Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

ПРОГРАМНИЙ МОДУЛЬ ІНДИВІДУАЛЬНОГО ПЛАНУВАННЯ СПОРТИВНИХ ТРЕНУВАНЬ

Виконав: студент 4 курсу, групи 2КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Олександр ВЛАСОК
(прізвище та ініціали)

Керівник: д.т.н., проф. каф. КН
Ярослав ІВАНЧУК
(прізвище та ініціали)
«04» червня 2025 р.

Рецензент: д.т.н., проф., зав. каф. КСУ
Вячеслав КОВТУН
(прізвище та ініціали)
«05» червня 2025 р.

Допущено до захисту
Зав. кафедри проф., д.т.н. Андрій ЯРОВИЙ
«06» червня 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо - професійна програма – Комп'ютерні науки

ЗАТВЕРДЖЕНО
Завідувач кафедри КН
проф., д.т.н. Андрій ЯРОВИЙ
«05» травня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Власку Олександру Михайловичу

1. Тема роботи: Програмний модуль індивідуального планування спортивних тренувань.
керівник роботи: Іванчук Ярослав Володимирович, д.т.н, професор кафедри КН
затверджені наказом вищого навчального закладу "20" 03 2025 року №97
2. Термін подання студентом роботи 30 травня 2025
3. Вихідні дані до роботи : об'єктно-документний підхід із серіалізацією даних у форматі JSON; кількість даних про користувача - 20, кількість фізичних вправ - 15, тренувальні плани та результати; інтерфейс програмного застосунку — графічний користувацький інтерфейс.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): визначення об'єкту, предмета, мети та задач подальшого дослідження; аналіз предметної області, існуючих методів, алгоритмів та інструментів для створення програмного модуля індивідуального планування спортивних тренувань; проєктування програмного модуля індивідуального планування спортивних тренувань; програмна реалізація та тестування індивідуального планування спортивних тренувань.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): загальна архітектура програмного модуля індивідуального планування спортивних тренувань; UML-діаграма класів системи індивідуального планування спортивних тренувань; алгоритми роботи основних модулів системи (управління користувачами, оцінка фізичного стану, генерація тренувальних програм, відстеження прогресу); об'єктно-орієнтована модель даних предметної області "Індивідуальне планування спортивних тренувань"; Зовнішній вигляд роботи програмного модулю.

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Ярослав ІВАНЧУК, д.т.н., проф.. каф. КН		

7. Дата видачі завдання 05 травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапів	Термін виконання		Примітка
		початок	кінець	
1	Визначення об'єкта, предмета, мети та задачі подальшого дослідження	06.05.25	08.05.25	
2	Аналіз предметної області, існуючих методів, алгоритмів та інструментів для створення програмного модуля індивідуального планування спортивних тренувань	09.05.25	17.05.25	
3	Проектування програмних модулів для індивідуального планування спортивних тренувань	18.05.25	23.05.25	
4	Програмна реалізація програмного модуля індивідуального планування спортивних тренувань	24.05.25	01.06.25	
5	Оформлення пояснювальної записки	02.06.25	04.06.25	

Студент _____ Олександр ВЛАСОК
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ярослав ІВАНЧУК
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 123 сторінок формату А4, на яких є 34 рисунка, список використаних джерел містить 35 найменувань.

Дана бакалаврська робота присвячена розробці програмного модуля індивідуального планування спортивних тренувань. Основною метою роботи є підвищення рівня ефективності тренувального процесу та покращення фізичних результатів шляхом впровадження індивідуального підходу до планування спортивних тренувань за допомогою спеціалізованого програмного забезпечення.

Була розроблена трирівнева архітектура програмного модуля, що включає презентаційний рівень, рівень бізнес-логіки та рівень даних. Створено UML-діаграму класів системи, схеми алгоритмів роботи основних модулів (управління користувачами, оцінка фізичного стану, генерація тренувальних програм, відстеження прогресу) та об'єктно-орієнтовану модель даних предметної області.

Python та PyQt5 використовуються для розробки програмного модуля та забезпечення зручної взаємодії з користувачем через графічний інтерфейс. Для зберігання даних застосовано об'єктно-документний підхід із серіалізацією у форматі JSON, що забезпечує гнучкість схеми даних та спрощує розробку без залежності від зовнішніх СУБД.

Результатом роботи є функціональний та надійний програмний модуль, який дозволяє користувачам ефективно планувати тренувальний процес, автоматично генерувати індивідуальні програми тренувань, відстежувати прогрес та отримувати персоналізовані рекомендації з харчування. Проведене тестування підтвердило коректність роботи всіх компонентів системи та готовність до практичного використання.

Ключові слова: індивідуальне планування спортивних тренувань, програмний модуль, фізична підготовка, автоматизація процесу, спорт.

ABSTRACT

The Bachelor's thesis consists of 123 A4 pages with 34 figures, the list of references includes 35 titles.

This bachelor's thesis is devoted to the development of a software module for individual sports training planning. The main objective of the work is to increase the efficiency of the training process and improve physical performance by introducing an individual approach to sports training planning using specialised software.

A three-level architecture of the software module was developed, including the presentation level, the business logic level and the data level. A UML diagram of the system classes, algorithm diagrams for the main modules (user management, physical condition assessment, training programme generation, progress tracking) and an object-oriented data model of the subject area were created.

Python and PyQt5 are used to develop the software module and provide convenient user interaction through a graphical interface. An object-document approach with serialisation in JSON format is used for data storage, which provides flexibility of the data schema and simplifies development without dependence on external DBMSs.

The result is a functional and reliable software module that allows users to effectively plan the training process, automatically generate individual training programmes, track progress, and receive personalised nutritional advice. The conducted testing confirmed the correctness of all system components and their readiness for practical use.

Keywords: individual planning of sports training, software module, physical training, process automation, sport.

ЗМІСТ

ВСТУП

Актуальність дослідження. В сучасному світі люди все більше починають приділяти уваги здоровому способу життя та фізичній активності. Індивідуальне планування спортивних тренувань стає все більш затребуваним, оскільки дозволяє досягти максимальної ефективності, мінімізувати ризики травм та забезпечити стабільний прогрес у фізичному розвитку. З розширенням доступності інформації та технологій, тема розробки спеціалізованих програмних засобів для оптимізації тренувального процесу набуває особливої актуальності [1].

Однією з головних проблем, яку вирішить програмний модуль, є складність самостійного створення тренувальних планів. Більшість людей не мають достатніх знань у сфері тренувальних методик для розробки оптимальної програми тренувань. Крім того, традиційні підходи до планування тренувань часто не враховують індивідуальні особливості людини: фізіологічні показники, рівень підготовки, тренувальні цілі та доступні ресурси комплексно, що призводить до субоптимальних результатів та знижує мотивацію займатися спортом.

Програмний модуль допоможе оптимізувати тренувальний процес за рахунок аналізу індивідуальних параметрів користувача та автоматичної генерації персоналізованих програм тренувань. Це дозволить користувачам ефективніше управляти своїм фізичним розвитком та досягати кращих результатів при менших витратах часу.

Метою дослідження є підвищення ефективності тренувального процесу та покращення фізичних результатів шляхом впровадження індивідуального підходу до планування спортивних тренувань за допомогою спеціалізованого програмного забезпечення.

Предметом дослідження є програмний модуль індивідуального планування спортивних тренувань, його функціональні можливості та інтеграція з користувацьким інтерфейсом.

Об'єктом дослідження є процес створення індивідуального плану спортивних тренувань, що здійснюється за допомогою програмного модуля.

Задачі дослідження:

1. Провести аналіз предметної області, існуючих методів та алгоритмів індивідуального планування спортивних тренувань.

2. Проєктування архітектури програмного модуля індивідуального планування спортивних тренувань та взаємодії програмних модулів системи.

3. Розробка бази даних програмного модуля індивідуального планування спортивних тренувань з використанням об'єктно-документного підходу.

4. Розробка алгоритму функціонування програмного модуля індивідуального планування спортивних тренувань та генерації персоналізованих програм.

5. Програмна реалізація модуля індивідуального планування спортивних тренувань за допомогою мови програмування Python.

6. Тестування програмного модуля індивідуального планування спортивних тренувань та перевірка його функціональності.

7. Розробка інструкції користувача програмного модулю індивідуального планування спортивних тренувань

Апробація роботи

Робота апробувалась на LIV Всеукраїнській науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації (2025).

Публікації: електронні тези на тему «Програмний модуль індивідуального планування спортивних тренувань» [2].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз ключових аспектів індивідуального планування спортивних тренувань

Індивідуальне планування спортивних тренувань являється ключовим фактором у забезпеченні ефективного фізичного розвитку, реабілітації та спортивного вдосконалення людини. Сьогодні, коли питання здоров'я, фізичної форми та активного способу життя стають все більш популярними в житті кожної людини, створення індивідуальних тренувальних програм стає необхідністю для досягнення оптимальних результатів та запобігання травм.

В основі індивідуального планування тренувань лежать чотири ключових принципа. Принцип індивідуалізації враховує унікальні характеристики кожної людини: вік, стать, рівень фізичної підготовки, стан здоров'я та генетичні фактори. Дослідження у сфері спортивної фізіології показують, що індивідуальна реакція на тренувальні навантаження може суттєво відрізнятися через генетичні фактори, включаючи тип м'язових волокон [3-6]. Люди з переважанням швидких м'язових волокон (тип II) зазвичай краще реагують на силові тренування високої інтенсивності, тоді як люди з переважанням повільних волокон (тип I) демонструють кращі результати при тренуваннях на витривалість [3-6].

Принцип поступовості передбачає систематичне та поступове збільшення тренувального навантаження. Різкі зміни інтенсивності чи обсягу тренувань можуть призвести до перевантаження або травм. На практиці тренувань використовуються різні стратегії прогресії: лінійна (рівномірне збільшення навантаження), хвилеподібна (чергування періодів підвищення та зниження навантаження) та ступінчаста (періоди стабільного навантаження з подальшим значним підвищенням) [3-6].

Принцип систематичності наголошує на важливості регулярності тренувань. Дослідження показують, що для досягнення стійких адаптаційних

змін в організмі, таких як збільшення м'язової маси, підвищення міцності зв'язок та сухожиль, покращення нейром'язової координації, необхідне регулярне тренувальне навантаження [3-6]. Оптимальна частота тренувань залежить від багатьох факторів, включаючи тип тренування, рівень підготовки та індивідуальні особливості відновлення.

Принцип варіативності забезпечує різноманітність тренувальних стимулів, що запобігає адаптивному плато – стану, коли організм звикає до однотипних навантажень і прогрес сповільнюється. Цей принцип реалізується через періодичну зміну вправ, інтенсивність, обсяг та інших параметрів тренувань [3-6]. Різноманітність тренувальних стимулів не лише запобігає фізичному та психологічному плато, але й знижує ризик травм від перевикористання окремих структур організму.

На процес індивідуального планування тренувань впливає ряд ключових факторів. Фізіологічні параметри, такі як вік, стать, антропометричні показники та склад тіла, відіграють важливу роль. З віком змінюються можливості організму, здатність до відновлення та адаптації до навантажень. Гормональні відмінності між чоловіками та жінками також впливають на швидкість розвитку різних фізичних якостей, особливо на здатність нарощувати м'язову масу та розподіл жирової тканини [7].

Рівень фізичної підготовки є важливим фактором, який визначає стартову точку та темп прогресування тренувань. Початківці потребують програми з акцентом на базові вправи, правильну техніку виконання та поступове введення в тренувальний процес. З розвитком програми можна ускладнювати, включаючи більше варіацій вправ, вищу інтенсивність та більший обсяг тренувань [8].

Тренувальні цілі також суттєво впливають на структуру та зміст тренувальної програми. Для схуднення ефективними є програми, що включають комбінацію кардіо та силових тренувань з акцентом на створення дефіциту калорій.

Для набору м'язової маси ключовими є силові тренування з достатнім навантаженням, обсягом та відповідним харчуванням. Для підвищення витривалості необхідні специфічні кардіо-тренування з прогресуючою інтенсивністю та тривалістю.

Важливо також враховувати медичні обмеження та попередні травми при плануванні тренувань. Програми тренувань для людей з хронічними захворюваннями або травмами потребують спеціальної адаптації та часто розробляються у співпраці з медичними фахівцями [9].

Не менш важливим фактором є доступність обладнання та умов для тренувань. Програми можуть суттєво відрізнятися залежно від того, чи тренується людина в повністю обладнаному тренажерному залі, вдома з мінімальним набором обладнання або на відкритому повітрі [10].

У сучасному тренувальному процесі значну роль відіграють різні методики періодизації – систематичного планування тренувань з розподілом на цикли різної тривалості. Лінійна періодизація передбачає поступове збільшення інтенсивності при зменшенні обсягу тренувань. Нелінійна (хвилеподібна) періодизація, яка набуває великої популярності, включає варіювання інтенсивності та обсягу навіть в межах одного мікроциклу [11].

Для різних видів тренувань характерні свої особливості планування. Силові тренування зазвичай структуруються з урахуванням залучених м'язових груп, часу відновлення та типу силових здібностей, що розвиваються (максимальна сила, гіпертрофія, витривалість). Кардіотренування плануються з урахуванням цільових пульсових зон, типу кардіонавантаження (безперервне, інтервальне) та загального обсягу [12].

Моніторинг і корекція тренувальних планів є невід'ємною частиною індивідуального підходу. Регулярна оцінка прогресу, збір суб'єктивних (самопочуття, сприйняття навантаження) та об'єктивних (показники сили, витривалості, складу тіла) даних дозволяють своєчасно адаптувати тренувальну програму. Сучасні технології, такі як фітнес-трекери, програми

для аналізу тренувань та спеціалізовані тести, значно полегшують цей процес [13].

Індивідуальне планування враховує також психологічні аспекти тренувань. Дослідження показують, що прихильність до тренувального плану, мотивація та психологічний стан значно впливають на результати. Програми, які враховують психологічні особливості особистості (тип темпераменту, вподобання, мотиваційні фактори), зазвичай демонструють вищу ефективність [14].

Узагальнюючи, індивідуальне планування спортивних тренувань є комплексним процесом, який вимагає врахування фізіологічних, психологічних, технічних та практичних аспектів. Застосування наукових принципів та методів у поєднанні з індивідуальним підходом дозволяє створювати ефективні тренувальні програми, які максимально відповідають потребам та можливостям конкретної людини.

1.2 Аналіз існуючих методів та алгоритмів для індивідуального планування тренувань

Для ефективного планування тренувань розроблено ряд методів та алгоритмів, які допомагають оптимізувати тренувальний процес, роблячи його більш науково обґрунтованим та ефективним.

Одним з найважливіших є метод визначення оптимального навантаження на основі концепції максимального повторення (1RM – one repetition maximum) – найбільшої ваги, яку людина може підняти один раз для конкретної вправи. Знаючи 1RM, можна розрахувати робочі ваги для різних цілей: для розвитку максимальної сили використовують 85-95% від 1RM, для гіпертрофії – 65-85%, для м'язової витривалості – 40-65% [15-16].

1RM можна визначити прямим методом (поступово збільшуючи вагу до досягнення максимуму) або непрямим (розрахунок за формулами на основі

виконання підходу з більшою кількістю повторень). Популярною є формула Бржицького: 1RM = вага × (1,0278 / (0,0278 - кількість повторень)) [15-16].

Цікавим підходом є метод суб'єктивної оцінки навантаження (RPE – Rating of Perceived Exertion), який враховує не лише об'єктивні показники, але й суб'єктивне сприйняття навантаження за шкалою від 1 до 10, де 10 – максимальне можливе зусилля [17]. Цей метод дозволяє адаптувати тренування відповідно до поточного стану людини, який може змінюватися під впливом різних факторів (якість сну, рівень стресу, загальна втома).

Для забезпечення постійного прогресу важливі алгоритми прогресії навантаження. Лінійна прогресія передбачає поступове збільшення ваги з фіксованим кроком, наприклад, на 2.5 кг щотижня [18]. Цей метод ефективний для початківців, але з часом втрачає ефективність.

Більш ефективною в довгостроковій перспективі є хвилеподібна прогресія, яка передбачає чергування періодів підвищення та зниження навантаження. Наприклад, три тижні поступового збільшення ваги, після чого тиждень з меншою вагою для активного відновлення. Цей метод дозволяє організму краще адаптуватися до навантажень, зменшує ризик перетренованості.

Ступінчаста прогресія характеризується періодами стабільного навантаження, після яких відбувається значне одноразове підвищення. Наприклад, протягом місяця виконується вправа з однаковою вагою, після чого вага збільшується на 10%. Цей метод часто використовується для подолання плато в тренуваннях [19].

Особливе місце займають методи періодизації, які дозволяють структурувати тренувальний процес на тривалий термін. Лінійна періодизація передбачає послідовну зміну фаз тренувань, зазвичай від високого обсягу/низької інтенсивності до низького обсягу/високої інтенсивності [19]. Наприклад, спочатку фаза гіпертрофії (8-12 повторень з 70-80% від 1RM), потім фаза сили (4-6 повторень з 80-90% від 1RM), і нарешті фаза піку (1-3 повторення з 90-100% від 1RM).

Нелінійна періодизація передбачає часту зміну тренувальних параметрів, навіть протягом одного тижня. Наприклад, понеділок – тренування на гіпертрофію, середа – на силу, п'ятниця – на м'язову витривалість. Цей підхід забезпечує різноманітність стимулів та запобігає адаптаційному плато [19].

Блокова періодизація передбачає поділ тренувального циклу на блоки, кожен з яких спрямований на розвиток певної якості: блок накопичення, блок трансформації і блок реалізації. Цей підхід дозволяє сконцентрувати тренувальний стимул для максимального розвитку конкретних фізичних якостей [19].

Для структурування тренувального тижня використовуються різні підходи до розподілу тренувань. Розподіл за сплітом м'язових груп передбачає поділ тренувань за цільовими м'язовими групами (наприклад, окремі дні для тренування грудей, спини, ніг), що дозволяє приділити достатньо уваги кожній м'язовій групі та забезпечити відпочинок [20].

Альтернативний підхід – розподіл за типом тренувань, наприклад, чергування силових, кардіо, інтервальних тренувань та тренувань на гнучкість. Такий розподіл забезпечує всебічний розвиток фізичних якостей та знижує ризик перевтоми конкретних систем організму [20].

Розподіл за інтенсивністю передбачає чергування тренувань різної інтенсивності протягом тижня: важкі, середні та легкі [20]. Такий розподіл враховує хвилеподібну природу відновлення та адаптації, забезпечуючи оптимальний баланс навантаження та відпочинку.

Для оцінки ефективності тренувальних програм використовуються різноманітні методи моніторингу та оцінки прогресу: антропометричні вимірювання (вага, обхвати тіла, товщина шкірних складок), тестування фізичних показників (максимальна сила, витривалість, гнучкість), аналіз тренувальних даних (вага, кількість повторень, загальний обсяг).

Сучасні методи індивідуалізації тренувань все частіше використовують технології для збору та аналізу даних про тренувальний процес та реакцію

організму. Смарт-годинники та фітнес-трекери дозволяють відстежувати серцевий ритм, активність, якість сну та інші параметри, які можуть впливати на ефективність тренувань [21]. Спеціалізовані програми та застосунки автоматизують планування тренувань на основі індивідуальних параметрів та цілей, а також аналізують прогрес для оптимізації майбутніх тренувань.

Окремо слід відзначити методи автоматичного регулювання навантаження на основі зворотного зв'язку. Метод автоматичного збільшення чи зменшення ваги залежно від виконання цільової кількості повторень (наприклад, якщо вдалося виконати більше 12 повторень, вага збільшується на наступному тренуванні, якщо менше 8 – зменшується) дозволяє підтримувати оптимальну інтенсивність [22].

Для забезпечення комплексного розвитку важливі методи оптимального поєднання різних типів тренувань. Дослідження показують, що конкуруючі адаптації можуть виникати при одночасному тренуванні різних фізичних якостей, особливо сили та витривалості [23]. Для мінімізації цього ефекту розроблені методи оптимального розподілу тренувань різного типу протягом тижня та дня, врахування часу відновлення між тренуваннями та пріоритезації тренувальних цілей.

Використання цих методів та алгоритмів дозволяє створювати науково обґрунтовані, ефективні та індивідуалізовані тренувальні програми, які максимально відповідають потребам та можливостям конкретної людини, забезпечуючи стабільний прогрес та мінімізуючи ризики перетренованості та травм.

1.3 Аналіз сучасних програмних засобів для індивідуального планування спортивних тренувань

В сучасному ринку представлено багато програмних засобів, які допомагають користувачам планувати та відстежувати свої спортивні тренування. Ці застосунки відрізняються між собою функціональністю,

підходами до індивідуалізації та цільовою аудиторією, але всі вони спрямовані на оптимізацію тренувального процесу та досягнення кращих результатів.

Fitbod є одним з популярних програмних засобів, який використовує алгоритми машинного навчання для створення персоналізованих тренувальних програм (рис. 1.1). Ключовою особливістю Fitbod є здатність адаптувати тренування на основі попередніх результатів, наявного обладнання та цілей користувача. Застосунок аналізує дані про попередні тренування та обчислює рівень відновлення різних м'язових груп, що дозволяє оптимально розподілити навантаження та забезпечити постійний прогрес. Fitbod автоматично налаштовує програму тренувань на основі отриманих даних від користувача – якщо вправа виявилась занадто складною або легкою, параметри наступних тренувань зміняться.

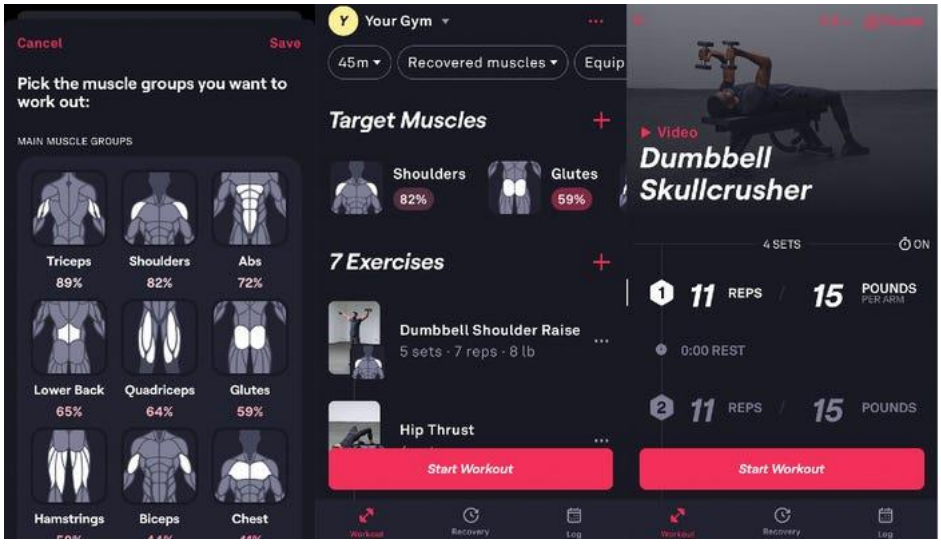

Рисунок 1.1 – Загальний вигляд інтерфейсного вікна програмного засобу «Fitbod»

Цей програмний засіб пропонує понад 400 вправ з детальними інструкціями та відео, що допоможе користувачам правильно виконувати рухи. Однак, незважаючи на високий рівень автоматизації, Fitbod має обмежені можливості для планування кардіотренувань та не враховує деякі фізіологічні параметри користувача [24].

Jefit є комплексним програмним засобом, який поєднує функції планування тренувань, відстеження прогресу та соціальну мережу для любителів. На відміну від Fitbod, Jefit дозволяє користувачам створювати власні тренувальні програми або вибирати з бібліотеки готових шаблонів. Застосунок містить велику базу даних вправ (понад 1300) з детальними інструкціями та відео-демонстраціями. Jefit забезпечує детальне відстеження тренувань з можливістю запису ваги, повторень, підходів, а також надає аналіз прогресу з візуалізацією у вигляді графіків (рис. 1.2).

