analysis using Pandas [Електронний ресурс]. URL: https://pandas.pydata.org/docs/user_guide/timeseries.html (Last accessed: 24.05.2026).

36. StandardScaler (Z-score Normalization) in scikit-learn [Електронний ресурс]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html (Last accessed: 24.05.2026).

37. Pearson Correlation Coefficient [Електронний ресурс]. URL: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient (Last accessed: 24.05.2026).

ДОДАТОК А

Лістинг фрагментів коду

Clustering.py

```
import os
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from database.db_config import SessionLocal
from database.models import User, DailyActivity, ClusterProfile


def update_user_cluster(user_id):
    session = SessionLocal()

    activities = session.query(DailyActivity).filter_by(user_id=user_id).all()

    if not activities:
        session.close()
        return None, "Немає даних для аналізу. Введіть статистику."

    avg_steps = np.mean([act.steps for act in activities])
    avg_cal = np.mean([act.calories for act in activities])
    avg_min = np.mean([act.active_minutes for act in activities])

    dataset_path = os.path.join(os.getcwd(), 'data', 'fitbit_dataset.csv')

    if not os.path.exists(dataset_path):
        dataset_path = os.path.join(os.getcwd(), 'data',
'dailyActivity_merged.csv')

        if not os.path.exists(dataset_path):
            session.close()
            error_msg = f"Помилка: Файл не знайдено!\nПрограма шукала його
тут:\n{dataset_path}"
            return None, error_msg

    df = pd.read_csv(dataset_path)

    X_train = df[['TotalSteps', 'Calories', 'VeryActiveMinutes']].copy()

    X_train = X_train[(X_train['TotalSteps'] > 0) & (X_train['Calories'] > 0)]

    user_df = pd.DataFrame([[avg_steps, avg_cal, avg_min]], columns=['TotalSteps',
'Calories', 'VeryActiveMinutes'])
    X_train = pd.concat([X_train, user_df], ignore_index=True)

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X_train)

    kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
    clusters = kmeans.fit_predict(X_scaled)

    user_cluster_index = clusters[-1]

    cluster_centers = scaler.inverse_transform(kmeans.cluster_centers_)
```

```
    sorted_indices = np.argsort(cluster_centers[:, 0])

    mapping = {sorted_indices[0]: 0, sorted_indices[1]: 1, sorted_indices[2]: 2}

    final_db_cluster_id = mapping[user_cluster_index]

    user = session.query(User).filter_by(id=user_id).first()
    if user:
        user.cluster_id = int(final_db_cluster_id)
        session.commit()

    cluster_info =
session.query(ClusterProfile).filter_by(id=final_db_cluster_id).first()

    result = {
        "cluster_name": cluster_info.cluster_name,
        "description": cluster_info.description,
        "avg_steps": round(avg_steps),
        "avg_cal": round(avg_cal),
        "avg_min": round(avg_min),
        "dataset_size": len(X_train) - 1
    }

    session.close()
    return result, "Успіх"
```

Forecasting.py

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from database.db_config import SessionLocal
from database.models import DailyActivity, Prediction
import datetime


def generate_prediction(user_id):
    session = SessionLocal()

    activities =
session.query(DailyActivity).filter_by(user_id=user_id).order_by(DailyActivity.act
ivity_date).all()

    if len(activities) < 4:
        session.close()
        return None, "Для роботи предиктивної моделі потрібно мінімум 4 дні
історії тренувань."

    data = []
    for act in activities:
        data.append({
            'date': act.activity_date,
            'steps': act.steps,
            'calories': act.calories,
            'active_minutes': act.active_minutes,
            'sleep_hours': getattr(act, 'sleep_hours', 7.0),
            'water_ml': getattr(act, 'water_ml', 2000)
        })

    df = pd.DataFrame(data)
```

```
    df['prev_steps'] = df['steps'].shift(1)
    df['prev_calories'] = df['calories'].shift(1)
    df['prev_minutes'] = df['active_minutes'].shift(1)
    df['prev_sleep'] = df['sleep_hours'].shift(1)
    df['prev_water'] = df['water_ml'].shift(1)

    df_train = df.dropna().copy()

    features = ['prev_steps', 'prev_calories', 'prev_minutes', 'prev_sleep',
'prev_water']
    X = df_train[features]
    y = df_train['steps']

    model = RandomForestRegressor(n_estimators=100, max_depth=5, random_state=42)
    model.fit(X, y)

    last_day = df.iloc[-1]
    X_predict = pd.DataFrame([[
        last_day['steps'],
        last_day['calories'],
        last_day['active_minutes'],
        last_day['sleep_hours'],
        last_day['water_ml']
    ]], columns=features)

    predicted_steps = int(model.predict(X_predict)[0])

    tomorrow = datetime.date.today() + datetime.timedelta(days=1)

    existing_pred = session.query(Prediction).filter_by(user_id=user_id,
target_date=tomorrow).first()
    if existing_pred:
        existing_pred.predicted_steps = predicted_steps
    else:
        new_pred = Prediction(user_id=user_id, target_date=tomorrow,
predicted_steps=predicted_steps)
        session.add(new_pred)
    session.commit()
    session.close()

    advice = ""
    if last_day['sleep_hours'] < 6.5:
        advice += "Вчора ви спали мало. Модель врахувала вашу втому і знизила
очікувану активність. Головна рекомендація: відновіть режим сну. "
    elif last_day['water_ml'] < 1500:
        advice += "Вчора ви пили мало води, що негативно впливає на витривалість.
Спробуйте пити більше. "

    if predicted_steps > last_day['steps']:
        advice += "Загалом, модель очікує збільшення вашої активності завтра.
Підготуйтеся до інтенсивнішого дня!"
    else:
        advice += "Очікується невеликий спад або стабілізація активності. Це
нормально для циклічних навантажень."

    df['date'] = pd.to_datetime(df['date'])
    history_dates = df['date'].tail(7).dt.strftime('%m-%d').tolist()
    history_steps = df['steps'].tail(7).tolist()

    history_dates.append("Завтра")
    history_steps.append(predicted_steps)
```

```
result = {
    "target_date": tomorrow.strftime("%Y-%m-%d"),
    "predicted_steps": predicted_steps,
    "last_steps": int(last_day['steps']),
    "advice": advice,
    "graph_dates": history_dates,
    "graph_steps": history_steps,
    "feature_importances": model.feature_importances_.tolist(),
    "feature_names": ["Кроки", "Ккал", "Хв.", "Сон", "Вода"]
}

return result, "Успіх"
```