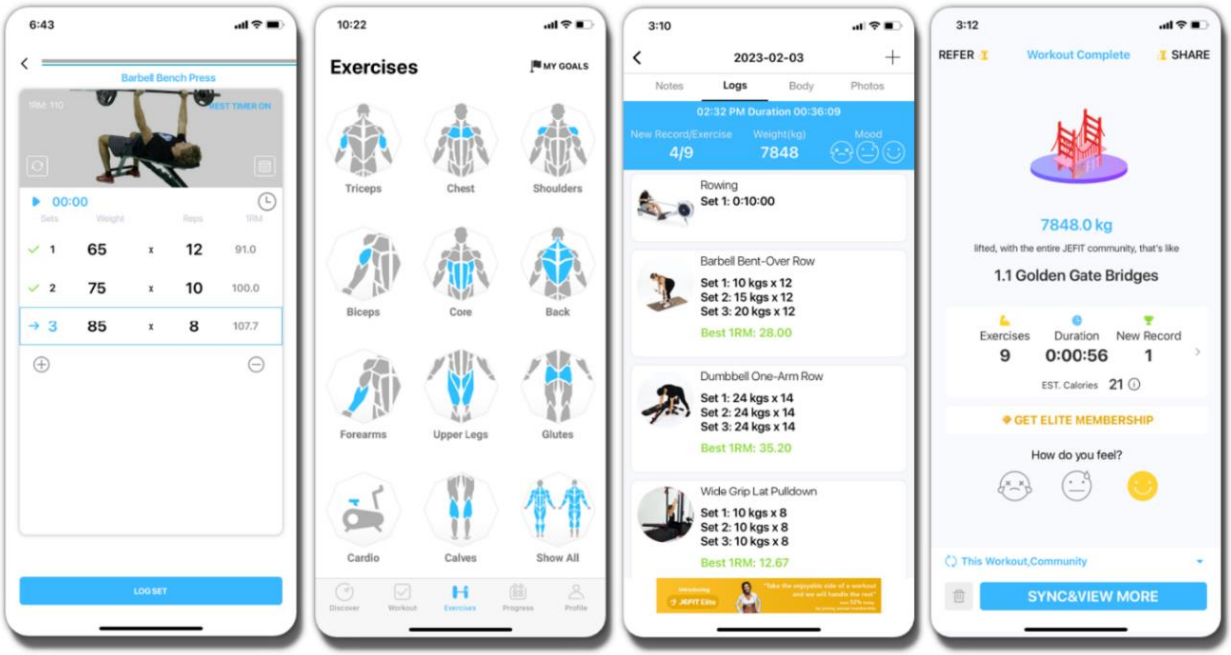

Рисунок 1.2 – Загальний вигляд інтерфейсного вікна програмного засобу «Jefit»

Програмний засіб використовує систему прогресивного перевантаження, що допомагає користувачам поступово збільшувати вагу або кількість повторень на основі результатів попередніх тренувань. Перевагами Jefit є гнучкість налаштування, детальна статистика тренувань та можливість створення високоспеціалізованих програм. Однак цей застосунок не пропонує по-справжньому індивідуальних програм на основі автоматичного аналізу

даних та потреб користувача, що вимагає від користувачів базового розуміння про принципи планування тренувань [25].

Nike Training Club пропонує широкий вибір готових тренувань різного типу: силові, кардіо, йога, пілатес та комбіновані програми. Особливістю цього програмного засобу є відео-інструкції високої якості від професійних тренерів та спортсменів. Застосунок дозволяє фільтрувати тренування за рівнем складності, необхідним обладнанням, тривалістю та цілями. Nike Training Club пропонує багатотижневі програми тренувань, спрямовані на досягнення конкретних цілей, таких як схуднення, набір м'язової маси або підвищення витривалості (рис. 1.3).

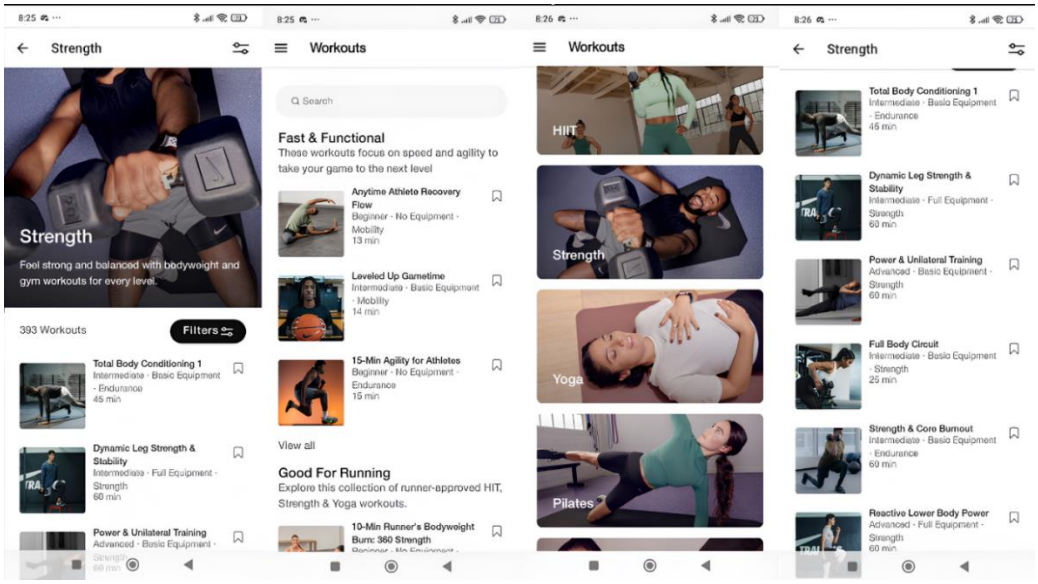

Рисунок 1.3 – Загальний вигляд інтерфейсного вікна програмного засобу «Nike Training Club»

Важливою функцією цієї програми є можливість отримання рекомендацій на основі результатів фітнес-оцінки, яка допомагає визначити поточний рівень фізичної підготовки. Застосунок інтегрується з іншими продуктами Nike, такими як Nike Run Club, для комплексного відстеження фізичної активності. Проте ступінь індивідуальності в Nike Training Club є обмеженою – користувачі вибирають з готових програм, а не отримують

тренування, адаптовані під їхні особливості та цілі. Крім того, застосунок не пропонує детального відстеження прогресу для силових тренувань з обмеженнями [26].

MyFitnessPal, хоч і не є суто застосунком для планування тренувань, але заслуговує на увагу через свої функції відслідковування фізичної активності та можливість інтеграції з іншими фітнес-застосунками. Основною функцією MyFitnessPal є слідкування за харчуванням та калоріями, він також дозволяє записувати тренування та спалені калорії. Застосунок має велику базу даних вправ з оцінкою витрат калорій, що дозволяє користувачам відстежувати загальний енергетичний баланс (рис. 1.4). MyFitnessPal інтегрується з багатьма фітнес-трекерами та застосунками, імпортуючи дані про активність для комплексного аналізу. Користувачі можуть встановлювати цілі щодо фізичної активності та відстежувати прогрес у їх досягненні.

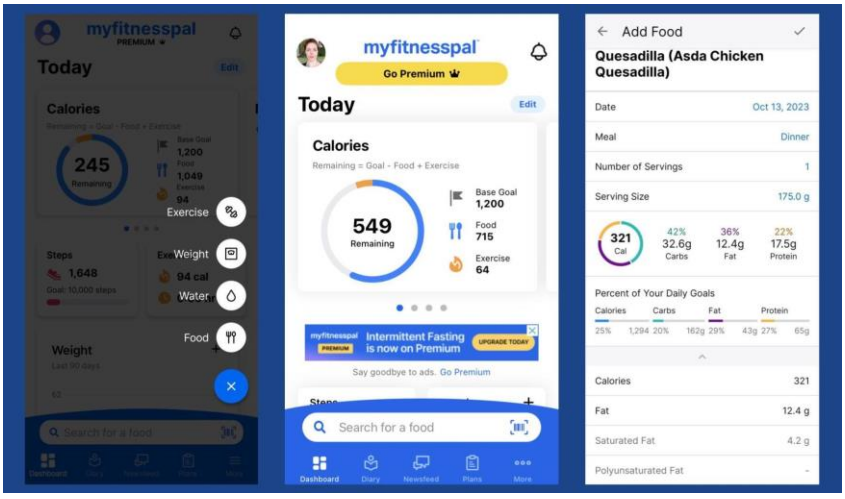

Рисунок 1.4 – Загальний вигляд інтерфейсного вікна програмного засобу «MyFitnessPal»

Цінністю MyFitnessPal є здатність об'єднувати дані про харчування та фізичну активність, що є важливим для таких цілей, як схуднення або набір м'язової маси. Однак застосунок не пропонує функцій планування тренувань і має обмежені можливості для детального відстеження параметрів силових тренувань [27].

Strava є популярним програмним засобом, який орієнтований на циклічні види активності, такі як біг, велоспорт, плавання та ходьба. Основною особливістю Strava є можливість відстежування маршрутів через GPS, аналіз швидкості, темпу, серцебиття та інших показників. Застосунок надає детальну статистику та візуалізацію тренувань, що дозволяє аналізувати прогрес та ефективність (рис. 1.5). Важливою функцією є можливість створення сегментів – ділянок маршруту, на яких можна змагатися з іншими користувачами або з власними попередніми результатами.

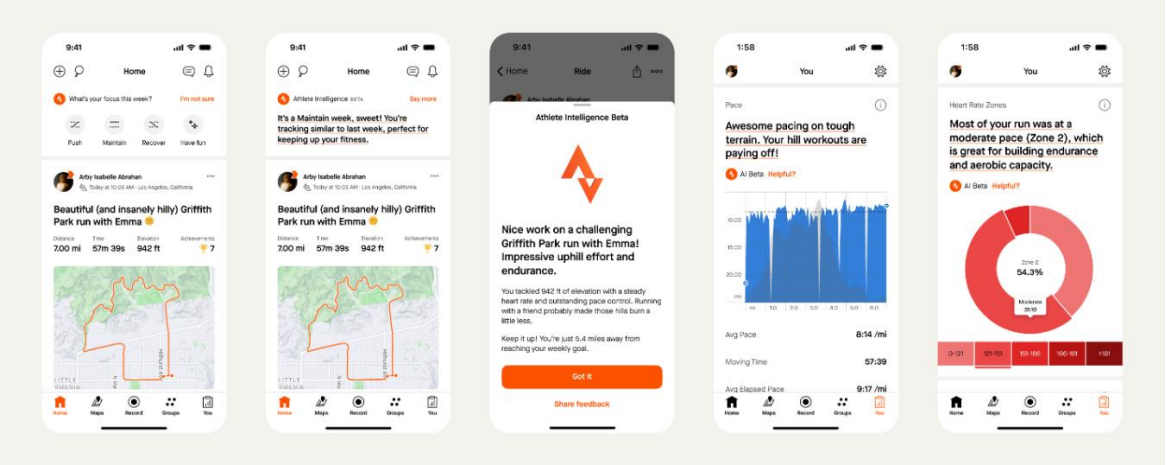

Рисунок 1.5 – Загальний вигляд інтерфейсного вікна програмного засобу «Strava»

Strava пропонує функції соціальної мережі, де користувачі можуть ділитися своїми тренуваннями, отримувати підтримку від друзів та брати участь у групових викликах. Для преміум-користувачів доступні додаткові функції, такі як персоналізовані тренувальні плани, розширена аналітика та індивідуальні цілі. Програмний засіб інтегрується з різними фітнес-трекерами та смарт-годинниками, що забезпечує точність та зручність відстеження тренувань. Однак Strava має обмежені можливості для планування та відстеження силових тренувань, що обмежує її корисність для комплексної фізичної підготовки [28].

Аналіз сучасних програмних засобів для індивідуального планування спортивних тренувань дозволяє виявити кілька важливих тенденцій та обмежень. Більшість застосунків або ставлять фокус на автоматичному створенні програм без достатньої гнучкості, або надають інструменти для відстеження без функцій автоматичного планування. Лише деякі програмні засоби пропонують справді індивідуалізований підхід, який враховує всі аспекти тренувального процесу.

Крім того, багато застосунків спеціалізуються на певному типі тренувань (силові, кардіо), що обмежує можливості для комплексного фізичного розвитку. Важливим обмеженням також є недостатня увага до техніки виконання вправ та запобігання травмам, що є критичним для безпечного та ефективного тренування. Розробка програмного модуля, який поєднує гнучкість налаштування, автоматичне планування на основі наукових принципів та комплексний підхід до різних типів тренувань, може значно покращити якість індивідуального планування спортивних тренувань.

1.4 Висновок до розділу 1

У результаті аналізу предметної області індивідуального планування спортивних тренувань було визначено ключові аспекти, методи, алгоритми та програмні засоби, що використовуються в цій сфері.

Розглянуто основні принципи індивідуального планування тренувань, такі як принцип індивідуалізації, принцип поступовості, принцип систематичності та принцип варіативності. З'ясовано, що ефективне планування тренувань вимагає врахування багатьох факторів, включаючи фізіологічні параметри, рівень фізичної підготовки, тренувальні цілі, медичні обмеження та доступність обладнання.

Проаналізовано існуючі методи та алгоритми індивідуального планування тренувань, зокрема методи визначення оптимального навантаження на основі концепції максимального повторення (1RM), метод

суб'єктивної оцінки навантаження (RPE), алгоритми прогресії навантаження (лінійна, хвилеподібна, ступінчаста) та методи періодизації (лінійна, нелінійна, блокова). Розглянуто різні підходи до структурування тренувального тижня та методи оцінки ефективності тренувальних програм.

Досліджено сучасні програмні засоби для індивідуального планування спортивних тренувань, такі як Fitbod, Jefit, Nike Training Club, MyFitnessPal та Strava. Визначено їхні сильні та слабкі сторони, особливості функціонування та підходи до індивідуалізації тренувань. Виявлено, що більшість існуючих програмних засобів мають певні обмеження щодо рівня індивідуалізації, комплексності підходу або зручності використання.

На основі проведеного аналізу можна стверджувати, що існує потреба в розробці програмного модуля для індивідуального планування спортивних тренувань, який би поєднував науково обґрунтовані принципи та методи тренувань, високий рівень індивідуалізації та зручний інтерфейс користувача. Такий програмний модуль має враховувати індивідуальні особливості користувача, його цілі, доступне обладнання та часові обмеження, а також забезпечувати адаптацію тренувальних програм на основі прогресу та зворотного зв'язку.

Розробка програмного модуля індивідуального планування спортивних тренувань вимагає комплексного підходу, який включає використання математичних моделей для оптимізації тренувальних параметрів, алгоритмів адаптації програм до індивідуальних особливостей користувачів та методів аналізу прогресу для коригування тренувальних планів.

2 ПРОЄКТУВАННЯ ПРОГРАМОГО МОДУЛЯ ІНДИВІДУАЛЬНОГО ПЛАНУВАННЯ СПОРТИВНИХ ТРЕНУВАНЬ

2.1 Розробка архітектури програмного модуля індивідуального планування спортивних тренувань

Архітектура програмного модуля індивідуального планування спортивних тренувань є фундаментальним елементом, який визначає загальну організацію і структуру системи. Вона відокремлює основні компоненти, їхні властивості та взаємозв'язки, створюючи концептуальну основу для подальшої розробки програмного забезпечення. Правильно спроектована архітектура стає ключовим фактором успіху програми, забезпечуючи його стабільність, надійність та гнучкість [29].

Розробка архітектури програмного модуля для індивідуального планування спортивних тренувань потребує комплексного підходу, який би враховував специфіку предметної області, особливості користувацького досвіду та технічні аспекти реалізації. Індивідуальне планування тренувань є складним процесом, оскільки вимагає врахування багатьох факторів, таких як фізіологічні особливості спортсмена, цілі тренувань, доступні ресурси, часові обмеження та інші параметри. Ці фактори значною мірою впливають на вибір архітектурних рішень та організацію компонентів системи.

При проектуванні архітектури даного програмного модуля має бути враховано специфіку сфери спортивних тренувань, особливості індивідуального планування та вимоги до гнучкості системи. Важливими аспектами були також питання безпеки даних, оскільки система оперує персональною інформацією користувачів, та питання продуктивності, оскільки система має забезпечувати швидке генерування та оновлення тренувальних планів.

На рисунку 2.1 представлено загальну архітектуру програмного модуля індивідуального планування спортивних тренувань.

Рисунок 2.1 – Загальна архітектура програмного модуля індивідуального планування спортивних тренувань

Подана архітектура програмного модуля слідує класичній трирівневій моделі, яка включає: презентаційний рівень, рівень бізнес-логіки та рівень даних. Такий підхід забезпечує чіткий розподіл відповідальності між різними компонентами системи та сприяє її модульності та масштабованості. Трирівнева архітектура є оптимальним вибором для даного програмного модуля, оскільки вона дозволяє ефективно розділити інтерфейс користувача, бізнес-логіку та управління даними, що спрощує розробку, тестування та підтримку програмного продукту.

Презентаційний рівень представлений графічним інтерфейсом користувача, який забезпечує взаємодію між користувачем та системою. Цей рівень відповідатиме за відображення інформації, прийом команд від користувача та передачу цих команд до рівня бізнес-логіки.

Особливу увагу при розробці презентаційного рівня буде приділено питанням доступності та зручності використання. Інтерфейс користувача повинен забезпечити максимально простий та інтуїтивно зрозумілий процес взаємодії з системою. Буде використано сучасні підходи до проектування інтерфейсів, які забезпечують добре читабельний текст, чітку навігацію,

логічне розташування елементів управління та зрозумілий шлях виконання завдань.

Рівень бізнес-логіки стане ядром системи та міститиме основні алгоритми та правила, які визначають функціональність програмного модуля. Він оброблятиме запити від презентаційного рівня, виконуючи необхідні обчислення та звертатиметься до рівня даних для отримання чи збереження інформації. Саме на цьому рівні буде реалізовано алгоритми індивідуального планування спортивних тренувань, аналізу ефективності тренувань та генерації рекомендацій.

Бізнес-логіка програмного модуля включатиме складні алгоритми для створення індивідуальних тренувальних програм, які враховуватимуть фізіологічні параметри користувача (вік, стать, вага, зріст), його рівень фізичної підготовки, цілі тренувань (схуднення, набір м'язової маси, підвищення витривалості, тощо), медичні обмеження, доступне обладнання та інші фактори.

Крім того, на цьому рівні буде реалізовано механізми для аналізу ефективності тренувань на основі даних, введених користувачем, таких як показники виконаних вправ, суб'єктивні відчуття від тренувань, зміни фізичних параметрів тощо. На основі цього аналізу система зможе автоматично коригувати тренувальні програми для досягнення оптимальних результатів.

Ще одним важливим компонентом бізнес-логіки будуть системи рекомендацій, які надаватимуть користувачам поради щодо оптимізації тренувального процесу, корекції харчування, режиму відпочинку та інших аспектів, що впливають на ефективність тренувань.

Рівень даних буде відповідати за зберігання та управління всією інформацією, необхідною для функціонування системи. Також забезпечуватиме доступ до даних про користувачів, тренувальні плани, вправи, а також статистику та аналітику. Важливими аспектами цього рівня

будуть забезпечення цілісності даних, їх захист від несанкціонованого доступу та ефективне управління.

Програмний модуль буде розроблено з використанням об'єктно-орієнтованого програмування (ООП), що дозволить чітко структурувати код і забезпечить гнучкість та розширюваність системи. Для зберігання даних буде використано об'єктно-документний підхід з серіалізацією об'єктів у JSON-файли, що сможе доповнити ООП-архітектуру програми. Така комбінація дозволить ефективно моделювати складні взаємозв'язки між сутностями системи (користувачами, вправами, тренуваннями) у вигляді класів з методами, а потім зберігати їх стан у легко читабельному JSON-форматі. Цей метод забезпечить гнучкість схеми даних і спростить розробку, зменшуючи залежності системи від зовнішніх СУБД.

Архітектура програмного модуля також передбачатиме наявність допоміжних компонентів, таких як система автентифікації та авторизації, система логування та система безпеки, які забезпечують стабільне та безпечне функціонування всієї системи.

Система автентифікації та авторизації забезпечуватиме контроль доступу до системи та її функцій. Вона буде виконувати функцію перевірки особи користувача під час входу до системи (автентифікація) та визначатиме, які функції та дані доступні конкретному користувачу (авторизація). Це дозволяє захистити персональні дані користувачів та забезпечити контроль доступу до різних функцій системи.

Система логування відповідатиме за реєстрацію подій в системі, що дозволятиме відстежувати активність користувачів, виявляти та усувати помилки, а також аналізувати роботу системи для її подальшої оптимізації. Система безпеки забезпечуватиме захист даних від несанкціонованого доступу, шифрування чутливої інформації та захист від різних видів атак.

Важливою особливістю розробленої архітектури буде її гнучкість та розширюваність, що дозволятиме легко додавати нові функціональні

можливості та адаптувати систему до змінних вимог. Це досягається через використання принципів об'єктно-орієнтованого програмування.

Загальний процес функціонування програмного модуля можна представити у вигляді алгоритму, який описує основні етапи роботи системи від запуску до завершення роботи. На рисунку 2.2 представлено загальний алгоритм роботи програмного модуля індивідуального планування спортивних тренувань.

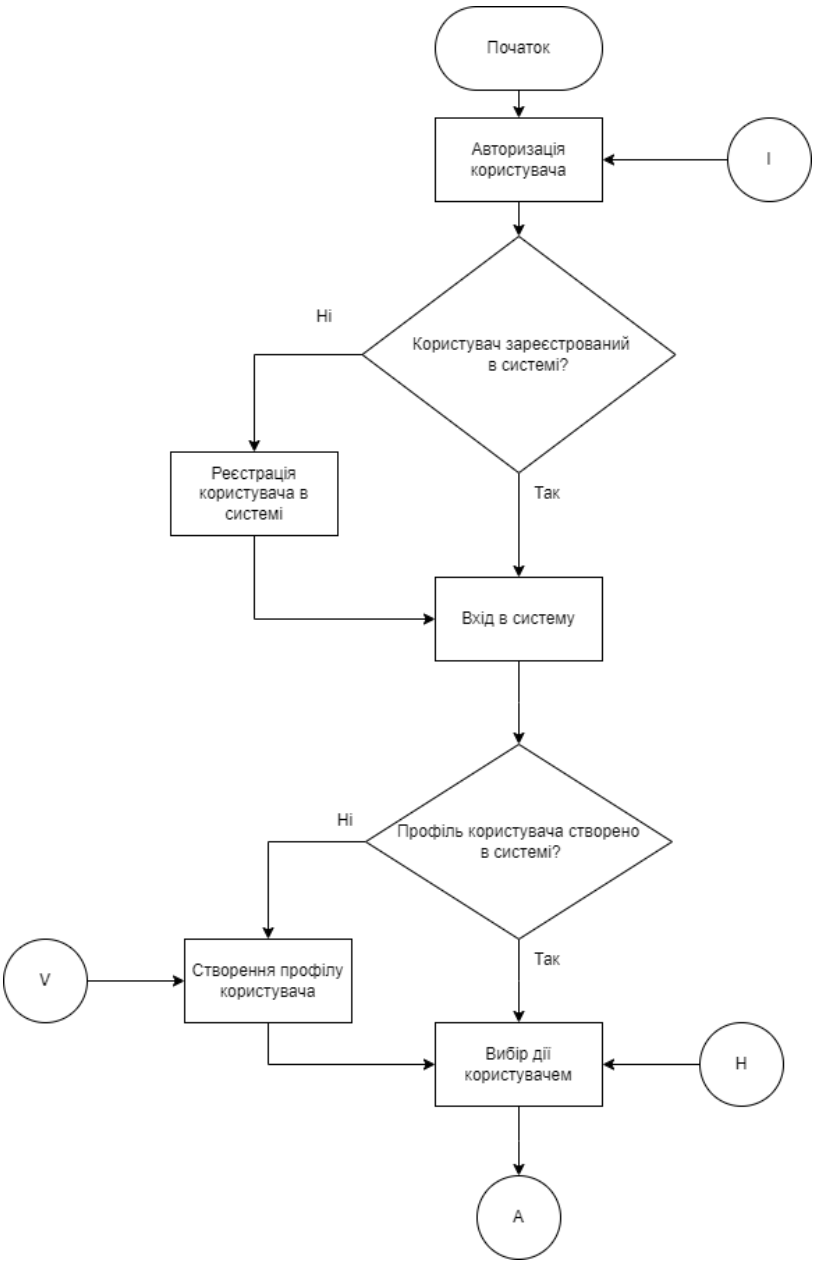

Рисунок 2.2 – Схема загального алгоритму роботи програмного модуля індивідуального планування спортивних тренувань

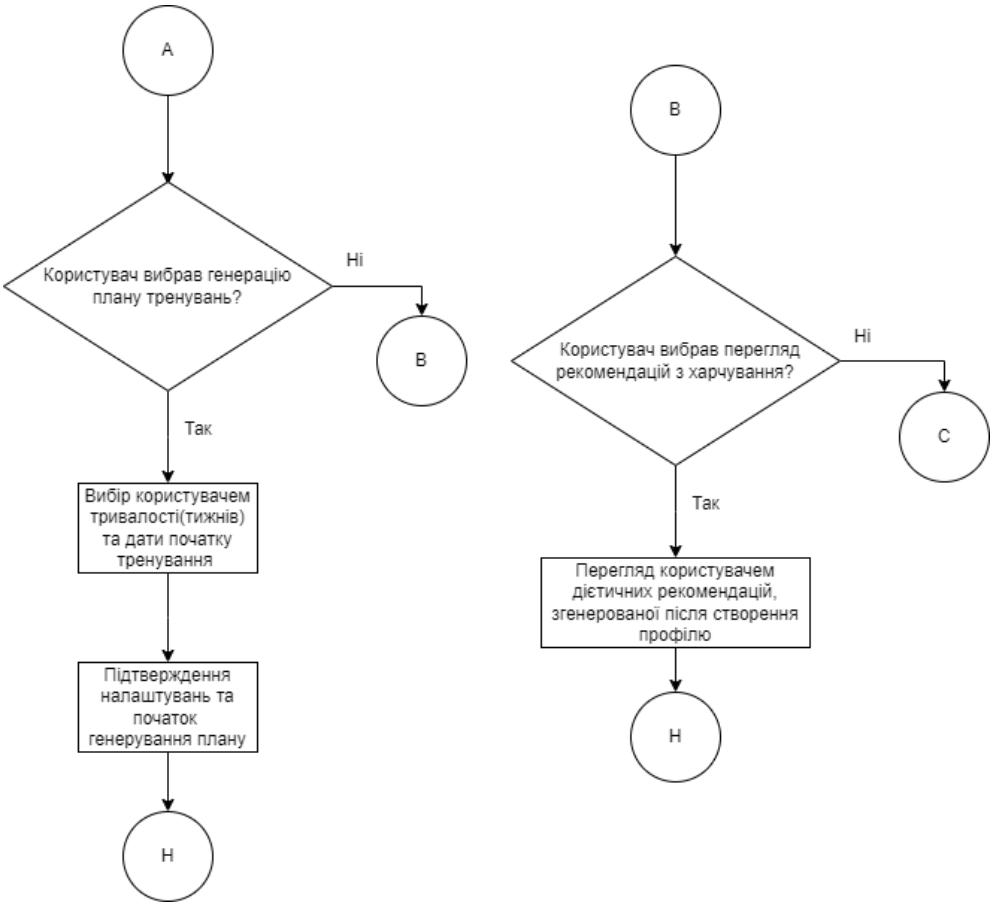

Рисунок 2.2 – Продовження

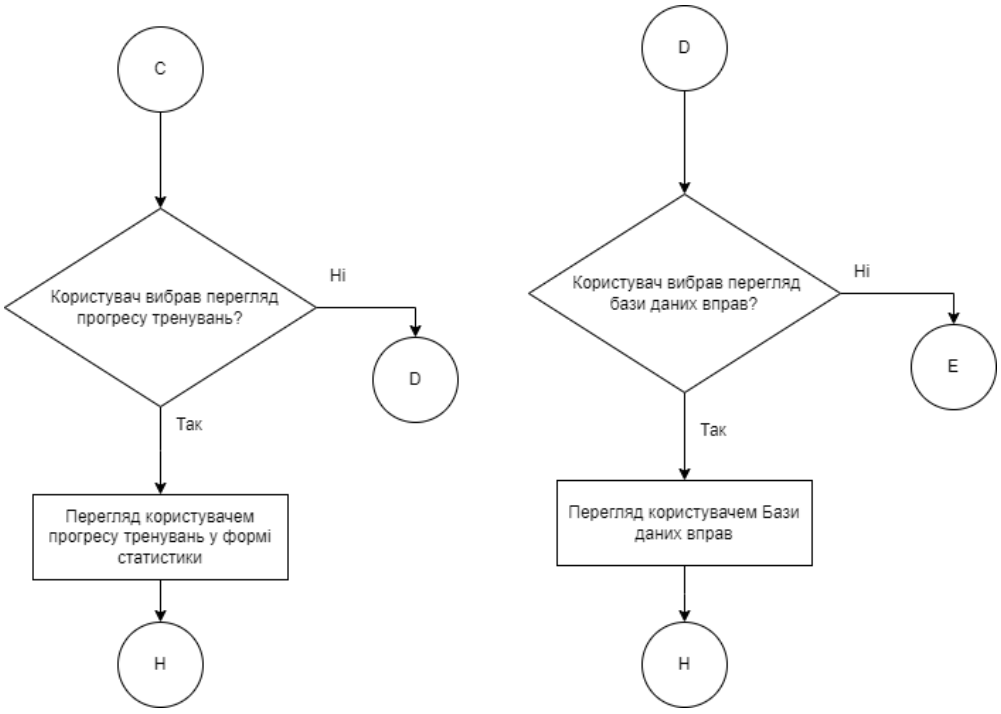

Рисунок 2.2 – Продовження

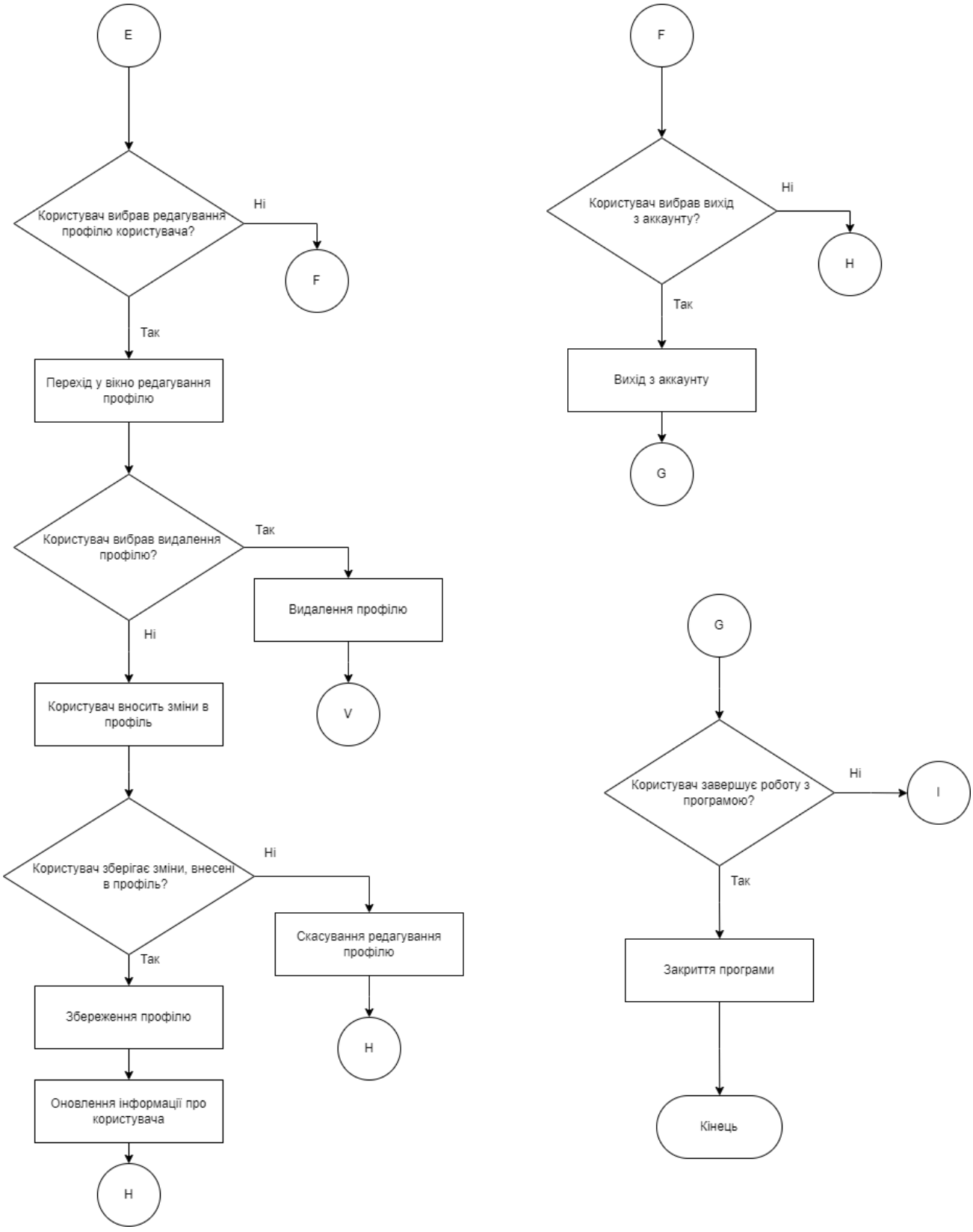

Рисунок 2.2 – Продовження

Алгоритм починатиметься з ініціалізації системи, під час якої завантажуються необхідні компоненти та встановлюються з'єднання з базою даних. Цей етап є критичним для забезпечення стабільної роботи системи, оскільки помилки на цьому етапі можуть призвести до неправильної роботи всієї системи.

Далі відбуватиметься автентифікація користувача, яка може включати введення логіну та паролю або реєстрацію нового користувача. Цей етап забезпечуватиме контроль доступу до системи та захист персональних даних користувачів. Для забезпечення безпеки можуть бути використані сучасні методи захисту, такі як хешування паролів, захист від підбору паролів тощо.

Після успішної автентифікації користувач потраплятиме до головного меню, де йому будуть доступні різні функції системи, такі як перегляд та редагування профілю, створення нового тренувального плану, перегляд існуючих планів, аналіз прогресу, отримання рекомендацій тощо. Головне меню буде розроблено таким чином, щоб забезпечити швидкий та зручний доступ до найбільш часто використовуваних функцій.

Залежно від обраної функції, система буде активізувати відповідний модуль бізнес-логіки, який виконуватиме необхідні операції та взаємодіє з базою даних для отримання чи збереження інформації. Кожен модуль бізнес-логіки відповідатиме за конкретну функціональність системи та реалізує відповідні алгоритми та правила.

Коли користувач вирішить створити новий тренувальний план, система почне збирати інформацію з профілю користувача та додаткові параметри, введені користувачем, такі як цілі тренувань, доступне обладнання, часові обмеження тощо. Цей етап є особливо важливим, оскільки точність та повнота зібраної інформації безпосередньо впливають на якість генерованого тренувального плану.

На основі зібраних даних система генеруватиме індивідуальний тренувальний план, який буде враховувати всі введені параметри та застосовувати наукові принципи тренувань для створення оптимальної

програми. Генерація плану буде базуватися на складних алгоритмах, які враховуватимуть принципи періодизації тренувань, оптимальне співвідношення навантаження та відпочинку, індивідуальні особливості користувача та інші фактори.

Згенерований план представлятиметься користувачу для перегляду та редагування. Користувач сможе внести зміни в план, наприклад, змінити вправи, кількість підходів та повторень, дні тренувань тощо. Така можливість забезпечуватиме більшу гнучкість системи та дозволяє враховувати особисті преференції користувача.

Під час виконання тренувань користувач сможе вносити дані про виконані вправи, досягнуті результати (вага, кількість повторень, тривалість) та суб'єктивні відчуття (складність вправи, відчуття втоми, наявність болю тощо). Ця інформація є надзвичайно важливою для аналізу ефективності тренувального плану та його подальшої корекції.

Система аналізуватиме введені дані та оцінюватиме ефективність тренувального плану, порівнюючи фактичні результати з очікуваними та відстежуючи прогрес користувача. На основі цього аналізу будуть формуватися рекомендації щодо корекції плану, які зможуть включати зміну інтенсивності тренувань, заміну вправ, корекцію режиму відпочинку тощо.

Завершення роботи програми передбачатиме збереження всіх незбережених змін, закриття з'єднань з базою даних та коректне звільнення системних ресурсів. Цей етап є важливим для забезпечення цілісності даних та уникнення їх втрати.

Таким чином, представлена архітектура програмного модуля індивідуального планування спортивних тренувань забезпечуватиме всі необхідні компоненти та взаємозв'язки для ефективного функціонування системи, а загальний алгоритм роботи визначає послідовність дій, які виконуються під час використання програми.

2.2 Розробка взаємодії програмних модулів системи індивідуального планування спортивних тренувань

Розробка ефективної системи індивідуального планування спортивних тренувань потребує забезпечення злагодженої взаємодії між різними програмними модулями. Кожен модуль у системі виконує специфічні функції, але разом вони формують цілісну архітектуру, що дозволяє генерувати персоналізовані тренувальні програми, відстежувати прогрес користувача та надавати рекомендації на основі аналізу даних.

Програмний модуль індивідуального планування спортивних тренувань складається з шести основних компонентів: модуль управління користувачами, модуль оцінки фізичного стану, модуль генерації тренувальних програм, модуль відстеження прогресу, модуль бази даних та модуль інтерфейсу користувача. Структурна схема програмного модуля представлена на рисунку 2.3.

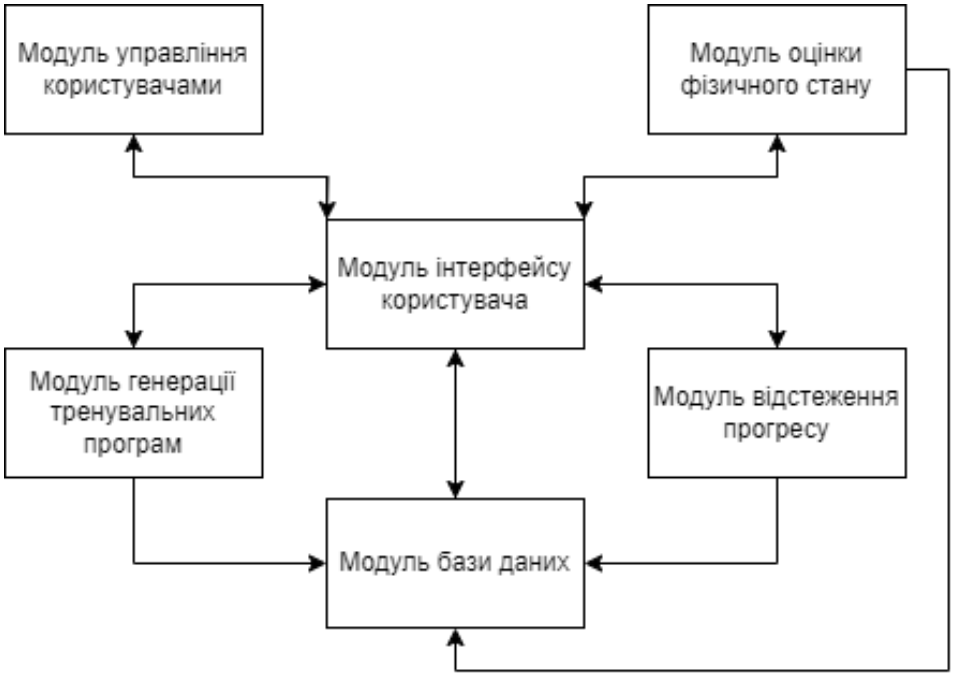

Рисунок 2.3 – Структурна схема програмного модуля індивідуального планування спортивних тренувань

Модуль управління користувачами відповідає за реєстрацію та автентифікацію користувачів, управління профілями та забезпечення безпеки персональних даних. Цей модуль містить компоненти для створення та збереження інформації про користувачів, включаючи особисті дані, параметри фізичного стану, тренувальні цілі та обмеження. Алгоритм роботи модуля управління користувачами представлено на рисунку 2.4.

Він включає реєстрацію нового користувача або автентифікацію існуючого, збір та валідацію даних профілю, збереження даних у базі даних через взаємодію з відповідним модулем та надання доступу до функціоналу системи відповідно до прав користувача.

Рисунок 2.4 – Схема алгоритму роботи модуля управління користувачами

Модуль оцінки фізичного стану є ключовим компонентом системи, який забезпечує аналіз фізичних параметрів користувача та визначення його рівня підготовки. Цей модуль використовує алгоритми для розрахунку таких показників як індекс маси тіла (ІМТ), відсоток жиру в організмі, рівень фізичної підготовки на основі введених користувачем даних та результатів тестувань.

Алгоритм роботи модуля оцінки фізичного стану (рис. 2.5) включає отримання базових антропометричних даних користувача, розрахунок ІМТ та інших базових показників, аналіз даних фізичних тестів, визначення рівня фізичної підготовки, виявлення потенційних обмежень та протипоказань, та передачу результатів оцінки до модуля генерації тренувальних програм.

Рисунок 2.5 – Схема алгоритму роботи модуля оцінки фізичного стану

Модуль генерації тренувальних програм є центральним елементом системи, який на основі даних, отриманих від модуля управління користувачами та модуля оцінки фізичного стану, створює персоналізовані програми тренувань. Цей модуль реалізує складні алгоритми, які враховують рівень підготовки користувача, його цілі, обмеження та доступне обладнання для створення оптимальної програми тренувань.

Алгоритм роботи модуля генерації тренувальних програм (рис. 2.6) включає отримання даних про рівень фізичної підготовки користувача, аналіз тренувальних цілей та обмежень, визначення оптимальної структури тренувального плану, вибір відповідних вправ з бази даних, розрахунок оптимального навантаження, формування повного тренувального плану з розкладом та збереження програми в базі даних.

Модуль відстеження прогресу відповідає за збір та аналіз даних про виконані тренування та досягнуті результати. Цей модуль дозволяє користувачам записувати результати тренувань, такі як вага, кількість повторень, відчуття від тренування, а також відстежувати зміни в фізичних параметрах, таких як вага тіла, обхвати тіла та результати фізичних тестів.

Алгоритм роботи модуля відстеження прогресу (рис. 2.7) включає збір даних про виконані тренування, порівняння фактичних результатів з запланованими, аналіз прогресу з використанням статистичних методів, виявлення тенденцій та потенційних проблем, генерацію рекомендацій для коригування тренувальної програми, візуалізацію прогресу для користувача та передачу даних аналізу до модуля генерації тренувальних програм для адаптації плану.

Рисунок 2.6 – Схема алгоритму роботи модуля генерації тренувальних програм

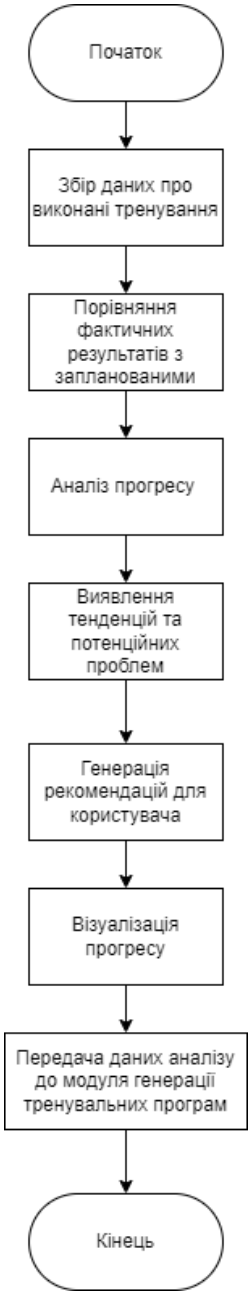

Рисунок 2.7 – Схема алгоритму роботи модуля відстеження прогресу

Модуль бази даних відповідає за зберігання, отримання та управління усіма даними в системі. Цей модуль забезпечує збереження інформації про користувачів, їхні фізичні параметри, тренувальні програми, вправи, результати тренувань та інші дані, необхідні для функціонування системи.

Алгоритм роботи модуля бази даних (рис. 2.8) включає встановлення з'єднання з базою даних, обробку запитів на збереження, отримання,

оновлення або видалення даних, забезпечення цілісності та безпеки даних, оптимізацію запитів для швидкого доступу до даних, резервне копіювання даних та закриття з'єднання з базою даних.

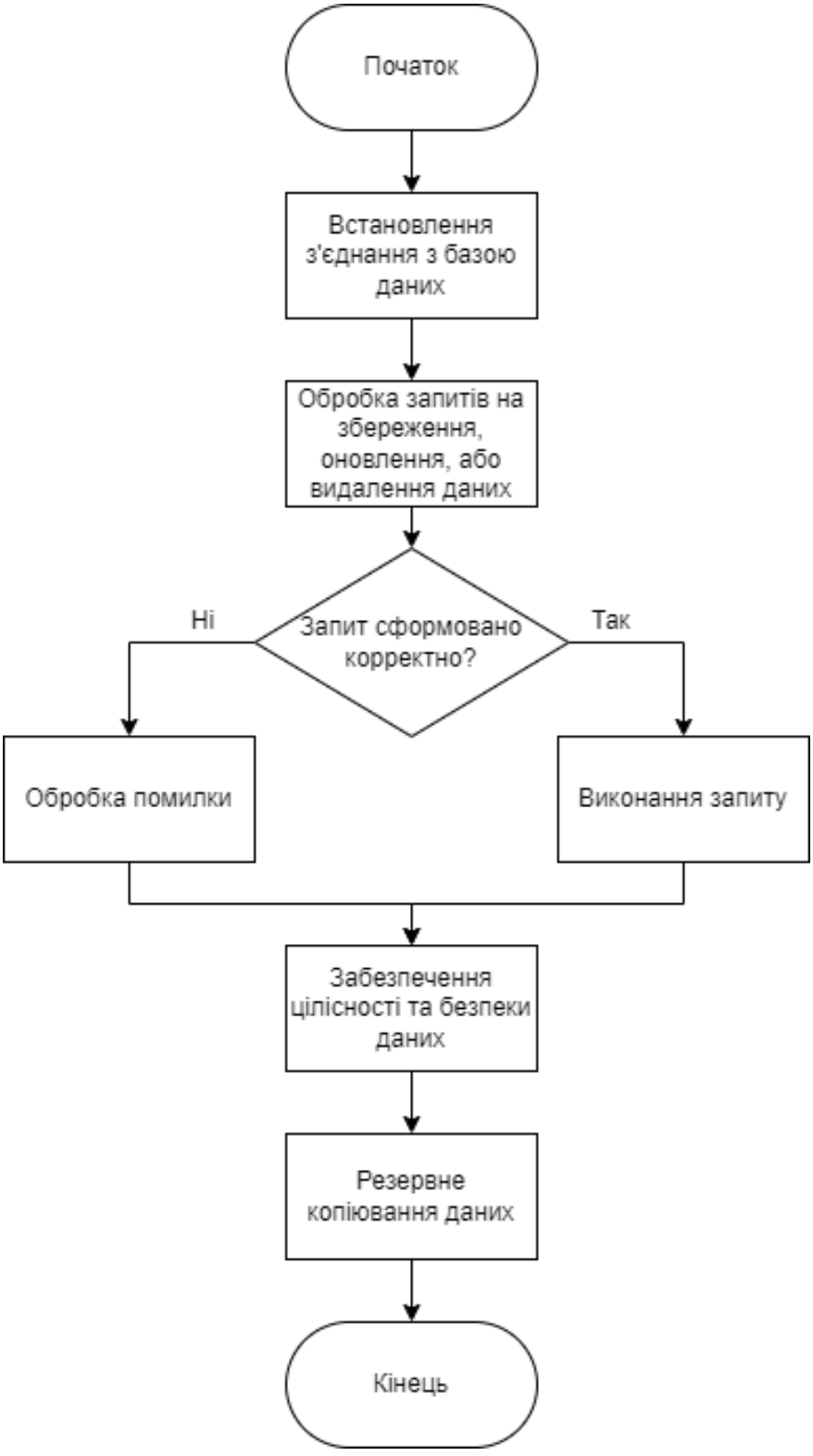

Рисунок 2.8 – Схема алгоритму роботи модуля бази даних

Модуль інтерфейсу користувача є зовнішнім представленням системи, яке забезпечує взаємодію користувача з усіма функціями програмного модуля. Цей модуль відповідає за відображення інформації, прийом команд від користувача та передачу цих команд до відповідних модулів системи.

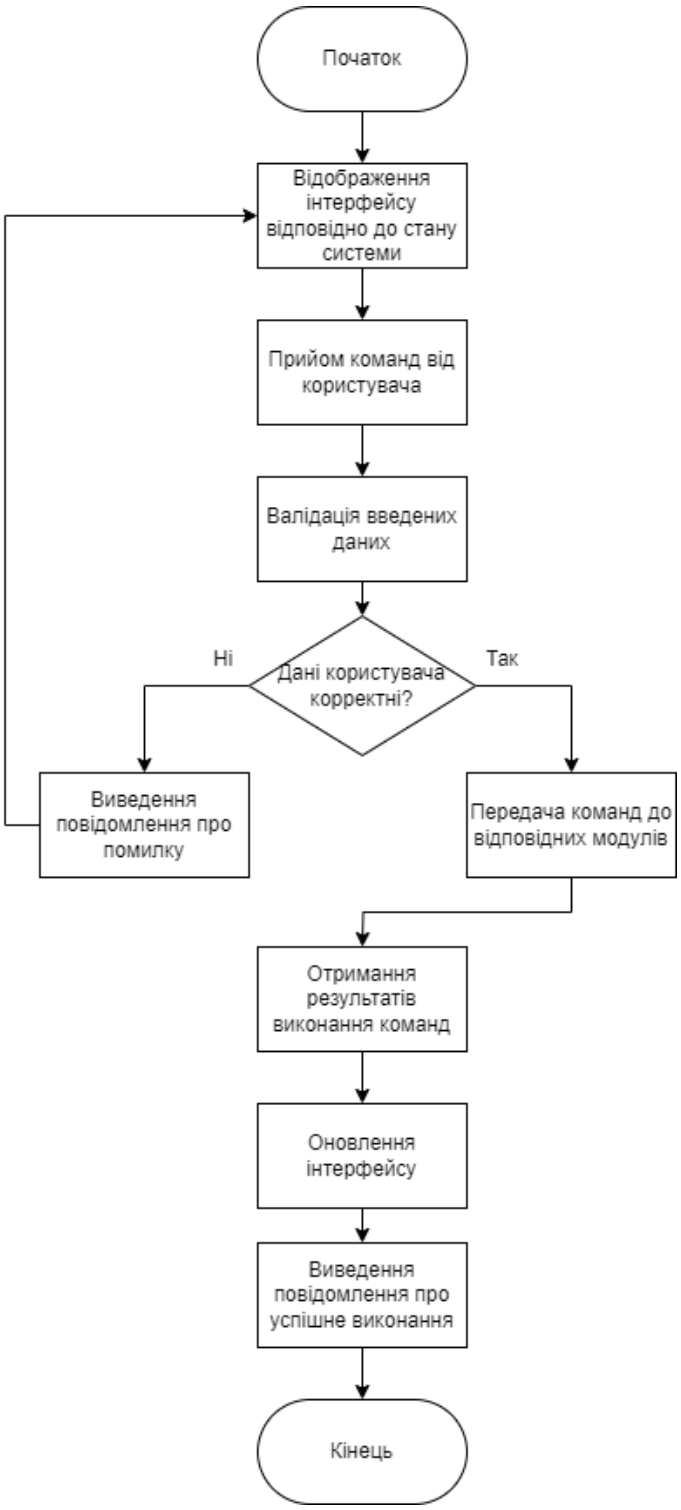

Рисунок 2.9 – Схема алгоритму роботи модуля інтерфейсу користувача

Алгоритм роботи модуля інтерфейсу користувача (рис. 2.9) включає відображення інтерфейсу відповідно до стану системи, прийом команд від користувача, валідацію введених даних, передачу команд до відповідних модулів, отримання результатів виконання команд, оновлення інтерфейсу відповідно до отриманих результатів та відображення повідомлень про помилки або успішне виконання операцій.

Взаємодія між модулями в програмному модулі індивідуального планування спортивних тренувань реалізована у вигляді ієрархічної структури, де модуль інтерфейсу користувача (FitnessPlannerGUI) виступає як центральний координуючий елемент, який має прямі посилання на інші модулі системи. Такий підхід, при якому головний клас взаємодіє з усіма іншими компонентами, забезпечує просту та ефективну комунікацію між модулями, а також спрощує контроль над потоком даних в системі.

Центральний клас системи ініціалізує всі інші компоненти під час запуску програми та зберігає посилання на них для подальшої взаємодії. Це дозволяє централізовано управляти станом системи та забезпечувати узгоджену роботу всіх модулів. Клас також відповідає за обробку подій, таких як вхід користувача в систему, створення профілю, генерація тренувального плану та виконання тренувань.

Для забезпечення зворотного зв'язку між модулями використовується механізм подій, який дозволяє компонентам повідомляти про важливі зміни. Наприклад, коли користувач успішно входить в систему, модуль автентифікації генерує відповідну подію, яка обробляється головним класом для подальшої завантаження профілю користувача та оновлення інтерфейсу.

На рисунку 2.10 представлена UML-діаграма класів системи, яка відображає структуру основних компонентів та їх взаємозв'язки.

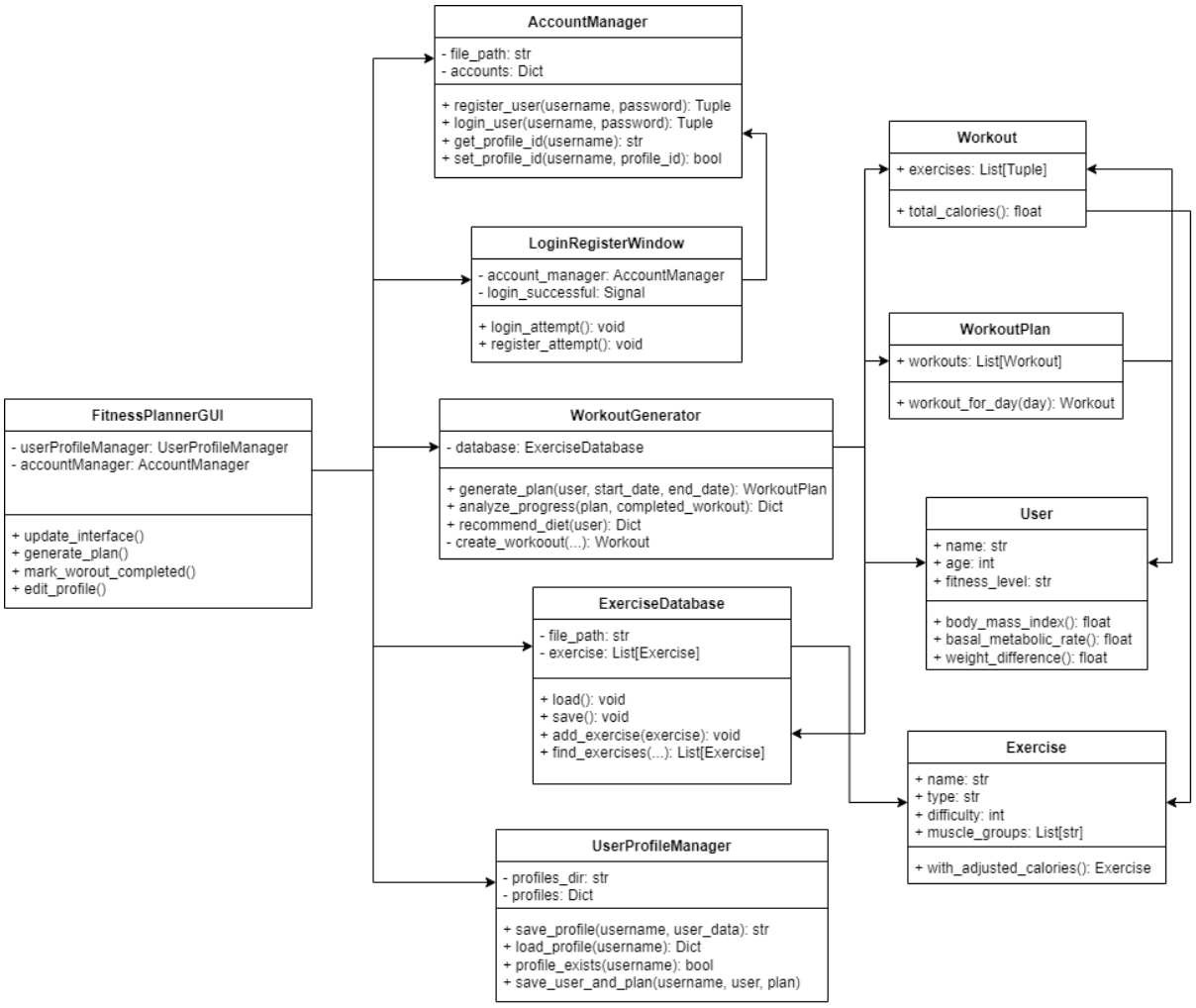

Рисунок 2.10 – UML-діаграма класів системи індивідуального планування спортивних тренувань

UML-діаграма демонструє ключові класи системи та взаємозв'язки між ними. Центральний клас FitnessPlannerGUI взаємодіє з усіма іншими компонентами системи, включаючи UserProfileManager, AccountManager, ExerciseDatabase, WorkoutGenerator та LoginRegisterWindow.

Клас UserProfileManager відповідає за збереження та завантаження профілів користувачів, а також за управління даними про фізичний стан та тренувальні плани. Клас AccountManager забезпечує функціональність реєстрації та автентифікації користувачів, а також захист їх персональних даних. Клас ExerciseDatabase надає доступ до бази даних вправ, що використовуються при генерації тренувальних програм. Клас

WorkoutGenerator реалізує алгоритми для створення персоналізованих тренувальних програм на основі параметрів користувача та доступних вправ.

Класи User, Workout, WorkoutPlan та Exercise представляють основні сутності системи та містять дані та методи для роботи з ними. Наприклад, клас User зберігає інформацію про користувача, включаючи його фізичні параметри та тренувальні цілі, а також надає методи для розрахунку індексу маси тіла та базового метаболізму.

Важливою частиною архітектури системи є механізм оновлення інтерфейсу при зміні стану, який реалізований через метод update_interface() в центральному класі. Цей метод викликається після виконання основних операцій, таких як вхід користувача, створення профілю, генерація плану тренувань та відмітка виконаних тренувань, забезпечуючи актуальність відображуваної інформації.

Така архітектура взаємодії модулів забезпечує чітку структуру системи, де кожен компонент має свою зону відповідальності, але при цьому всі компоненти ефективно взаємодіють через центральний модуль інтерфейсу. Це спрощує розуміння, розробку та підтримку системи, а також забезпечує її гнучкість та розширюваність.

2.3 Розробка бази даних програмного модуля індивідуального планування спортивних тренувань

Розробка бази даних програмного модуля індивідуального планування спортивних тренувань є важливим етапом створення системи, що забезпечує ефективне зберігання та управління даними користувачів, тренувальних планів та вправ. При проєктуванні бази даних було враховано особливості предметної області, а також необхідність забезпечення гнучкості та розширюваності програмного модуля.

Для реалізації бази даних було обрано об'єктно-документний підхід із серіалізацією даних у форматі JSON, що дозволяє зберігати складні структури

даних без необхідності використання зовнішньої СУБД. Такий підхід має низку переваг у контексті розроблюваної системи: простота реалізації, відсутність залежності від зовнішніх СУБД, природна інтеграція з об'єктно-орієнтованим підходом до програмування, а також можливість легкого перенесення даних між різними платформами.

Об'єктно-орієнтована модель даних дозволяє маніпулювати базою, використовуючи принципи об'єктно-орієнтованого програмування. Об'єктна база даних являє собою сукупність взаємозв'язаних об'єктів, які відповідають певній схемі. У контексті програмного модуля індивідуального планування спортивних тренувань об'єктно-орієнтована модель даних буде містити такі основні класи: User, Exercise, Workout, WorkoutPlan, UserProfileManager, ExerciseDatabase, WorkoutGenerator та AccountManager. Побудуємо об'єктно-орієнтовану модель даних предметної області "Індивідуальне планування спортивних тренувань", що зображена на рисунку 2.11.

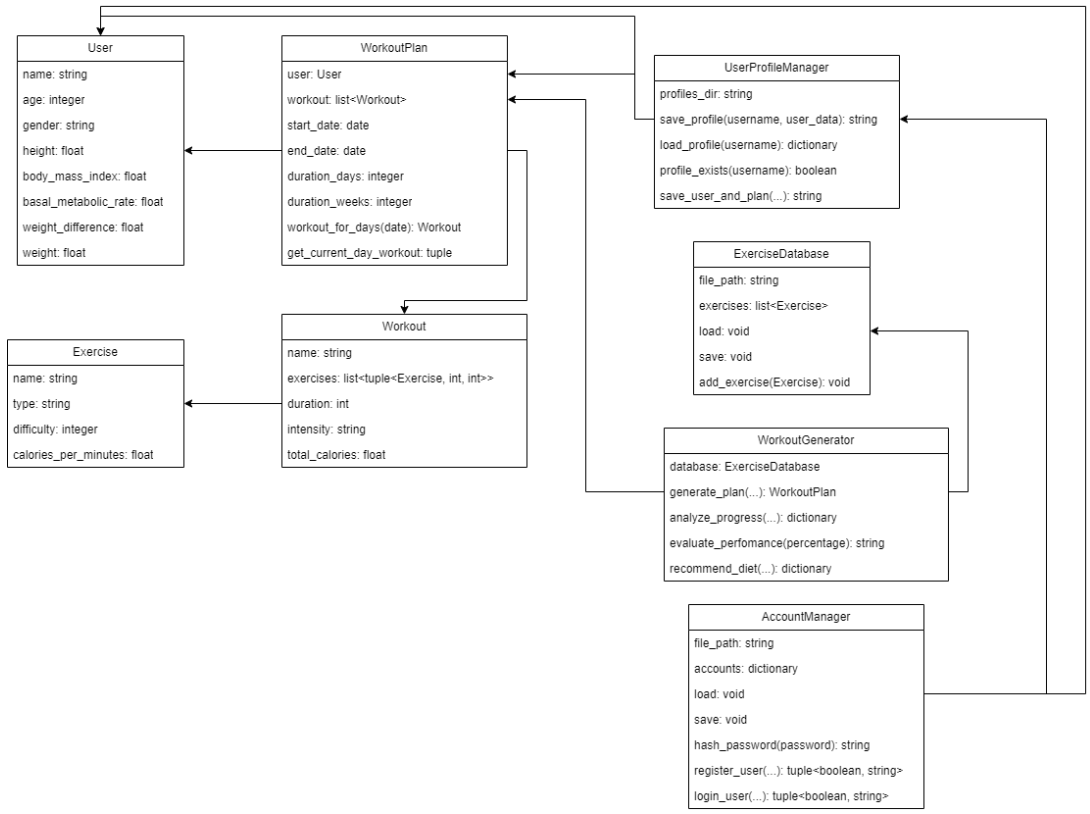

Рисунок 2.11 – Схема об'єктно-орієнтованої моделі даних предметної області "Індивідуальне планування спортивних тренувань"

Клас User буде представляти користувача системи та містить інформацію про його фізичні параметри, цілі тренувань та обмеження. Цей клас включає такі атрибути як ім'я, вік, стать, вага, зріст, рівень фізичної підготовки, цілі тренувань (наприклад, схуднення, набір м'язової маси, підвищення витривалості), обмеження за здоров'ям та доступне обладнання. Клас також містить методи для розрахунку індексу маси тіла (BMI), базового метаболізму та різниці між поточною та бажаною вагою.

Клас Exercise представлятиме фізичну вправу та містить інформацію про її назву, тип (силова, кардіо, гнучкість, тощо), рівень складності (від 1 до 5), групи м'язів, які залучаються при виконанні, приблизні витрати калорій за хвилину, опис вправи та необхідне обладнання. Цей клас дозволяє створювати різноманітні вправи, які потім можуть бути включені до тренувань.

Клас Workout представлятиме одне тренування та містить інформацію про його назву, набір вправ (з кількістю підходів та повторень для кожної), загальну тривалість, інтенсивність, день тижня, додаткові примітки та, за потреби, конкретну дату. Клас включаєтиме метод для розрахунку загальних витрат калорій під час тренування на основі характеристик вправ та кількості підходів і повторень.

Клас WorkoutPlan представлятиме план тренувань та об'єднуватиме кілька тренувань у цілісну програму для конкретного користувача. Клас буде містити інформацію про користувача, для якого створено план, список тренувань, дати початку та закінчення плану, назву плану, загальний рівень інтенсивності та опис цілі щодо зміни ваги, якщо така є. Він також міститиме методи для розрахунку тривалості плану в днях і тижнях, отримання тренування на конкретний день, поточного і наступного тренування, а також тренувань на конкретний тиждень.

Клас UserProfileManager буде відповідати за зберігання та завантаження профілів користувачів. Він буде забезпечувати створення, завантаження та оновлення профілів, а також перевірку наявності профілю для конкретного

користувача. Цей клас працюватиме з файловою системою, зберігаючи дані користувачів у JSON-файлах з унікальними іменами.

Клас ExerciseDatabase керуватиме базою даних вправ, забезпечуючи їх завантаження з JSON-файлу, додавання нових вправ та пошук за різними критеріями (тип, група м'язів, необхідне обладнання, рівень складності). Ця функціональність є критично важливою для генерації персоналізованих тренувальних планів.

Клас WorkoutGenerator буде містити алгоритми для створення персоналізованих тренувальних програм на основі параметрів користувача та доступних вправ. Він також включатиме функціональність для аналізу прогресу тренувань та формування рекомендацій щодо дієти.

Клас AccountManager буде забезпечувати функціональність авторизації та реєстрації користувачів, а також зв'язок між обліковими записами та профілями. Він відповідатиме за безпечне зберігання паролів (з використанням хешування) та управління ідентифікаторами профілів.

Дані в системі будуть зберігатися у кількох JSON-файлах. Основний файл облікових записів міститиме інформацію про користувачів, їхні хешовані паролі та ідентифікатори профілів. Для кожного користувача створюватиметься окремий файл профілю, який буде містити детальну інформацію про користувача, його тренувальні плани, виконані тренування та інші дані. База даних вправ буде зберігатися в окремому JSON-файлі, який міститиме інформацію про всі доступні вправи.

Розроблена база даних буде забезпечувати гнучкість та розширюваність системи, дозволяючи легко додавати нові типи вправ, впроваджувати додаткові метрики для користувачів та тренувань, розширювати аналітичні можливості та інтегруватися з іншими системами. У майбутньому, при збільшенні обсягу даних або кількості користувачів, система зможе бути масштабована шляхом переходу до повноцінної СУБД з мінімальними змінами в логіці роботи з даними.

2.4 Висновок до розділу 2

У результаті виконаної роботи в другому розділі було розроблено архітектуру програмного модуля індивідуального планування спортивних тренувань, що забезпечує ефективне функціонування системи та відповідає всім поставленим вимогам.

Було спроектовано трирівневу архітектуру модуля, яка включає презентаційний рівень, рівень бізнес-логіки та рівень даних. Таке рішення забезпечує чіткий розподіл відповідальності між компонентами системи, підвищує модульність та полегшує подальшу підтримку і розширення функціональності. Для кожного рівня було визначено основні компоненти та їх взаємозв'язки, що забезпечує цілісність та ефективність системи.

Детально опрацьовано алгоритмічну базу програмного рішення від моменту запуску до завершення роботи. Це охоплює всі етапи взаємодії користувача із системою: перевірку облікових даних, збір інформації, формування індивідуальних програм, фіксацію досягнутих результатів та подальший їх аналіз. Особливу увагу приділено механізмам адаптації тренувальних планів на основі фактичних показників.

У ході роботи також було визначено шість основних програмних модулів системи: модуль управління користувачами, модуль оцінки фізичного стану, модуль генерації тренувальних програм, модуль відстеження прогресу, модуль бази даних та модуль інтерфейсу користувача. Для кожного з модулів розроблено алгоритми функціонування, що забезпечують злагоджену взаємодію всіх компонентів системи.

Створено UML-діаграму класів, яка відображає структуру основних компонентів та їх взаємозв'язки. Ця діаграма стала важливим інструментом для розуміння архітектури системи та її подальшої реалізації. Ключовим елементом архітектури став центральний клас FitnessPlannerGUI, який координує взаємодію всіх інших компонентів.

Запропоновано ефективну модель даних на основі об'єктно-орієнтованого підходу, що включає набір класів для відображення ключових сутностей: користувачів, фізичних вправ, окремих тренувань та комплексних програм. Обґрунтовано вибір формату JSON для зберігання інформації, що відзначається простотою впровадження, відсутністю залежності від сторонніх систем керування базами даних та природньою інтеграцією з об'єктно-орієнтованою парадигмою програмування.

Характерною рисою розробленої архітектури є її відкритість до нових функціональних можливостей та здатність адаптуватися до зростаючих потреб користувачів. Система має всі необхідні механізми для генерування персоналізованих тренувальних програм з урахуванням індивідуальних особливостей, аналізу ефективності занять та формування рекомендацій щодо подальшого тренувального процесу.

Таким чином, розроблений програмний модуль індивідуального планування спортивних тренувань має чітку та ефективну архітектуру, що забезпечує реалізацію всіх необхідних функцій та можливість подальшого розвитку системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЮ ІНДИВІДУАЛЬНОГО ПЛАНУВАННЯ СПОРТИВНИХ ПЛАНУВАНЬ

3.1 Обґрунтування вибору мови програмування

Для реалізації програмного модуля індивідуального планування спортивних тренувань була обрана така мова програмування як Python. Обгрунтуємо це рішення, проаналізувавши основні переваги та недоліки цієї мови, а також вкажемо її характерні особливості, що є для неї унікальними.

При виборі мови програмування були враховані особливості предметної області, вимоги до графічного інтерфейсу користувача, необхідність реалізації складних алгоритмів генерації тренувальних планів та специфіка роботи з даними користувачів. Основним критеріям оцінки стали простота реалізації об'єктно-орієнтованої архітектури, можливості створення зручного інтерфейсу та ефективність роботи з JSON-форматом даних.

Було розглянуто два основних варіанти: мова програмування Python та мова програмування Java.

Переваги мови програмування Python:

1. Простота та читабельність коду – Python має зрозумілий синтаксис, що значно спрощує процес розробки та подальшого супроводження програмного забезпечення. Це відіграє важливу роль при реалізації складних алгоритмів генерації тренувальних планів та обробки даних користувачів.

2. Повна підтримка об'єктно-орієнтованого програмування – мова дозволяє ефективно реалізувати спроєктовану архітектуру системи з чітким розділенням класів, що дозволить легко реалізувать класи User, Exercise, Workout, WorkoutPlan та забезпечити належну інкапсуляцію даних.

3. Розвинені можливості для створення графічного інтерфейсу – бібліотека PyQt5, яка наявна в Python, надає всі необхідні інструменти для створення професійного користувацького інтерфейсу з підтримкою сучасних елементів управління, стилізації та адаптивного дизайну.

4. Вбудована підтримка JSON-формату – модуль json дозволяє легко серіалізувати та десеріалізувати об'єкти Python, що ідеально підходить для реалізації зберігання даних без використання зовнішніх СУБД.

5. Багата стандартна бібліотека – Python включає модулі datetime для роботи з датами, математичні функції для розрахунків фізіологічних показників та багато інших різних корисних бібліотек, необхідних для реалізації функціональності системи.

6. Швидкість прототипування – можливість швидко створювати та тестувати алгоритми, що є важливим при розробці системи з складною бізнес-логікою [30-34].

Недоліки мови програмування Python:

1. Низька швидкість виконання – як інтерпретована мова, Python поступається компільованим мовам у швидкості виконання, однак для даного проєкту це не є критичним фактором, оскільки основні операції системи не потребують високої обчислювальної потужності.

2. Залежність від інтерпретатора – для запуску програми необхідно мати встановлений Python, що може створити додаткові вимоги до користувачів [30-34].

Для порівняння з Python було також розглянуто мову програмування Java як альтернативний варіант реалізації системи. Java є широко використовуваною мовою програмування, що має свої переваги та недоліки у контексті розробки програмного модуля індивідуального планування спортивних тренувань [30-34].

Переваги мови програмування Java:

1. Висока продуктивність – компіляція в байт-код та виконання на віртуальній машині Java забезпечує швидкість роботи додатків, що може бути важливим для ресурсоємних обчислень.

2. Строга типізація – система типів Java дозволяє виявляти помилки на етапі компіляції, що підвищує надійність програмного забезпечення та зменшує кількість помилок виконання.

3. Кросплатформність – принцип "написати один раз, запускати скрізь" забезпечує можливість виконання програм на різних операційних системах без додаткових змін у коді.

4. Розвинена екосистема – наявність великої кількості бібліотек та фреймворків для створення графічних інтерфейсів (Swing, JavaFX), роботи з базами даних та реалізації складних алгоритмів [32-34].

Недоліки мови програмування Java для даного проєкту:

1. Складність синтаксису – більш багатослівний код та необхідність написання значної кількості "шаблонного" коду для реалізації простих операцій, що ускладнює швидку розробку прототипів.

2. Необхідність встановлення JVM – користувачі повинні мати встановлену віртуальну машину Java, що створює додаткові вимоги до цільових систем.

3. Повільніше прототипування – більша кількість коду та формальностей уповільнює процес тестування ідей та алгоритмів у порівнянні з більш гнучкими мовами [32-34].

У результаті проведеного аналізу була обрана мова програмування Python як оптимальний варіант для реалізації програмного модуля індивідуального планування спортивних тренувань. Незважаючи на недоліки у низькій швидкості виконання, переваги Python у простоті розробки, читабельності коду та наявності необхідних бібліотек значно переважають переваги Java. Вирішальними факторам при виборі стали зручність швидкого створення та тестування алгоритмів генерації тренувальних планів, а також простота роботи з JSON-файлами для зберігання даних користувачів, що повністю узгоджується з обраною архітектурою системи.

3.2 Програмна реалізація модулю індивідуального планування спортивних тренувань

Програмна реалізація модуля індивідуального планування спортивних тренувань здійснювалася відповідно до спроєктованої архітектури системи та з урахуванням обраних технологічних рішень. Система реалізована з використанням мови програмування Python та бібліотеки PyQt5 для створення графічного інтерфейсу користувача. Програмний код організовано у вигляді модульної структури з чітким розділенням відповідальності між компонентами, що забезпечує легкість розробки, тестування та подальшого супроводження системи.

Основна структура проєкту включає сім основних файлів, кожен з яких відповідає за конкретну функціональність системи:

main.py – головний файл запуску програми;

gui.py – реалізація графічного інтерфейсу користувача;

models.py – класи моделей даних (User, Exercise, Workout, WorkoutPlan);

database.py – управління базою даних вправ;

generator.py – алгоритми генерації тренувальних планів;

account_manager.py – управління обліковими записами користувачів;

login_register.py – інтерфейс авторизації та реєстрації.

Файл «main.py» є основою входу в програму та містить мінімальну логіку запуску системи, лише ініціалізуючи PyQt5 додаток та створюючи головне вікно інтерфейсу користувача.

Файл «gui.py» містить реалізацію повного графічного інтерфейсу користувача через клас FitnessPlannerGUI, який наслідується від QMainWindow та координує роботу всіх компонентів системи. Цей модуль включає створення всіх вкладок інтерфейсу, обробку вибору користувача, відображення даних та взаємодію з іншими модулями системи.

Файл «models.py» реалізує основні класи предметної області, включаючи User для представлення користувачів, Exercise для опису фізичних

вправ, Workout для окремих тренувань та WorkoutPlan для повних тренувальних програм. Ці класи захищають свої дані та містять функції для підрахунку потрібних значень.

Файл «database.py» містить клас ExerciseDatabase, який відповідає за управління базою даних вправ, включаючи завантаження інформації з JSON-файлів, пошук вправ, додавання нових вправ та збереження оновленої інформації.

Файл «generator.py» реалізовує найскладнішу частину системи через клас WorkoutGenerator, а саме алгоритми для генерації персоналізованих тренувальних планів, аналізу прогресу користувачів та формування дієтичних рекомендацій на основі параметрів користувача.

Файл «account_manager.py» забезпечує роботу облікових записів користувачів через клас AccountManager, включаючи реєстрацію нових користувачів, автентифікацію, зберігання паролів та управління зв'язками між обліковими записами та профілями.

Файл «login_register.py» містить окремий інтерфейс для входу та реєстрації користувача через клас LoginRegisterWindow, який забезпечує взаємодію з основним вікном програми через систему сигналів та слотів Qt.

Точкою входу в програму є файл «main.py», який містить мінімальну, але важливу функціональність для запуску всієї системи:

```
import sys
from PyQt5.QtWidgets import QApplication
from gui import FitnessPlannerGUI


def main():
    """Головна функція запуску програми"""
    print("Запуск програмного модуля індивідуального планування спортивних тренувань...")
    app = QApplication(sys.argv)
    # Створення основного вікна програми
```

```
    window = FitnessPlannerGUI()
    # Вихід з додатку коли всі вікна закрито
    sys.exit(app.exec_())
if __name__ == "__main__":
    main().
```

Цей код демонструє простоту запуску програми на Python з використанням PyQt5. Спочатку створюється об'єкт програми QApplication, який необхідний для роботи будь-якого GUI-додатку на PyQt5. Далі створюється основне вікно програми через клас FitnessPlannerGUI, який містить всю логіку інтерфейсу користувача. Метод app.exec_() запускає головний цикл обробки подій, який буде працювати до закриття програми користувачем.

Серцем системи є файл «models.py», який містить реалізацію основних класів предметної області. Ці класи забезпечують інкапсуляцію даних та бізнес-логіки, дозволяючи ефективно моделювати складні взаємозв'язки між користувачами, вправами та тренувальними планами. Використання декораторів dataclass значно спрощує створення класів, автоматично генеруючи методи __init__, __repr__ та інші необхідні методи.

Клас User є одним з найважливіших компонентів системи, оскільки він представляє користувача з усіма його характеристиками та надає методи для розрахунку ключових фізіологічних показників:

```
@dataclass
class User:
    name: str
    age: int
    gender: str
    weight: float  # кг
    height: float  # см
    fitness_level: str
    goals: List[str]
```

```
    limitations: List[str] = field(default_factory=list)
    available_equipment: List[str] = field(default_factory=list)
    target_weight: Optional[float] = None

    def body_mass_index(self) -> float:
        """Розрахунок індексу маси тіла"""
        return self.weight / ((self.height / 100) ** 2)

    def basal_metabolic_rate(self) -> float:
        if self.gender.lower() == "чоловіча":
            return 10 * self.weight + 6.25 * self.height - 5 * self.age + 5
        else:
            return 10 * self.weight + 6.25 * self.height - 5 * self.age – 161.
```

Методи body_mass_index() та basal_metabolic_rate() реалізують стандартні формули з фізіології спорту для розрахунку індексу маси тіла та базового метаболізму відповідно. Ці показники є критично важливими для генерації персоналізованих тренувальних планів та дієтичних рекомендацій.

Клас Exercise представляє фізичну вправу з усіма необхідними характеристиками для ефективного планування тренувань:

```
@dataclass
class Exercise:
    name: str
    type: str
    difficulty: int
    muscle_groups: List[str]
    calories_per_minute: float
    description: str = ""
    equipment: List[str] = field(default_factory=list).
```

Цей клас містить всю необхідну інформацію для складних алгоритмів вибору вправ. Поле difficulty дозволяє фільтрувати вправи за рівнем

складності відповідно до підготовки користувача. Список muscle_groups забезпечує можливість створювати збалансовані тренування, що працюють з різними групами м'язів. Параметр calories_per_minute використовується для розрахунку енергетичних витрат тренування.

Управління базою даних вправ реалізовано через клас ExerciseDatabase, який забезпечує ефективний пошук та фільтрацію вправ за різними критеріями:

```
class ExerciseDatabase:
    def __init__(self, file_path: str = "вправи.json"):
        self.file_path = file_path
        self.exercises: List[Exercise] = []
        self.load()

    def find_exercises(self, type: Optional[str] = None,
                       muscle_group: Optional[str] = None,
                       equipment: Optional[str] = None,
                       max_difficulty: Optional[int] = None) -> List[Exercise]:
        """Пошук вправ за параметрами"""
        result = self.exercises
        if type:
            result = [e for e in result if e.type.lower() == type.lower()]
        if muscle_group:
            result = [e for e in result if muscle_group.lower() in [g.lower() for g in e.muscle_groups]]
        if equipment:
            result = [e for e in result if equipment.lower() in [eq.lower() for eq in e.equipment]]
        if max_difficulty:
            result = [e for e in result if e.difficulty <= max_difficulty]
            return result.
```

Цей клас демонструє ефективне використання list comprehensions Python для фільтрації даних. Метод find_exercises дозволяє знаходити вправи за комбінацією різних критеріїв, що є критично важливим для алгоритмів генерації тренувальних планів. Використання Optional типів забезпечує гнучкість пошуку - будь-який з параметрів може бути None, що означає відсутність фільтрації за цим критерієм.

Найскладнішою частиною системи є модуль генерації тренувальних планів, реалізований у класі WorkoutGenerator. Цей клас містить складні алгоритми, які враховують безліч факторів для створення оптимальних персоналізованих програм тренувань:

```
def generate_plan(self, user: User, start_date: datetime.date,
                  end_date: datetime.date,
                  custom_workouts_per_week: int = None) -> WorkoutPlan:
    """Генерація плану тренувань з адаптивним навантаженням"""

    # Розрахунок тривалості плану
    days_duration = (end_date - start_date).days + 1
    weeks_duration = days_duration // 7 + (1 if days_duration % 7 > 0 else 0)

    # Визначення максимальної складності вправ
    max_difficulty = self._select_difficulty_level(user.fitness_level)

    # Генерація тренувань для кожного дня
    workouts = []
    current_date = start_date

    while current_date <= end_date:
        current_week = (current_date - start_date).days // 7
        day_of_week = current_date.weekday()
```

```
        # Створення тренування для поточного дня
        workout = self._create_workout(
            day_of_week=day_of_week,
            user=user,
            max_difficulty=max_difficulty,
            week_number=current_week + 1,
            total_weeks=weeks_duration
        )

        if workout:
            workouts.append(workout)

        current_date += datetime.timedelta(days=1)

    return WorkoutPlan(user=user, workouts=workouts,
                       start_date=start_date, end_date=end_date).
```

Цей метод демонструє комплексний підхід до планування тренувань, враховуючи прогресію навантаження протягом тижнів та індивідуальні особливості користувача. Алгоритм автоматично визначає оптимальну складність вправ на основі рівня підготовки користувача та створює структурований план з поступовим збільшенням навантаження.

Безпека користувацьких даних забезпечується через клас AccountManager, який реалізує надійні механізми автентифікації та авторизації:

```
def hash_password(self, password: str) -> str:
    """Хешування паролю для безпечного зберігання"""
    return hashlib.sha256(password.encode('utf-8')).hexdigest()

def register_user(self, username: str, password: str) -> Tuple[bool, str]:
    """Реєстрація нового користувача"""
```

```
    if username in self.accounts:
        return False, "Користувач з таким ім'ям вже існує"

    if len(username) < 3:
        return False, "Ім'я користувача повинно містити не менше 3 символів"

    if len(password) < 4:
        return False, "Пароль повинен містити не менше 4 символів"

    self.accounts[username] = {
        "password_hash": self.hash_password(password),
        "profile_id": None
    }

    self.save()
    return True, "Користувач успішно зареєстрований".
```

Окрім показаних основних методів, клас AccountManager також містить функціональність для авторизації існуючих користувачів через метод login_user, який перевіряє відповідність введеного паролю збереженому хешу. Додатково реалізовані методи для управління профілями користувачів, включаючи get_profile_id для отримання ідентифікатора профілю та set_profile_id для його встановлення після створення профілю користувача.

Клас ExerciseDatabase, окрім показаного методу пошуку, включає функціональність завантаження вправ з JSON-файлу через метод load, який обробляє серіалізовані дані та перетворює їх у об'єкти Exercise. Метод save забезпечує збереження оновленої бази даних вправ назад у JSON-формат. Важливою особливістю є метод add_exercise, який дозволяє користувачам додавати власні вправи до системи, автоматично зберігаючи оновлену базу даних. У випадку відсутності файлу бази даних, метод

_create_standard_exercises автоматично створює початковий набір вправ, що забезпечує працездатність системи навіть при першому запуску.

Модуль WorkoutGenerator містить найскладнішу логіку системи, включаючи приватні методи для створення окремих тренувань. Метод _create_workout аналізує параметри користувача та генерує індивідуальне тренування, враховуючи тип фокусу (верхня частина тіла, нижня частина, кардіо, все тіло), інтенсивність та прогресію по тижнях. Система включає складні алгоритми фільтрації вправ за обмеженнями користувача через метод _filter_exercises_by_limitations_equipment, який виключає вправи, що можуть бути небезпечними при певних медичних обмеженнях або недоступні через відсутність необхідного обладнання.

Для аналізу ефективності тренувань реалізовано метод analyze_progress, який обчислює статистику виконання плану, включаючи відсоток завершених тренувань, загальні витрати калорій та оцінку прогресу користувача. Метод recommend_diet генерує персоналізовані дієтичні рекомендації на основі фізичних параметрів користувача, його цілей та тривалості тренувального плану, розраховуючи необхідний калорійний баланс та розподіл макронутрієнтів.

Система управління профілями користувачів реалізована через клас UserProfileManager, який забезпечує серіалізацію та десеріалізацію складних об'єктів користувачів у JSON-формат. Цей клас включає методи для збереження повної інформації про користувача, включаючи його особисті дані, тренувальні плани та історію виконання через метод save_user_and_plan. Метод load_profile здійснює зворотну операцію, відновлюючи об'єкти Python з JSON-даних та правильно реконструюючи всі взаємозв'язки між компонентами.

Графічний інтерфейс користувача, реалізований у класі FitnessPlannerGUI, координує роботу всіх компонентів системи та забезпечує зручну взаємодію з користувачем. Клас містить методи для створення різних вкладок інтерфейсу, включаючи вкладку створення профілю через

create_profile_tab, вкладку планування тренувань через create_workouts_tab, та вкладку аналізу прогресу. Система автоматично оновлює інтерфейс при зміні стану через метод update_interface, забезпечуючи актуальність відображуваної інформації.

Важливою особливістю реалізації є система обробки подій, яка забезпечує плавну взаємодію між різними компонентами. Методи mark_workout_completed та show_next_day дозволяють користувачам відмічати виконані тренування та навігувати по календарю тренувань. Система автоматично зберігає всі зміни через виклики відповідних методів збереження, гарантуючи, що дані користувача не будуть втрачені.

Модуль авторизації LoginRegisterWindow реалізовує окреме вікно для входу та реєстрації користувачів з використанням сигналів та слотів Qt для забезпечення взаємодії з основним вікном програми. Цей підхід забезпечує безпечну передачу інформації про успішну авторизацію без прямих залежностей між компонентами.

Реалізована система забезпечує повну функціональність програмного модуля індивідуального планування спортивних тренувань, ефективно поєднуючи складні алгоритми генерації планів з зручним графічним інтерфейсом та надійними механізмами зберігання даних. Модульна архітектура дозволяє легко розширювати функціональність системи та додавати нові можливості без порушення існуючої логіки.

3.3 Тестування роботи програмного модулю індивідуального планування спортивних тренувань

Під час тестування програмного модуля індивідуального планування спортивних тренувань потрібно було перевірити коректність введення та відображення даних, систему авторизації та роботу кожного окремого модуля системи. Тестування проводилося поетапно з перевіркою всіх основних функціональних можливостей програми.

Для того, щоб запустити програмний модуль, необхідно мати встановлений Python версії 3.7 або вище та бібліотеку PyQt5. Запуск програми здійснюється за допомогою команди python main.py в терміналі або запуск «main.py» через будь-яке середовище розробки Python, наприклад PyCharm.

Після запуску програми користувач попадає на стартове вікно авторизації, у якому він може увійти в систему або зареєструвати новий обліковий запис. На рисунку 3.1 показано головне вікно авторизації програми з можливістю вибору між входом та реєстрацією.

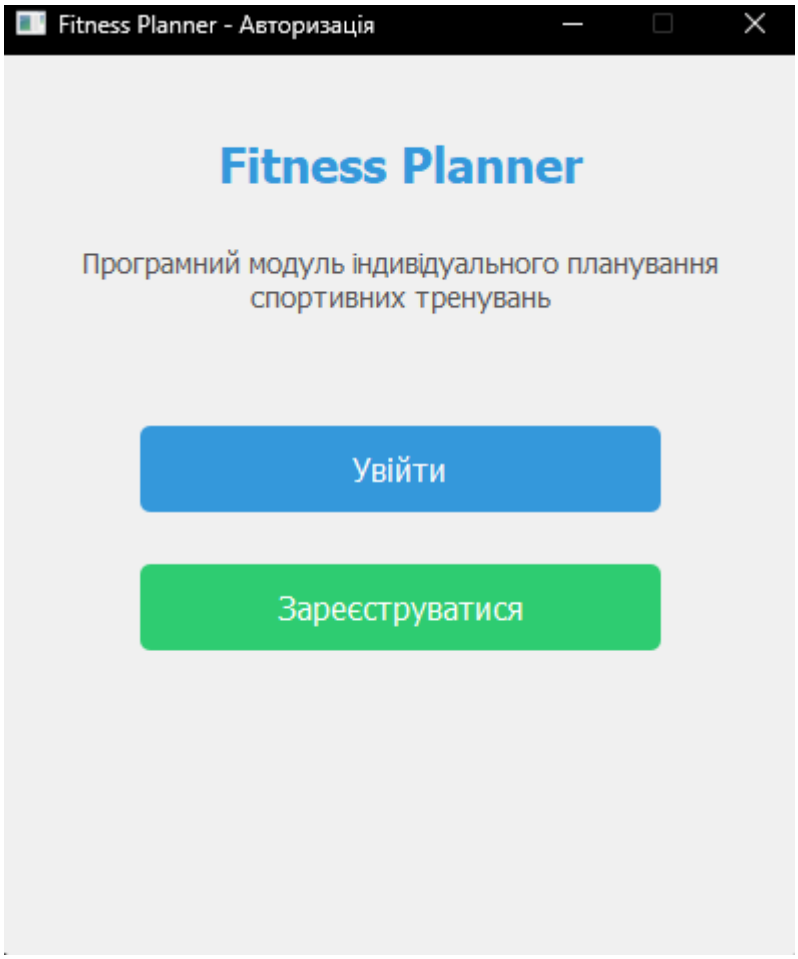

Рисунок 3.1 – Загальний вигляд інтерфейсу головного вікна авторизації до системи

Для прикладу тестування введемо ім'я користувача "tester" та довільний пароль "123". Як ми можемо побачити на рисунку 3.2, система відображає

повідомлення "Користувач не знайдений" та не дозволяє ввійти користувачеві до основного функціоналу.

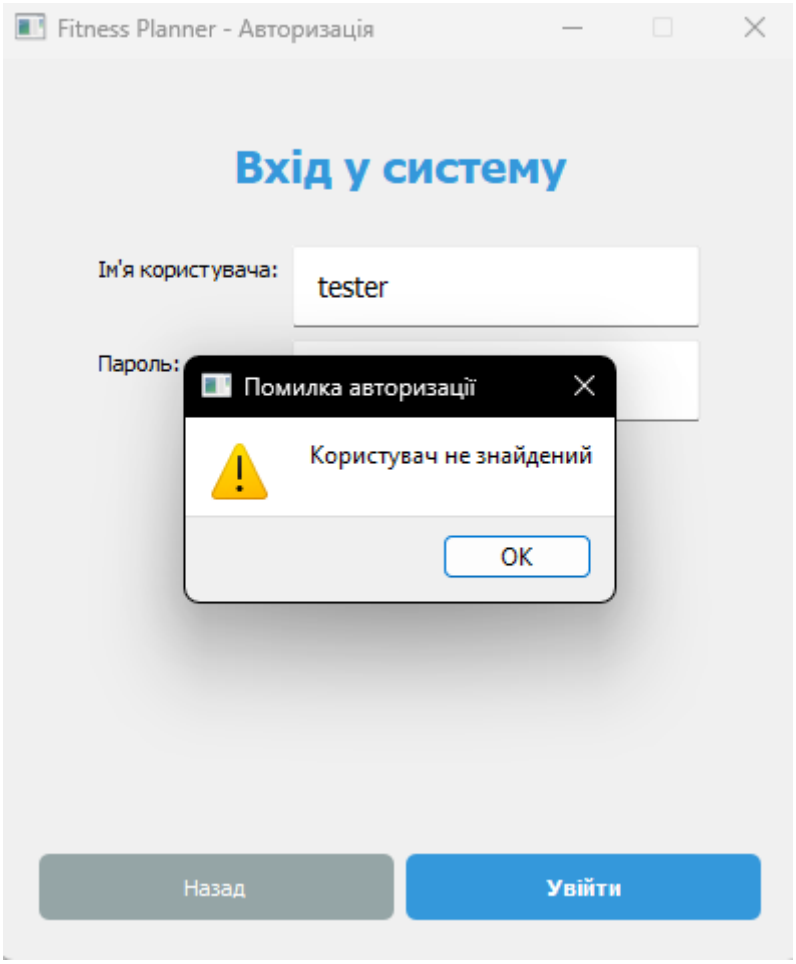

Рисунок 3.2 – Загальний вигляд повідомлення про помилку при невірних даних авторизації

Для того, щоб коректно ввійти в систему потрібно зареєструватися. Для цього ннатискаємо кнопку "Зареєструватися" та заповнюємо форму реєстрації. При спробі створити обліковий запис з іменем " tester " та паролем "123". Але, як ми бачимо, система відображає повідомлення "Пароль повинен містити не менше 4 символів", як показано на рисунку 3.3.

Вводимо новий пароль, наприклад "1231" та успішно реєструємо нового користувача. Система підтверджує успішну реєстрацію та автоматично

переводить на сторінку вход. Вводимо щойно створені облікові дані та потрапляємо до головного вікна програми.

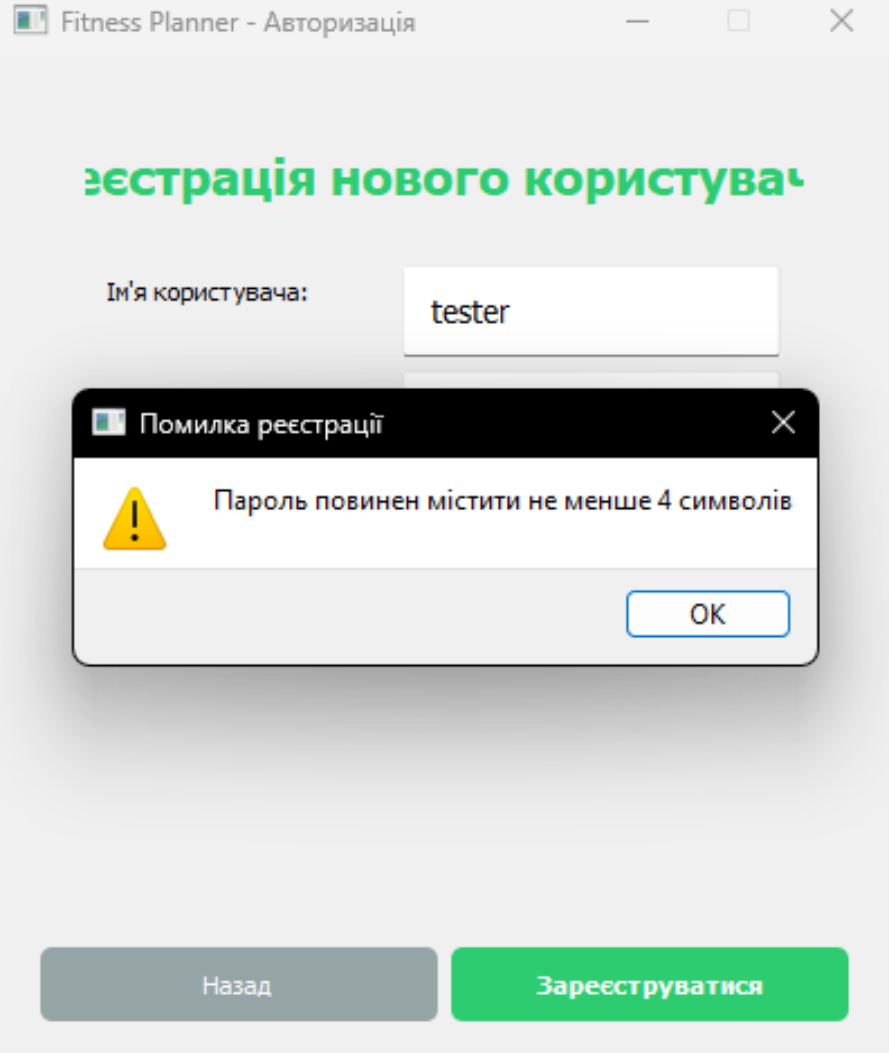

Рисунок 3.3 – Загальний вигляд повідомлення про недостатню довжину паролю

Після успішної авторизації нас зустрічає головне вікно програми з двума вкладками «Вітання» та «Профіль». Оскільки профіль ще не створено, більшість функцій для користувача недоступні. На рисунку 3.4 показано головне вікно з інформацією про необхідність створення профілю користувача.

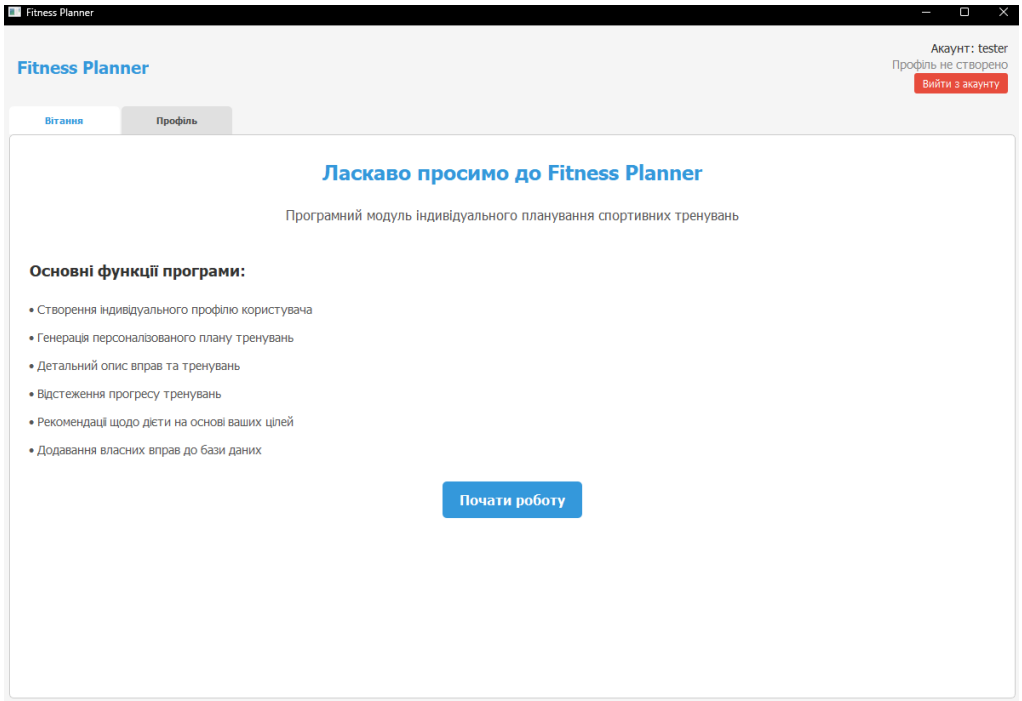

Рисунок 3.4 – Загальний вигляд інтерфейсу головного вікна програми після авторизації

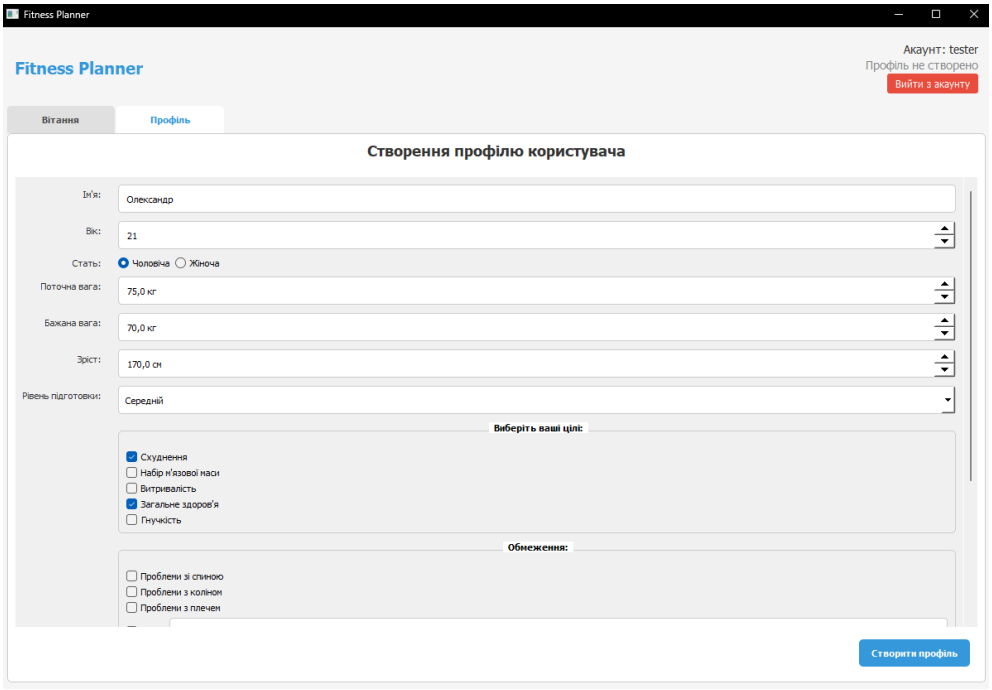

Рисунок 3.5 – Загальний вигляд інтерфейсу вкладки «Профіль», яка відповідає за створення профілю користувача

Переходимо до вкладки "Профіль" для створення нового профілю користувача. Заповнюємо основні дані: вказуємо ім'я "Олександр", вік 21 рік, стать "чоловіча", вага 75 кг, бажана вага 70 кг та зріст 180 см. Вибираємо рівень підготовки "середній" та цілі тренувань "схуднення" і "загальне здоров'я". На рисунку 3.5 показано заповнену форму створення профілю.

Якщо ж ми створимо профіль без вибору цілей тренувань, система відобразить повідомлення "Будь ласка, виберіть хоча б одну ціль".

Після заповнення всіх необхідних полів та натискання кнопки "Створити профіль" система успішно створює профіль та відображає повідомлення з розрахованими фізіологічними показниками. На рисунку 3.6 бачимо підтвердження створення профілю з ІМТ 26 та базовим метаболізмом 1712 ккал/день.

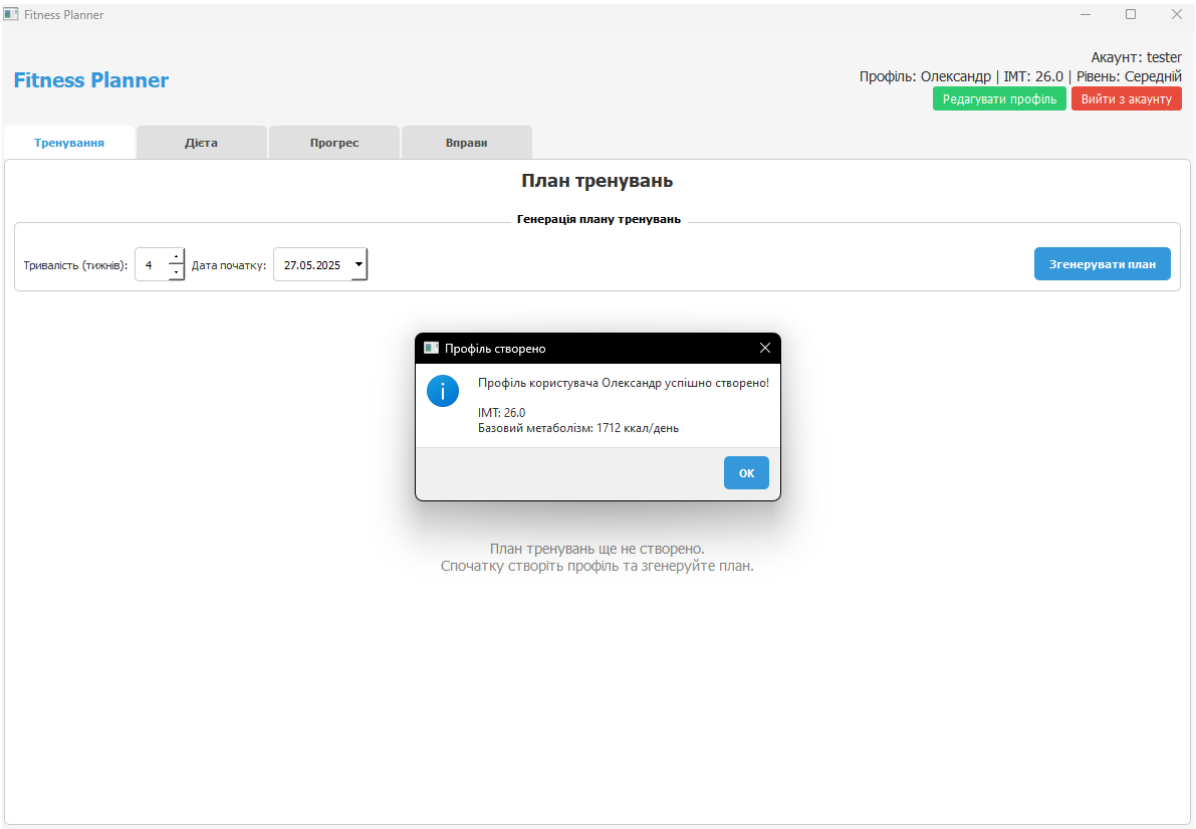

Рисунок 3.6 – Загальний вигляд повідомлення про успішне створення профілю з розрахованими показниками

Тепер переходимо до генерації тренувального плану. У вкладці "Тренування" налаштовуємо параметри плану: встановлюємо тривалість 5 тижнів та дату початку на поточний день. На рисунку 3.7 показано інтерфейс налаштування параметрів тренувального плану.

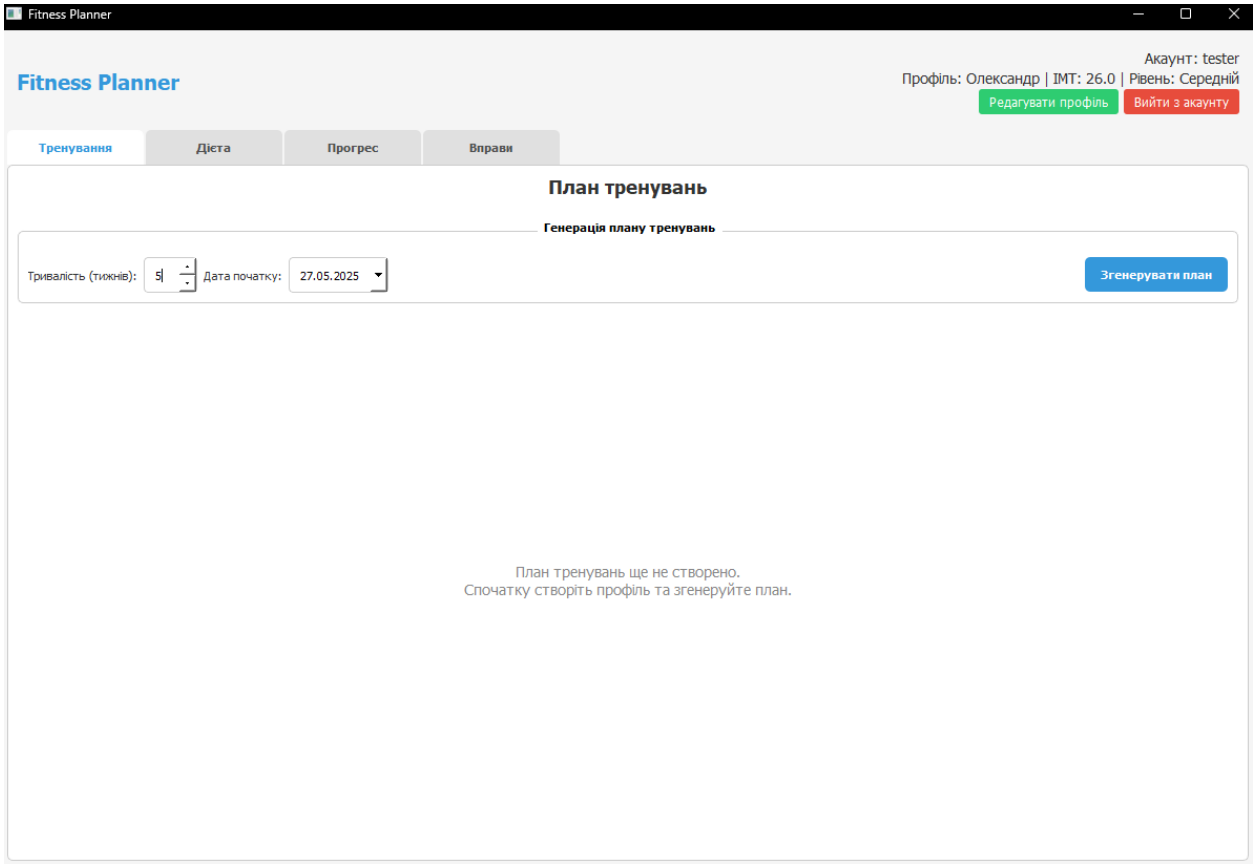

Рисунок 3.7 – Загальний вигляд інтерфейсу вкладки «Тренування» до генерації індивідуального плану тренування

Натискаємо кнопку "Згенерувати план" і система створює персоналізований план тренувань. На екрані з’явиться діалогове вікно з інформацією про згенерований план, включаючи тривалість, кількість тренувань на тиждень, рівень інтенсивності та можлививми порадами і попередженнями. На рисунку 3.8 показано повідомлення про успішну генерацію плану тренувань.

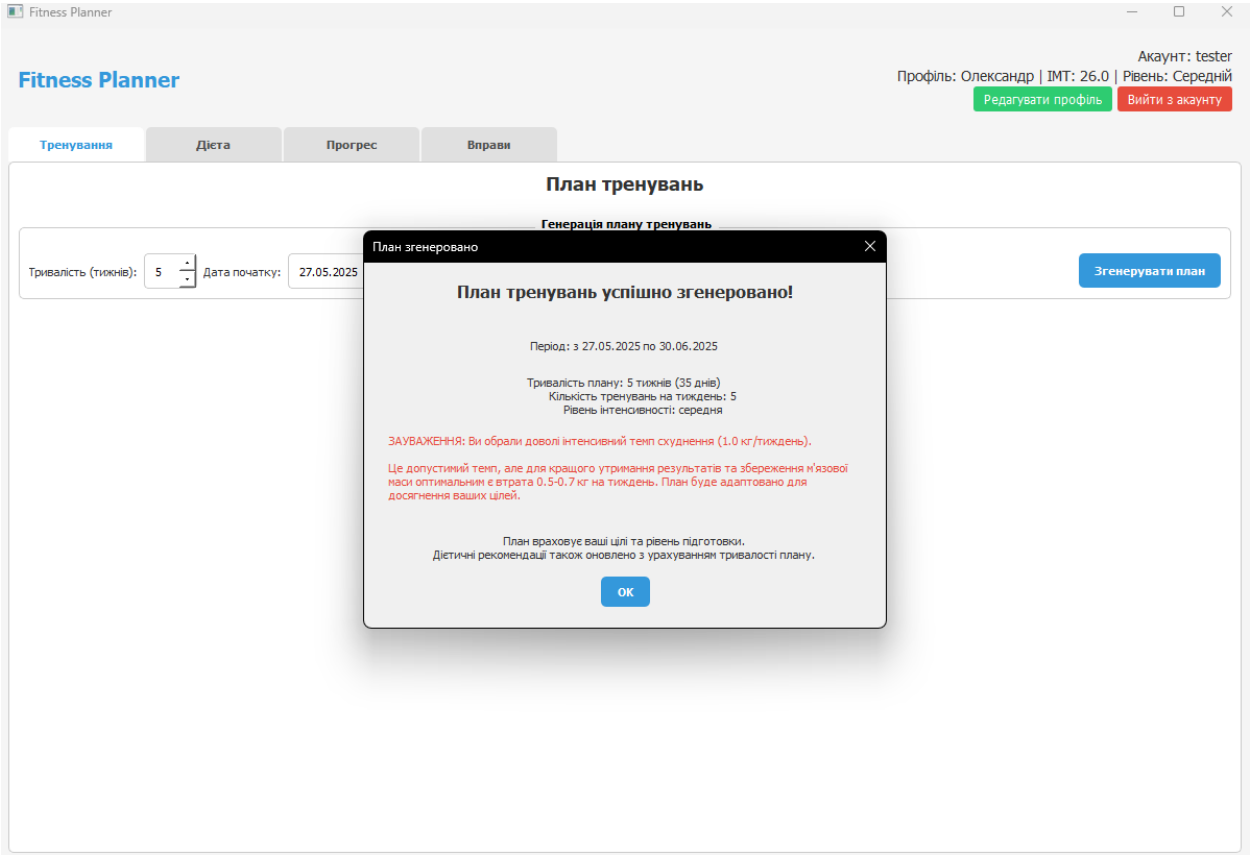

Рисунок 3.8 – Загальний вигляд повідомлення про успішну генерацію плану тренувань

Після натискання «Ок» в діалоговому вікні в вікні основної програми ми можемо побачити згенерований детальний план тренування на поточний день. Система відображає назву тренування, інтенсивність, тривалість та список вправ з кількістю підходів і повторень. На рисунку 3.9 показано згенерований план тренувань з деталізованою інформацією про вправи.

Після виконання відповідного плану на день, та виведення плану тренувань на наступний, необхідно проставити відмітку «Виконано». Для цього натискаємо кнопку "Відмітити тренування як виконане" і система підтверджує виконання тренування. На рисунку 3.10 показано підтвердження виконання тренування та оновлений інтерфейс з позначкою про завершення.

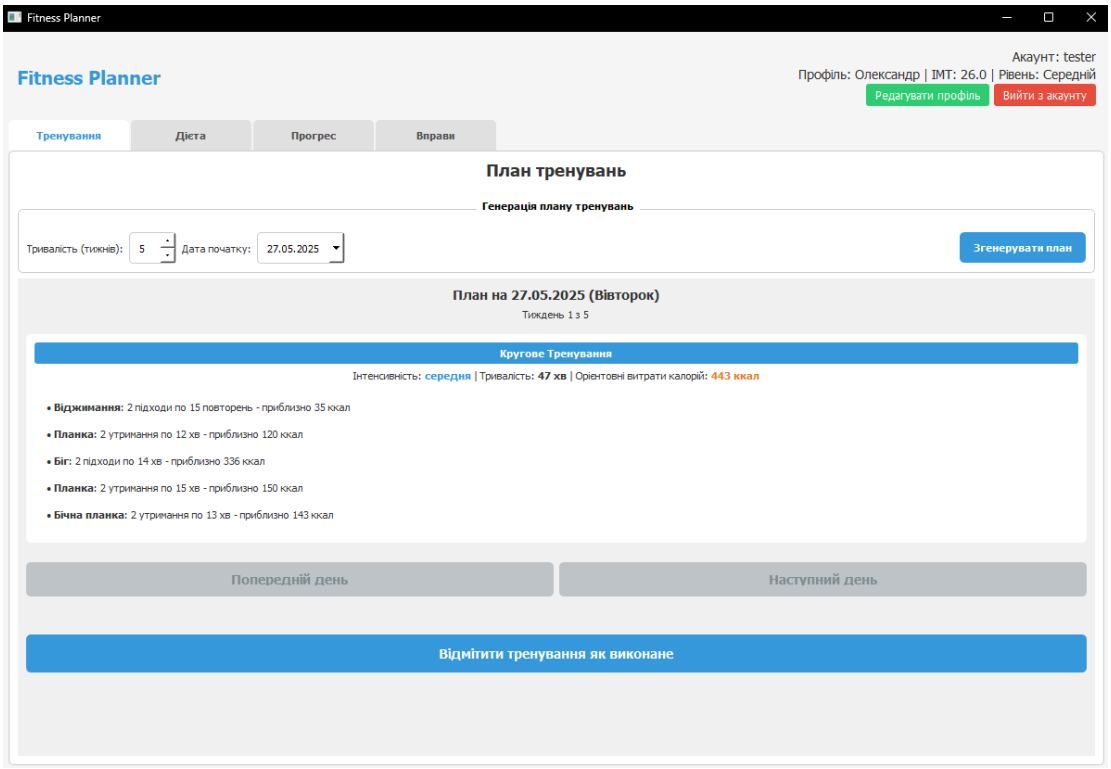

Рисунок 3.9 – Загальний вигляд інтерфейсу вкладки «Тренування» після генерації індивідуального плану тренування на поточний день

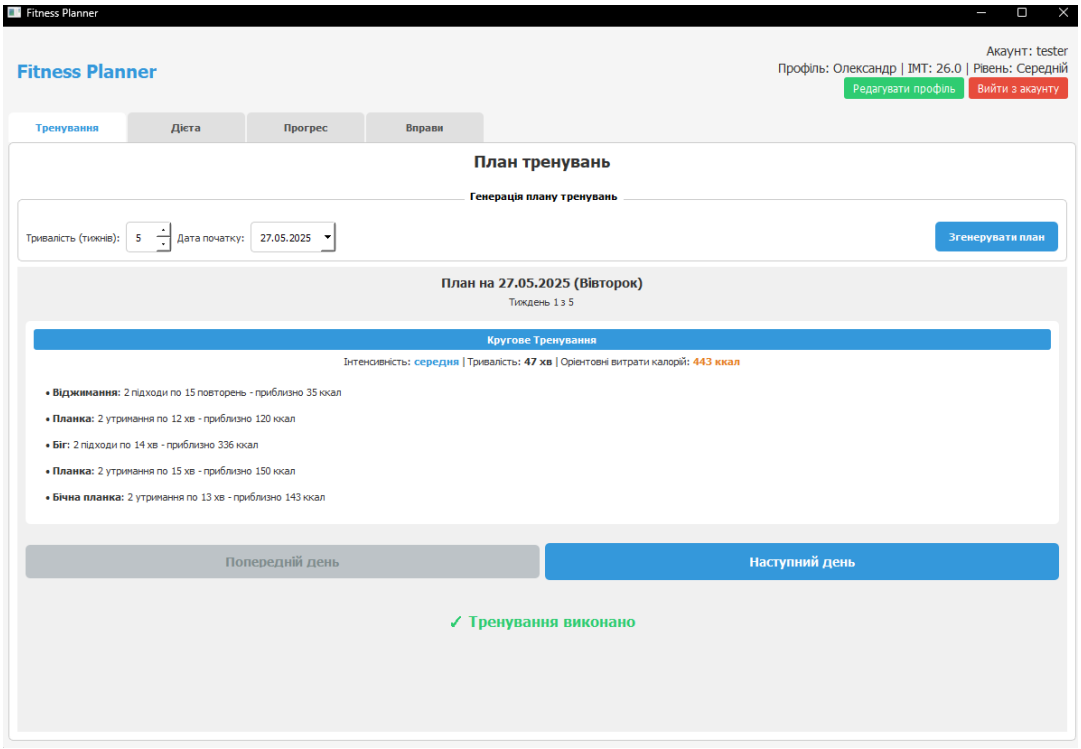

Рисунок 3.10 – Загальний вигляд інтерфейсу вкладки «Тренування» після відмітки виконаного тренування на поточний день

Переходимо до наступного дня за допомогою кнопки "Наступний день" та бачимо новий план тренування. Система автоматично адаптує програму відповідно до прогресії навантаження. На рисунку 3.11 показано план тренування на другий день з іншим набором вправ.

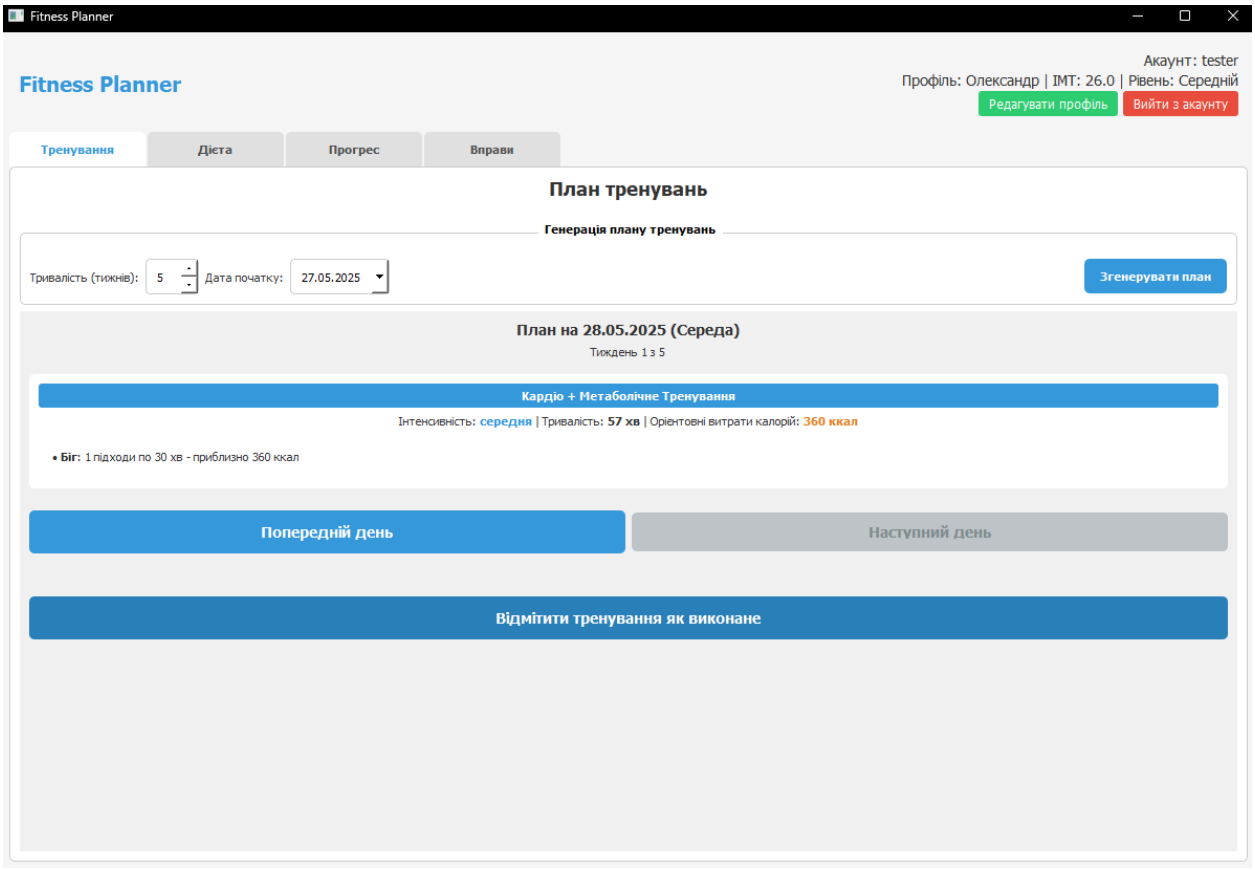

Рисунок 3.11 – Загальний вигляд інтерфейсу вкладки «Тренування» на наступний день плану тренувань

Після виконання декількох днів тренувань за згенерованим планом, ми можемо переглянути наш прогрес у вкладці «Прогрес». У вкладці відображено детальну статистику, включаючи загальну кількість тренувань, відсоток виконання, витрати калорій та загальну оцінку прогресу. На рисунку 3.12 показано статистику прогресу тренувань з візуальним прогрес-баром.

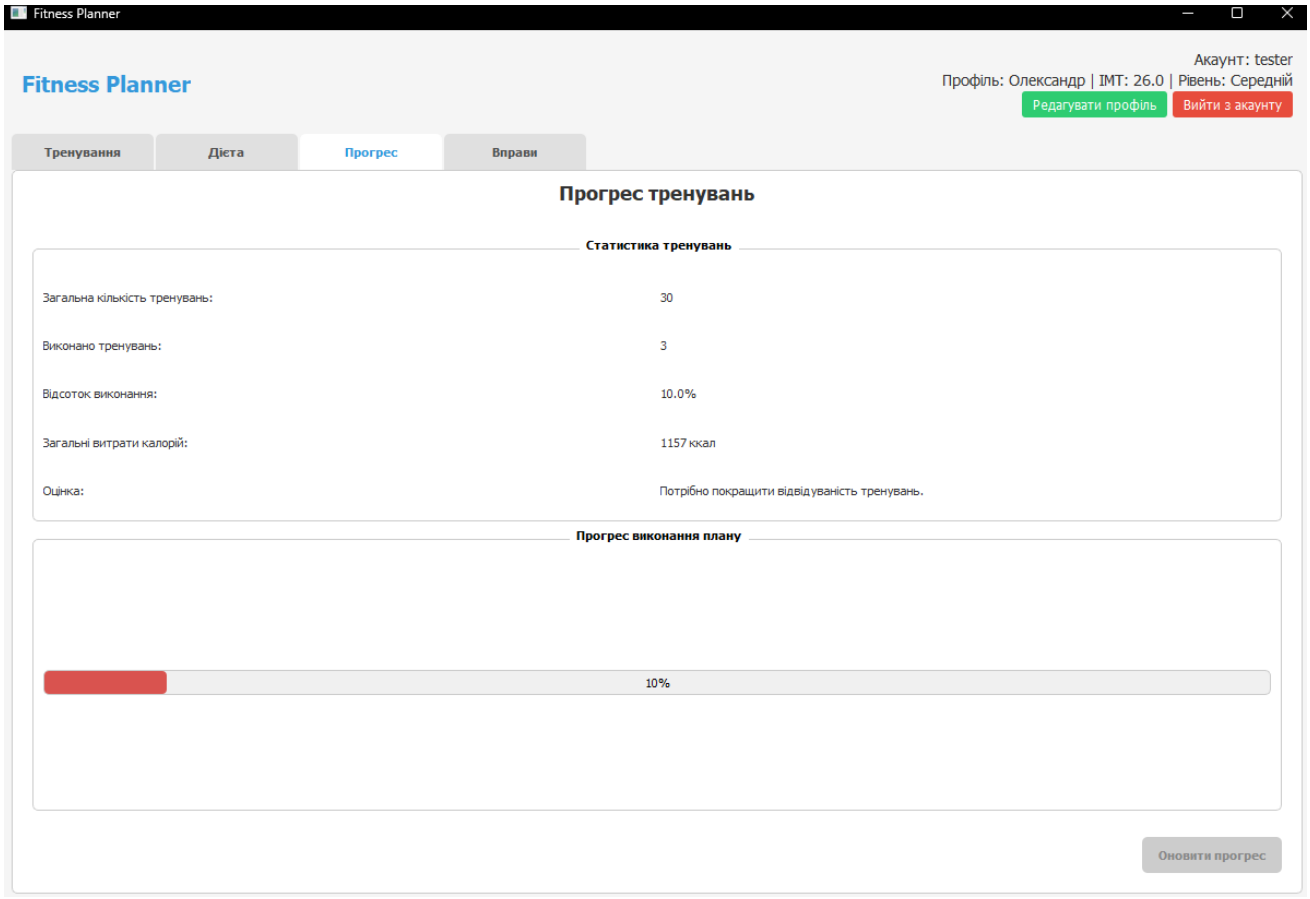

Рисунок 3.12 – Загальний вигляд інтерфейсу вкладки «Прогрес»

Переходимо до вкладки «Дієта», де система автоматично розраховує поради з харчування на основі створеного профілю. У цій вкладці ми можемо побачити розрахунки базового метаболізму, рекомендованої денної норми калорій, ІМТ та детальний розподіл нутрієнтів(білки, жири, вуглеводи). Також програма відображає «План досягнення бажаної ваги», де вказана рекомендована швидкістю схуднення та час для досягнення бажаної ваги без шкоди для вашого здоров’я. На рисунку 3.13 показано повні дієтичні рекомендації з розбором білків, жирів та вуглеводів.

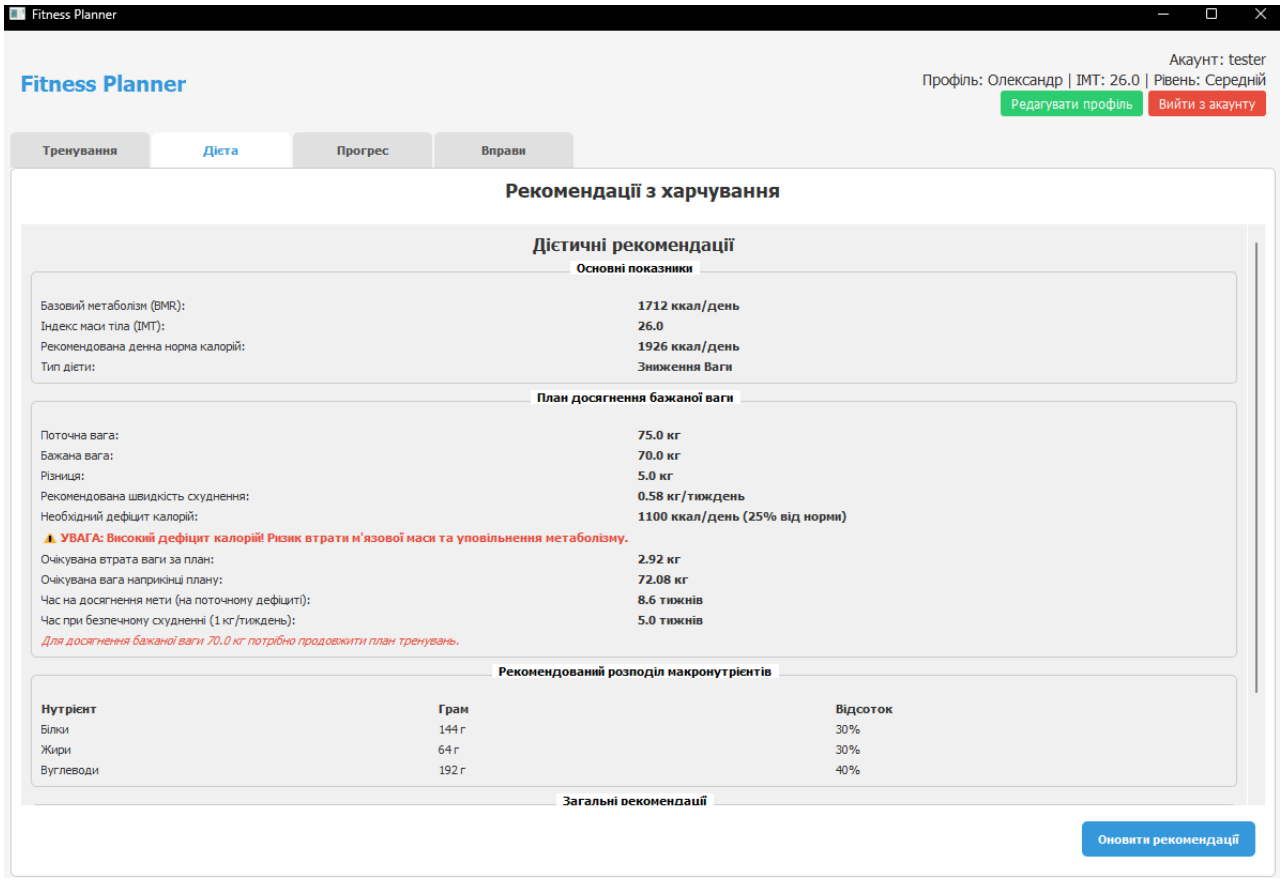

Рисунок 3.13 – Загальний вигляд інтерфейсу вкладки «Дієта»

У вкладці "Вправи" ми можемо переглянути базу даних вправ програми. Система дозволяє фільтрувати вправи за типом, групою м'язів та складністю. На рисунку 3.14 показано інтерфейс перегляду вправ з можливістю детального огляду кожної вправи.

Також при необіхності ми можемо добавити потрібні нам нові вправи в базу даних програми натисканням кнопки «Додати нову вправу». Наприклад, додамо вправу з назвою «Підтягування на турніку», вибираємо тип «силова», складність «4», групи м'язів «спина» і «біцепс», Кількість калоріїв на хвилину 2 ккал/хв та необхідне обладнання «Турнік/Бруси». На рисунку 3.15 показано заповнену форму додавання нової вправи.

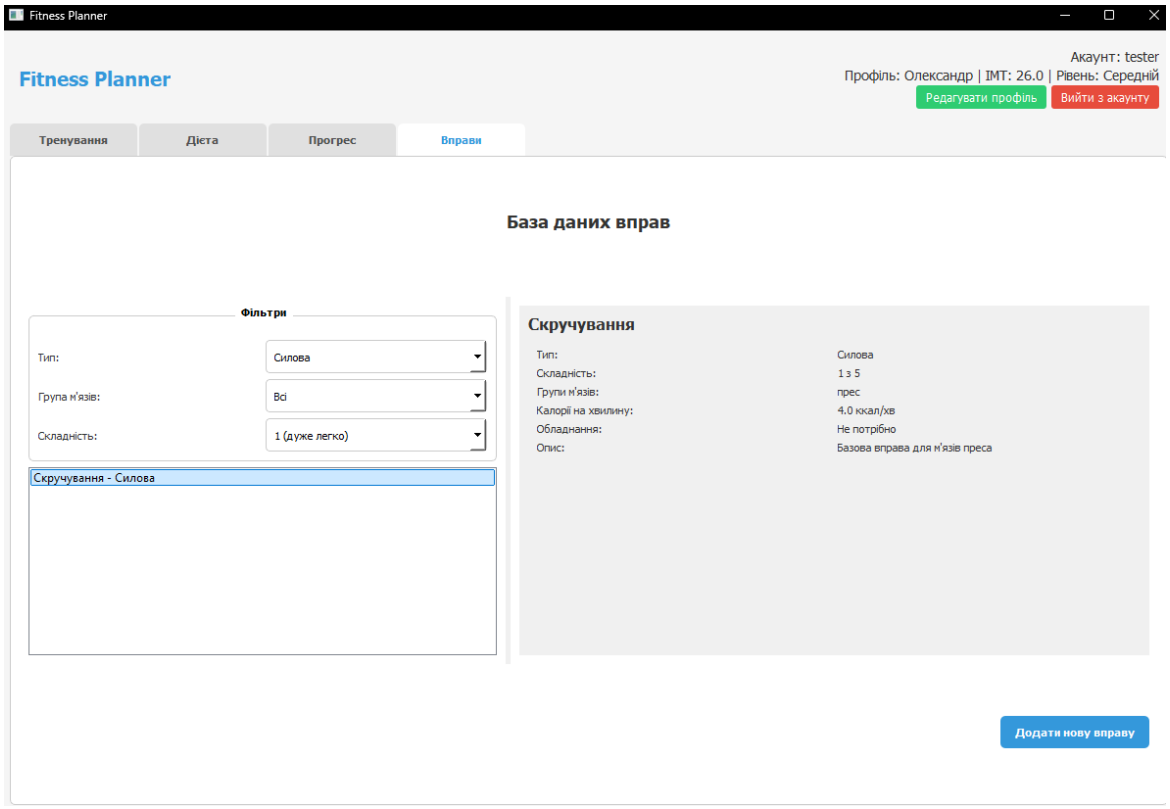

Рисунок 3.14 – Загальний вигляд інтерфейсу вкладки «Вправи»

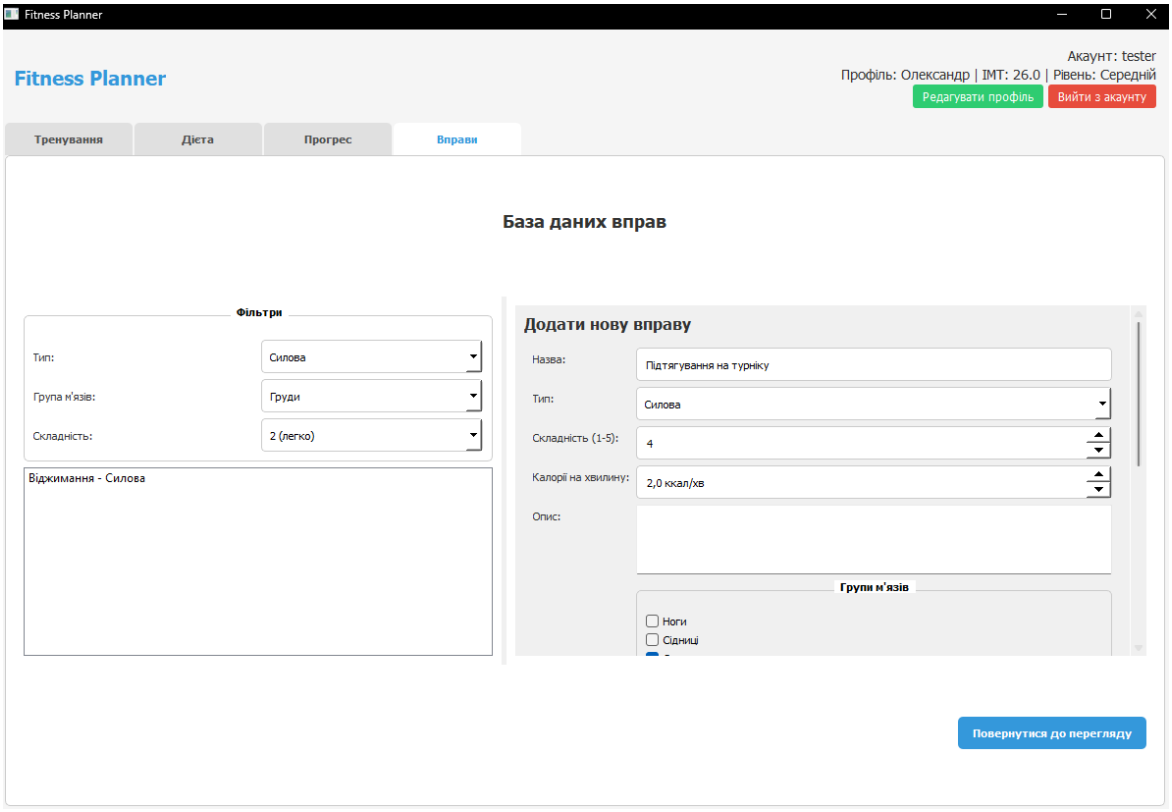

Рисунок 3.15 – Загальний вигляд форми додавання нової вправи до бази даних

Після натискання кнопки "Додати вправу" система підтверджує успішне додавання та оновлює список доступних вправ. Нова вправа тепер буде доступна для використання в майбутніх тренувальних планах.

Також при необхідності зміни інформації в профілі користувача можна відредагувати профіль і внести відповідні зміни. Для цього в любій вкладці необхідно справа зверху натиснути зелену кнопку «Редагувати профіль». Після чого відкривається нове вікно, в якому ми можемо редагувати профіль користувача, змінюючи необхідні дані, зберегти зміни або видалити профіль. На рисунку 3.16 зображено загальний вигляд вікна редагування профілю користувача.

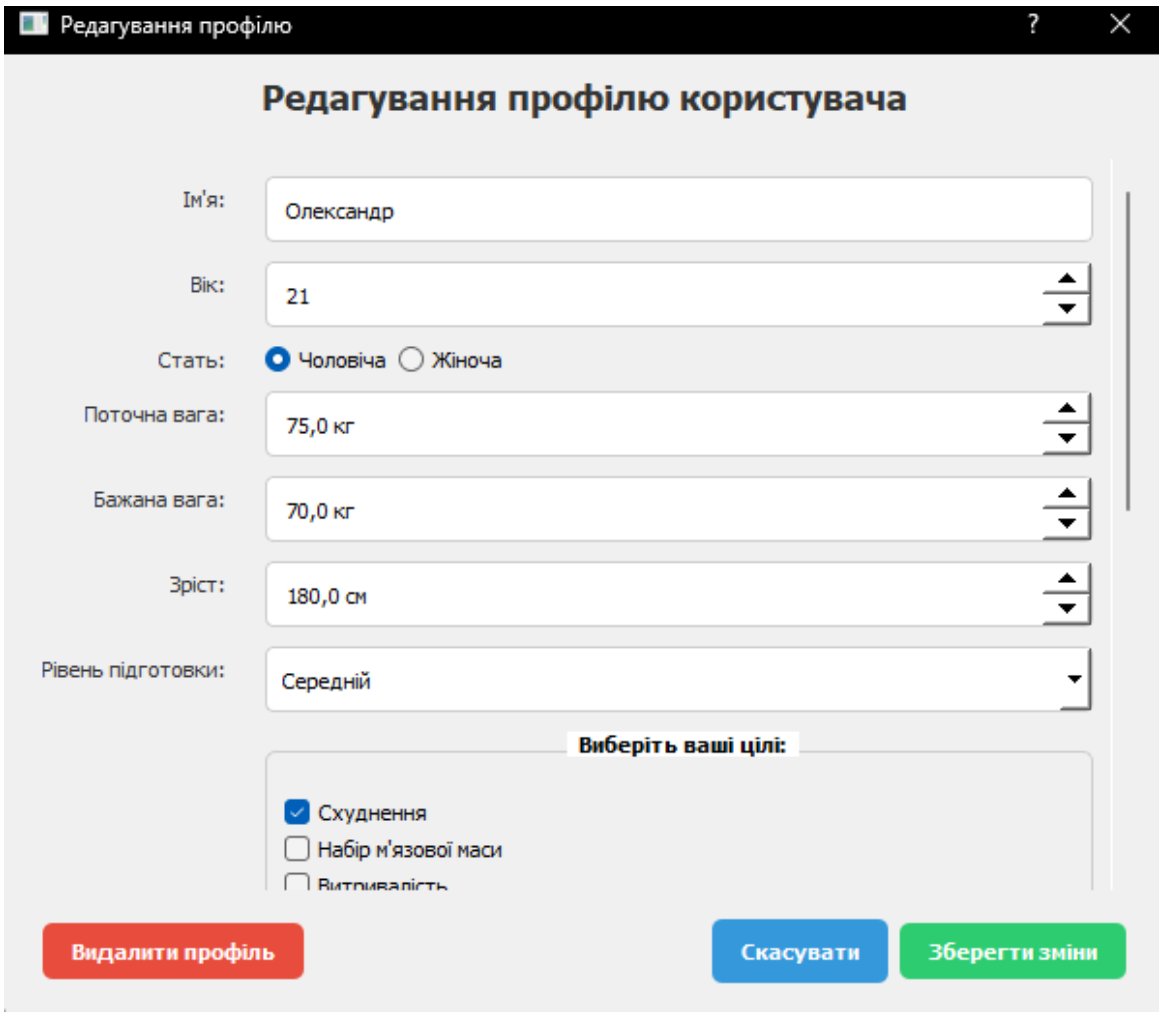

Рисунок 3.16 – Загальний вигляд вікна редагування профілю користувача

Тестування програмного модулю підтвердило коректність роботи всіх модулів програмної системи. Система демонструє стабільну роботу при різних сценаріях використання, правильно та планово обробляє помилки введення даних та забезпечує зберігання інформації користувачів. Алгоритми генерації тренувальних планів показали високу ефективність у створенні персоналізованих програм. Тестування підтвердило готовність програмного модуля до практичного використання та його здатність ефективно вирішувати завдання індивідуального планування спортивних тренувань.

3.4 Висновок до розділу 3

У третьому розділі було здійснено програмну реалізацію модуля індивідуального планування спортивних тренувань та проведено комплексне тестування всіх його компонентів.

На етапі обґрунтування вибору мови програмування було проведено порівняльний аналіз між Python та Java як основними альтернативами для реалізації системи. У результаті аналізу переваг та недоліків обох мов було прийнято рішення про використання Python. Ключовими факторами вибору стали простота синтаксису Python, що прискорює розробку, повна підтримка об'єктно-орієнтованого програмування, наявність потужної бібліотеки PyQt5 для створення графічного інтерфейсу та вбудована підтримка JSON-формату.

Програмна реалізація модуля була виконана відповідно до спроєктованої архітектури з чітким розділенням функціональності між сімома основними файлами. Головний модуль «main.py» забезпечує запуск системи, «models.py» реалізує класи предметної області з методами розрахунку фізіологічних показників, «database.py» управляє базою даних вправ, «generator.py», який містить складні алгоритми генерації персоналізованих тренувальних планів, «account_manager.py», що забезпечує безпечне управління обліковими записами, «login_register.py», який реалізовує

інтерфейс авторизації, а «gui.py» координує роботу всіх компонентів через графічний інтерфейс користувача.

Комплексне тестування програмного модуля підтвердило коректність роботи всіх реалізованих функцій. Тестування охопило всі основні сценарії використання системи, від процесу реєстрації та створення профілю до генерації тренувальних планів та відстеження прогресу. Система показує стабільну роботу при різних комбінаціях вхідних даних, правильно обробляє помилкові ситуації та забезпечує зберігання інформації користувачів.

Результати тестування підтвердили ефективність алгоритмів генерації персоналізованих тренувальних програм, точність розрахунків фізіологічних показників та дієтичних рекомендацій, а також зручність графічного інтерфейсу користувача.

ВИСНОВКИ

Поставлені перед бакалаврською кваліфікаційною роботою задачі виконано в повному обсязі, а саме:

1) Проведено аналіз предметної області індивідуального планування спортивних тренувань.

2) Проведено аналіз існуючих методів, алгоритмів та програмних реалізацій для індивідуального планування спортивних тренувань.

3) Розроблено архітектуру програмного модуля індивідуального планування спортивних тренувань.

4) Розроблено взаємодію програмних модулів системи та базу даних програмного модуля.

5) Здійснено програмну реалізацію модуля індивідуального планування спортивних тренувань.

6) Протестовано програмний модуль індивідуального планування спортивних тренувань.

7) Розроблено інструкції користувача для користування програмнии модулем індивідуального планування спортивних тренувань.

У першому розділі було досліджено ключові аспекти індивідуального планування спортивних тренувань, включаючи основні принципи тренувального процесу. Проаналізовано існуючі методи та алгоритми для індивідуального планування тренувань, такі як методи визначення оптимального навантаження, алгоритми прогресії та підходи до періодизації. Розглянуто сучасні програмні засоби для планування тренувань, зокрема Fitbod, Jefit, Nike Training Club, MyFitnessPal та Strava, виявлено їх переваги та обмеження.

У другому розділі розроблено трирівневу архітектуру програмного модуля, що включає презентаційний рівень, рівень бізнес-логіки та рівень даних. Детально опрацьовано взаємодію між шістьма основними модулями системи та створено UML-діаграму класів. Розроблено об'єктно-орієнтовану

модель даних предметної області з використанням об'єктно-документного підходу та серіалізації у форматі JSON.

У третьому розділі було обґрунтовано вибір мови програмування Python та бібліотеки PyQt5 для створення графічного інтерфейсу. Виконано програмну реалізацію модуля з модульною структурою, що включає сім основних файлів з чітким розділенням функціональності. Проведено комплексне тестування програмного модуля, яке підтвердило коректність роботи всіх компонентів системи та готовність до практичного використання.

Тестування розробленого програмного модуля продемонструвало підвищення ефективності створення індивідуального плану тренувань на 6,8% порівняно з існуючими програмними рішеннями за рахунок більш детального врахування індивідуальних особливостей та медичних обмежень користувача, що дозволяє генерувати оптимальні тренувальні програми з мінімальними ризиками травмування та максимальною відповідністю заданим цілям.

Метою дослідження є підвищення ефективності індивідуального планування спортивних тренувань шляхом впровадження автоматизованих процесів створення персоналізованих тренувальних програм була досягнута. Розроблений програмний модуль має потенціал для широкого використання фітнес-ентузіастами, тренерами та спортсменами для оптимізації тренувального процесу та досягнення кращих результатів у фізичному розвитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Формування культури здоров'я засобами фізичного виховання [Електронний ресурс] – Режим доступу до ресурсу: https://naurok.com.ua/formuvannya-kulturi-zdorov-ya-zasobami-fizichnogo-vihovannya-447962.html.
2. Власок О.М. «Програмний модуль індивідуального планування спортивних тренувань» в Матеріалах конференції «LIV Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2025)», Вінниця, 2025. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/23708.
3. Лекція №3 Принципи спортивного тренування [Електронний ресурс] – Режим доступу до ресурсу: https://studfile.net/preview/10045003/.
4. Типи м'язових волокон та найкращі вправи для їх тренування [Електронний ресурс] – Режим доступу до ресурсу: https://ua.estet-portal.com/statyi/tipy-myshechnykh-volokon.
5. Принципи фітнес тренування [Електронний ресурс] – Режим доступу до ресурсу: https://www.fitnesservice.com.ua/статті-та-фото/принципи-фітнес-тренування.
6. Клопов Р. В., Сватьєв А. В., Клопова В. О. Програмування індивідуалізації тренувального процесу: принципи, чинники, психологічний аспект. С. 1–7 DOI: https://doi.org/10.26661/2663-5925-2024-2-02.
7. Індивідуальний підхід: як тренер прискорює ваш прогрес [Електронний ресурс] – Режим доступу до ресурсу: https://www.ilovesport.com.ua/individualnij-pidkhid-yak-trener-priskoryuye-vash-progres/.

8. Як правильно скласти програму тренувань [Електронний ресурс] – Режим доступу до ресурсу: https://winger.com.ua/ua/blog-ua/yak-pravilno-sklasti-programu-trenuvan/.
9. Що потрібно знати про стан вашого організму перед тим, як йти до спортзалу [Електронний ресурс] – Режим доступу до ресурсу: https://varosh.com.ua/adv/shho-potribno-znaty-pro-stan-vashogo-organizmu-pered-tym-yak-jty-do-sportzalu/.
10. Тренування на вулиці і в залі: в чому відмінність, плюси і мінуси тренувань на повітрі [Електронний ресурс] – Режим доступу до ресурсу: https://fitcurves.org/ua/blog/trenirovki-na-ulitse-i-v-zale/.
11. Як часто слід змінювати план тренування, щоб не знудитись, отримати результат та користь [Електронний ресурс] – Режим доступу до ресурсу: https://life.liga.net/poyasnennya/news/kak-chasto-nujno-menyat-plan-trenirovki-chtoby-ne-bylo-skuchno-i-poluchit-rezultat.
12. Як створити якісний план тренувань у спортзалі? [Електронний ресурс] – Режим доступу до ресурсу: https://gymbeam.ua/blog/uk/yak-stvorytu-yakisnuy-plan-trenuvany-u-sportzali/.
13. Software for Strength Coaches: Program Analytics [Електронний ресурс] – Режим доступу до ресурсу: https://blog.bridgeathletic.com/software-for-strength-coaches-program-analytics-.
14. The Power Of Subjective Data In Sports Science. Part 2 [Електронний ресурс] – Режим доступу до ресурсу: https://luminsports.com/media/94-the-power-of-subjective-data-in-sports-science-part-2.
15. 1RM (one-repetition maximum) testing [Електронний ресурс] – Режим доступу до ресурсу: https://www.scienceforsport.com/1rm-testing/?srsltid=AfmBOooDXHs8VlNG0v7ZQj2ivc8esPp1bAszATYA_HPEUG0PUmjRu9Kd.
16. Як розрахувати максимум одного повторення? [Електронний ресурс] – Режим доступу до ресурсу: https://uk.lifestyle.fit/Тренінг/чайові/як-розрахувати-макс/.

17. Фізичне виховання, спорт та здоров'я людини:досвід, проблеми, перспективи [Електронний ресурс] – Режим доступу до ресурсу: https://ic.ac.kharkov.ua/navchannya/hm/fkz/materials/nauk_gurt/zb_12.12.24.pdf.
18. Основні методи прогресування на тренуванні [Електронний ресурс] – Режим доступу до ресурсу: https://fitness-shop.ua/blog/statti/osnovni-metody-prohresuvannia-na-trenuvanni.
19. Як важкі силові тренування впливають рівень тестостерону [Електронний ресурс] – Режим доступу до ресурсу: https://ukr-med.net/yak-vazhki-sylovi-trenuvannya-vplyvayut-riven-testosteronu-chomu-bez-periodyzacziyi-mozhna-otrymaty-peretrenovanist/.
20. Як скласти ефективну програму тренувань? 8 кроків до успіху у фітнесі та історія одного хлопця [Електронний ресурс] – Режим доступу до ресурсу: https://e-trener.com/yak-sklasti-efektivnu-programu-trenuvan-8-krokiv-do-uspihu-u-fitnesi-ta-istoriya/.
21. Клопов Р. В., Сватьєв А. В., Клопова В. О. Індивідуалізація тренувального процесу: Переваги застосування інформаційних технологій с. 1–4 DOI: https://doi.org/10.36059/978-966-397-383-8-9.
22. Як регулювати навантаження у функціональному груповому тренінгу [Електронний ресурс] – Режим доступу до ресурсу: https://fitnessacademy.com.ua/articles/iak-rehuliuvaty-navantazhennia-u-funktsionalnomu-hrupovomu-treninhu/.
23. Принципи і техніка силових вправ [Електронний ресурс] – Режим доступу до ресурсу: https://interatletika.com.ua/blog/printsipy-i-tekhnika-silovykh-uprazhneniy/.
24. How Fitbod Creates Your Workout [Електронний ресурс] – Режим доступу до ресурсу: https://fitbod.zendesk.com/hc/en-us/articles/360004429814-How-Fitbod-Creates-Your-Workout.
25. Fitbod vs Jefit apps [Електронний ресурс] – Режим доступу до ресурсу: https://casadesante.com/blogs/fitness/fitbod-vs-jefit-apps.

26. Nike Training Club [Електронний ресурс] – Режим доступу до ресурсу: https://www.nike.com/ntc-app.
27. Огляд MyFitnessPal. [Електронний ресурс] – Режим доступу до ресурсу: https://icoola.ua/blog/my-fitness-pal-app.
28. Strava Features [Електронний ресурс] – Режим доступу до ресурсу: https://www.strava.com/features.
29. Архітектура програмного забезпечення: як правильне проєктування забезпечує ефективність та безпеку [Електронний ресурс] – Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/arhitektura-programnogo-zabezpechennya.
30. Переваги і недоліки мови Python [Електронний ресурс] – Режим доступу до ресурсу: https://blog.ithillel.ua/articles/perevagi-i-nedoliki-movi-python.
31. Все, що ви маєте знати про Python. Які в нього недоліки, а які переваги? [Електронний ресурс] – Режим доступу до ресурсу: https://justjoin.it/blog/все-що-ви-маєте-знати-про-python-які-в-нього-н
32. Java vs Python. Що обрати? [Електронний ресурс] – Режим доступу до ресурсу: https://itvdn.com/ua/blog/article/java-vs-python.
33. Порівнюємо Java та Python або з чого краще почати? [Електронний ресурс] – Режим доступу до ресурсу: https://itproger.com/ua/news/sravnivaem-java-i-python-ili-s-chego-luchshe-nachat.
34. Мова програмування Java: переваги, недоліки та чи варто її вивчати [Електронний ресурс] – Режим доступу до ресурсу: https://discover.in.ua/tips/mova-programuvannja-java-perevagi-nedoliki-ta-chi-varto-yiyi-vivchati.html.
35. А. А. Яровий, О. В. Сілагін, І. В. Богач. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» Вінниця : ВНТУ, 2023. 54 с.

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмний модуль індивідуального планування спортивних тренувань

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0,75 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- ☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Іванчук Я.В.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Власок О.М.
(підпис) (прізвище, ініціали)

ДОДАТОК Б

Інструкція користувача

1) Запускаємо програму "Fitness Planner" подвійним кліком по файлу main.py або через командний рядок командою python main.py в любому інтерпретаторі Python, в нашому випадку це PyCharm з встановленим Python 3.12.

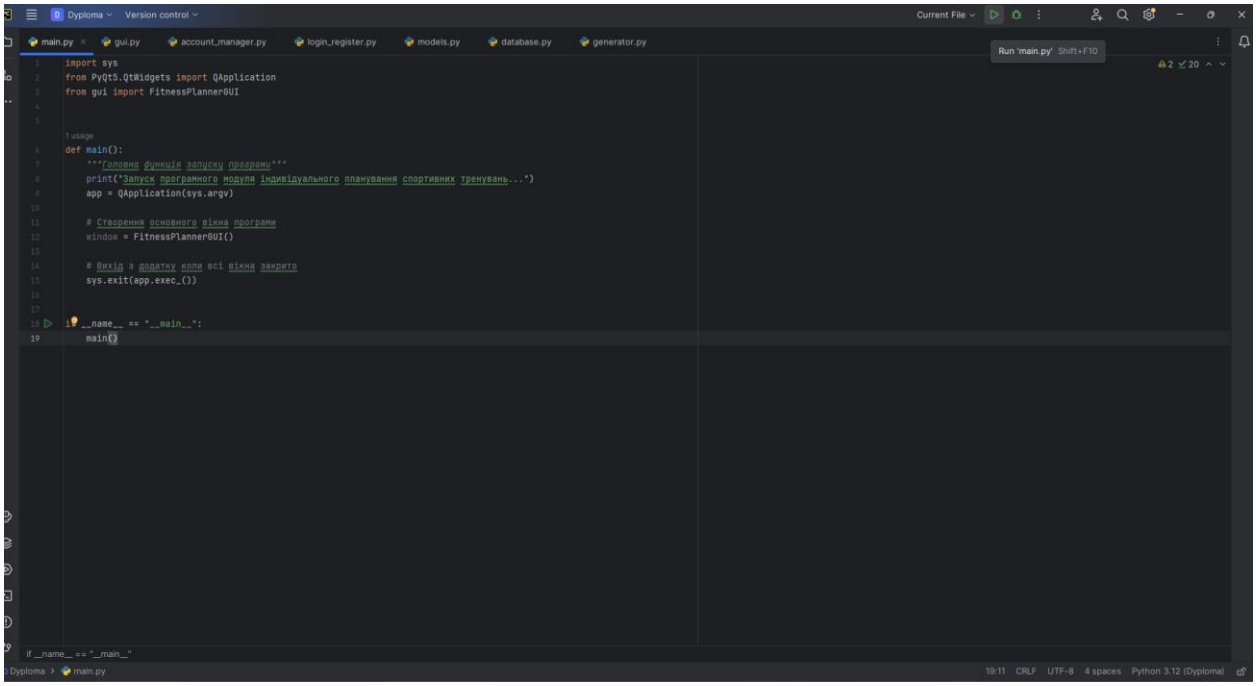

Рисунок Б.1 – Редактор коду PyCharm з встановленим Python 3.12

2) Спочатку користувач має зареєструватися в програмному модулі, ввівши ім’я користувача та пароль. Після успішної реєстрації користувач потрапляє на головну сторінку програми, де може створити свій профіль із зазначенням основних параметрів: ім'я, вік, стать, вага, зріст, рівень підготовки, цілі тренувань та доступне обладнання. Якщо в користувача уже існує аккаунт, тоді достатньо ввійти в систему, ввівши логін та пароль. Зовнішній вигляд вікна реєстації зображений на рисунку Б.2.

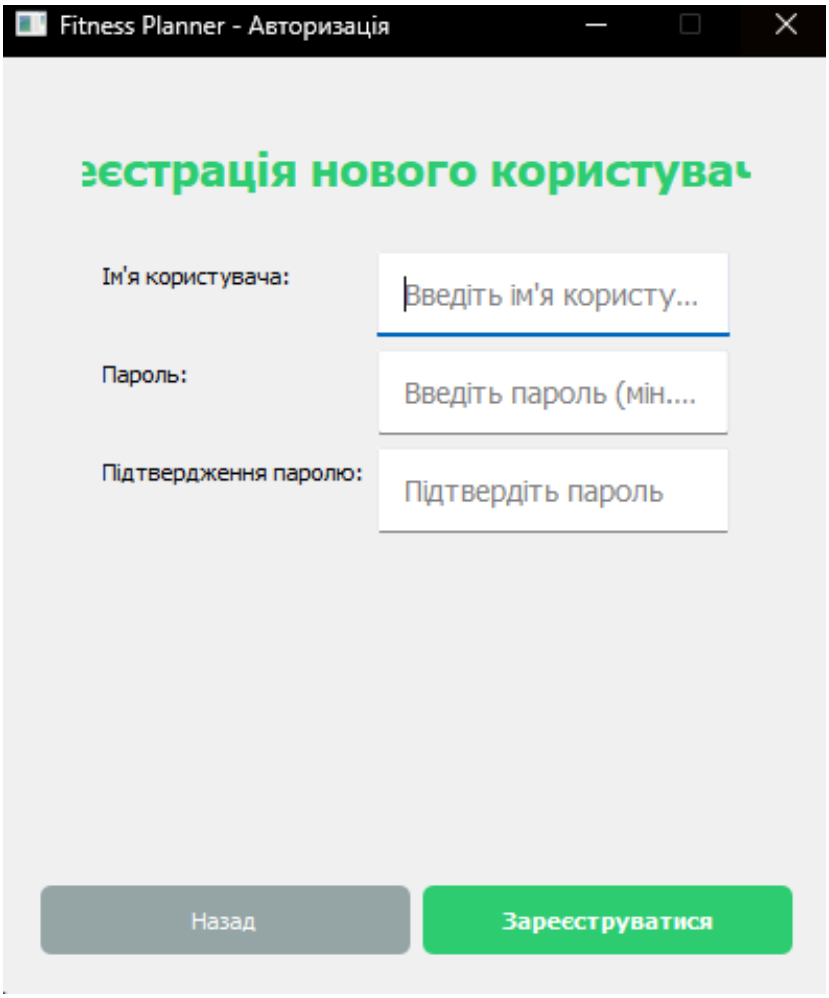

Рисунок Б.2 – Зовнісшій вигляд вікна реєстрації

3) Після створення профілю, користувач зможе побачити такі вкладки, як: «Тренування» – відповідає за відображення згенерованих планів тренувань; «Дієта» – відповідає за рекомендації з харчування та розрахунок калорій; «Прогрес» – відповідає за відстеження виконаних тренувань та статистика; «Вправи» – відповідає за базу даних вправ та можливість додавання нових(рис Б.3).

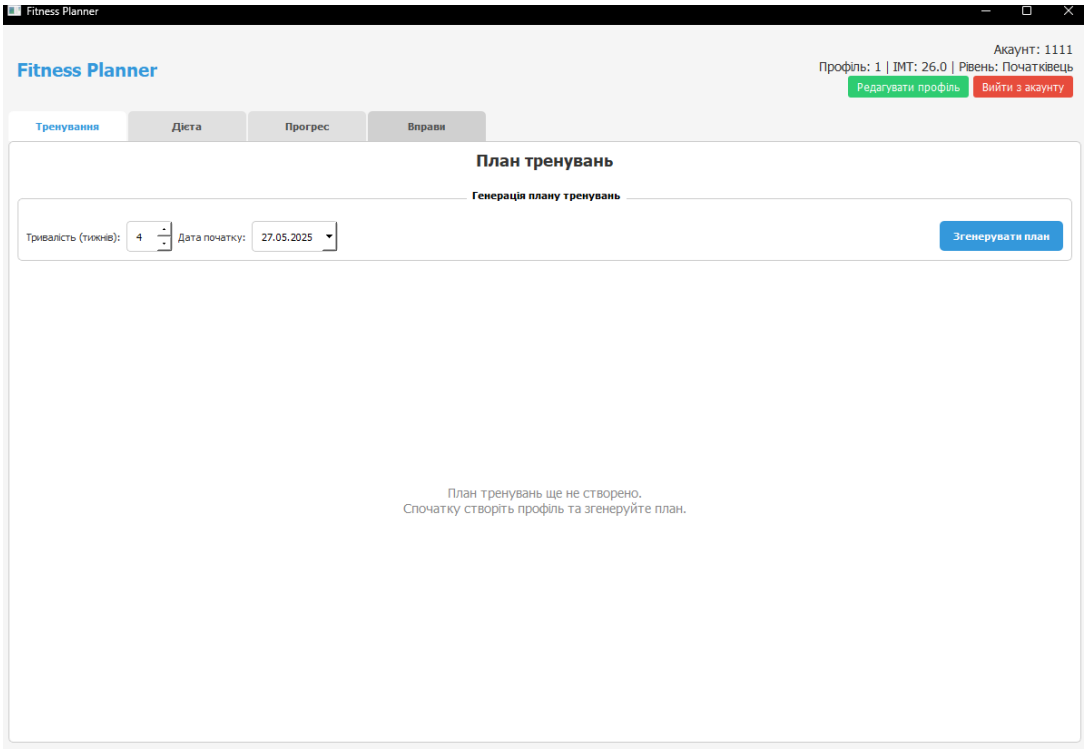

Рисунок Б.3 – Зовнішній вигляд інтерфейсу програмного модуля

4) У вкладці «Тренування» можна створити індивідуальний план тренувань, вказавши тривалість плану в тижнях та дату початку. Програма автоматично генерує персоналізований план з урахуванням усіх параметрів користувача. На екрані буде відображена детальна інформація про кожне тренування: назва, інтенсивність, тривалість, список вправ з кількістю підходів та повторень, як це зображено на рисунку Б.4.
5) Відстеження прогресу тренувань можна подивитися у вкладці «Прогрес». При виконанні тренувань у вкладці «Тренування» їх можна відмічати як виконані, після чого система автоматично буде оновлювати статистику. В статистиці буде відображатися загальна кількість тренувань, виконані тренування, відсоток виконання, витрати калорій та загальна оцінка прогресу, як це зображено на рисунку Б.5.

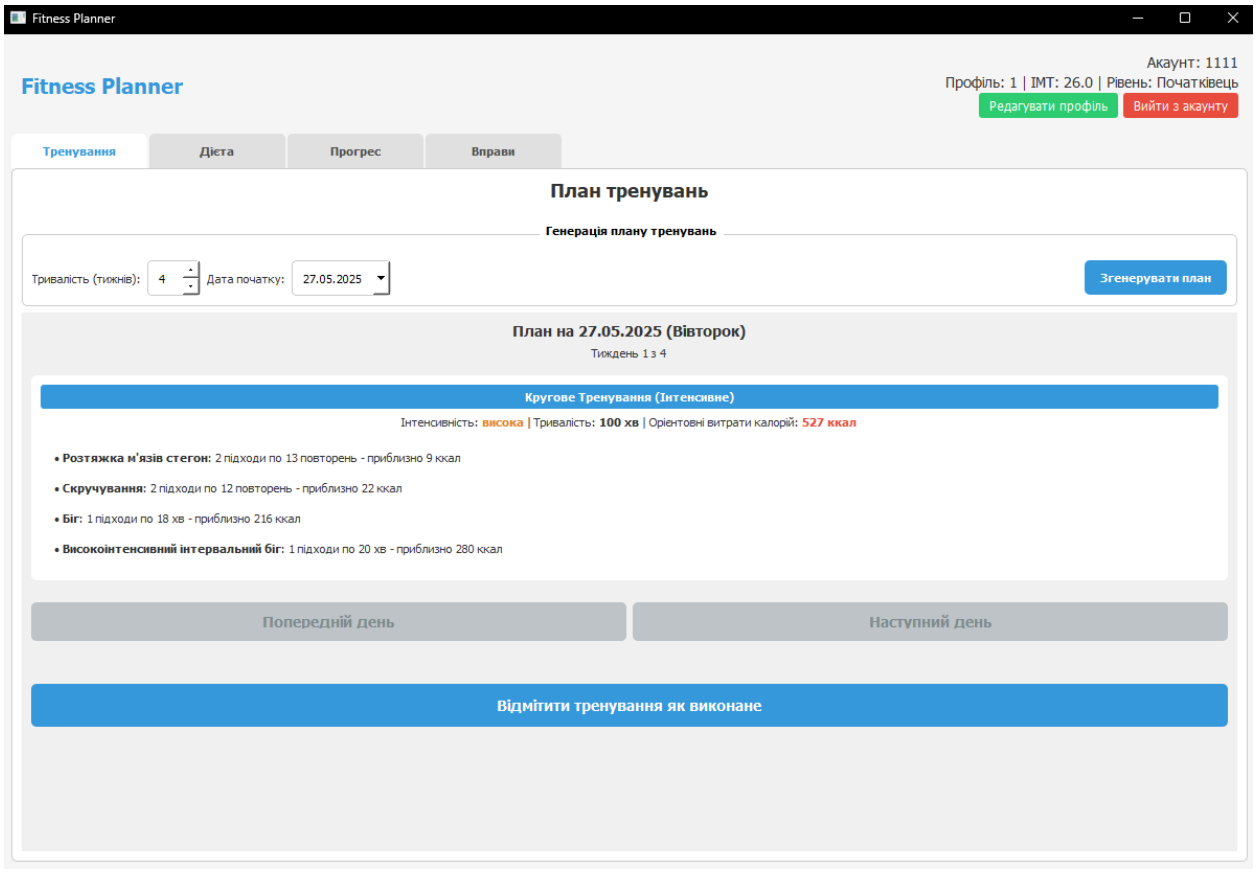

Рисунок Б.4 – Зовнішній вигляд вкладки «Тренування»

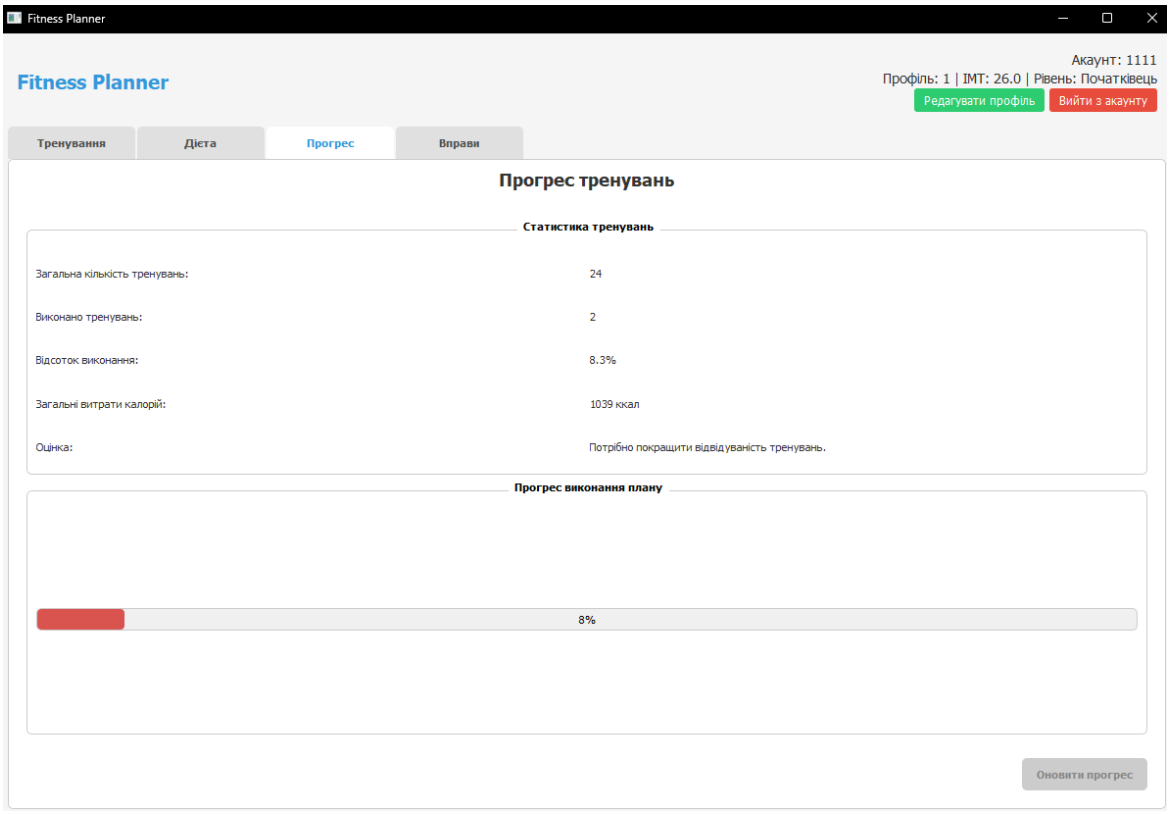

Рисунок Б.5 – Зовнішній вигляд вкладки «Прогрес»

6) Для того, щоб отримати рекомендації з харчування потрібно перейти в вкладку «Дієта», яка зображена на рисунку Б.6, де автоматично розраховуються індивідуальні потреби в калоріях та нутрієнтах(білки, жири, вуглеводи) на основі параметрів користувача, його цілей та плану тренувань, що він вказав під час створення профілю. Система також враховує бажану вагу для розрахунку оптимального дефіциту або профіциту калорій.

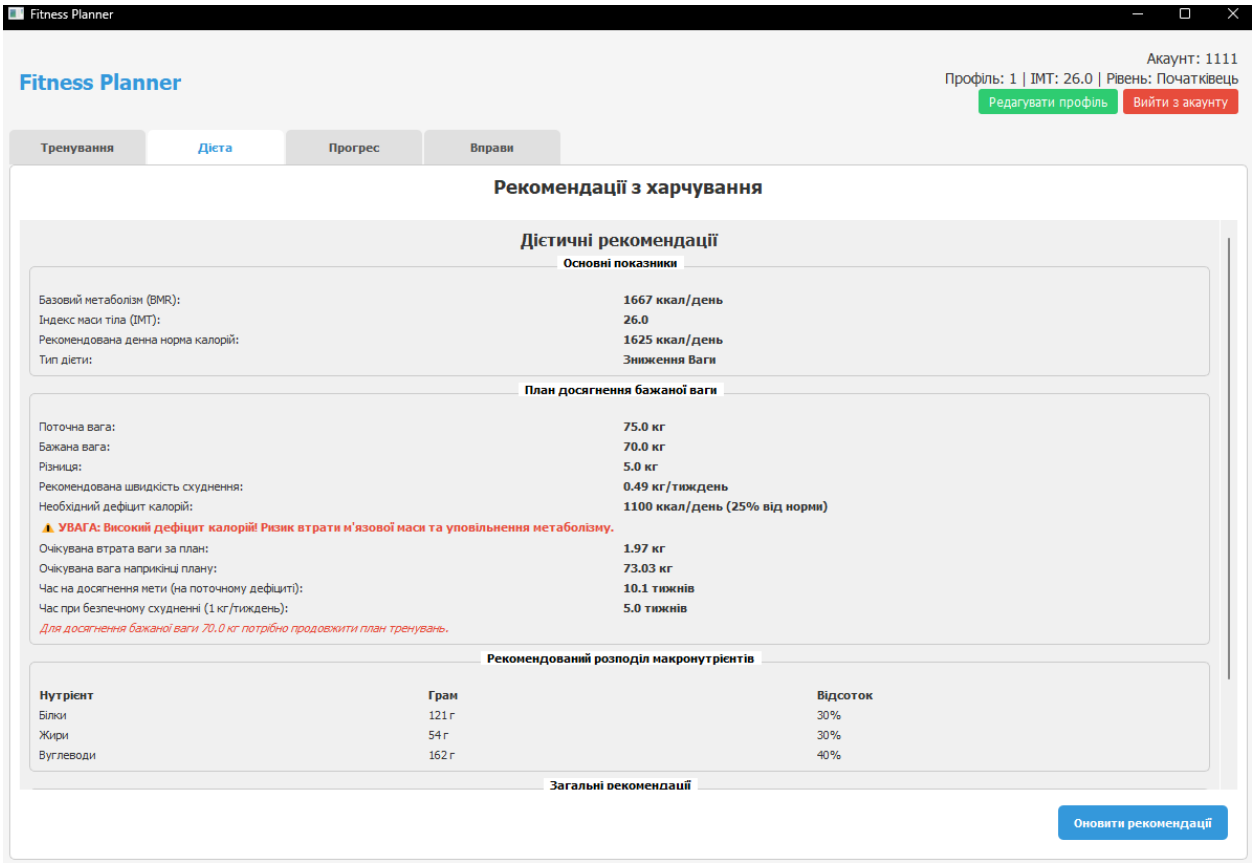

Рисунок Б.6 – Зовнішній вигляд вкладки «Дієта»

7) У вкладці «Вправи» користувач може переглядати базу даних вправ, фільтрувати їх за типом, групою м'язів та складністю, а також додавати власні вправи до бази даних. Кожна вправа містить детальний опис, рівень складності, групи м'язів та необхідне обладнання, як це зображено на рисунку Б.7.

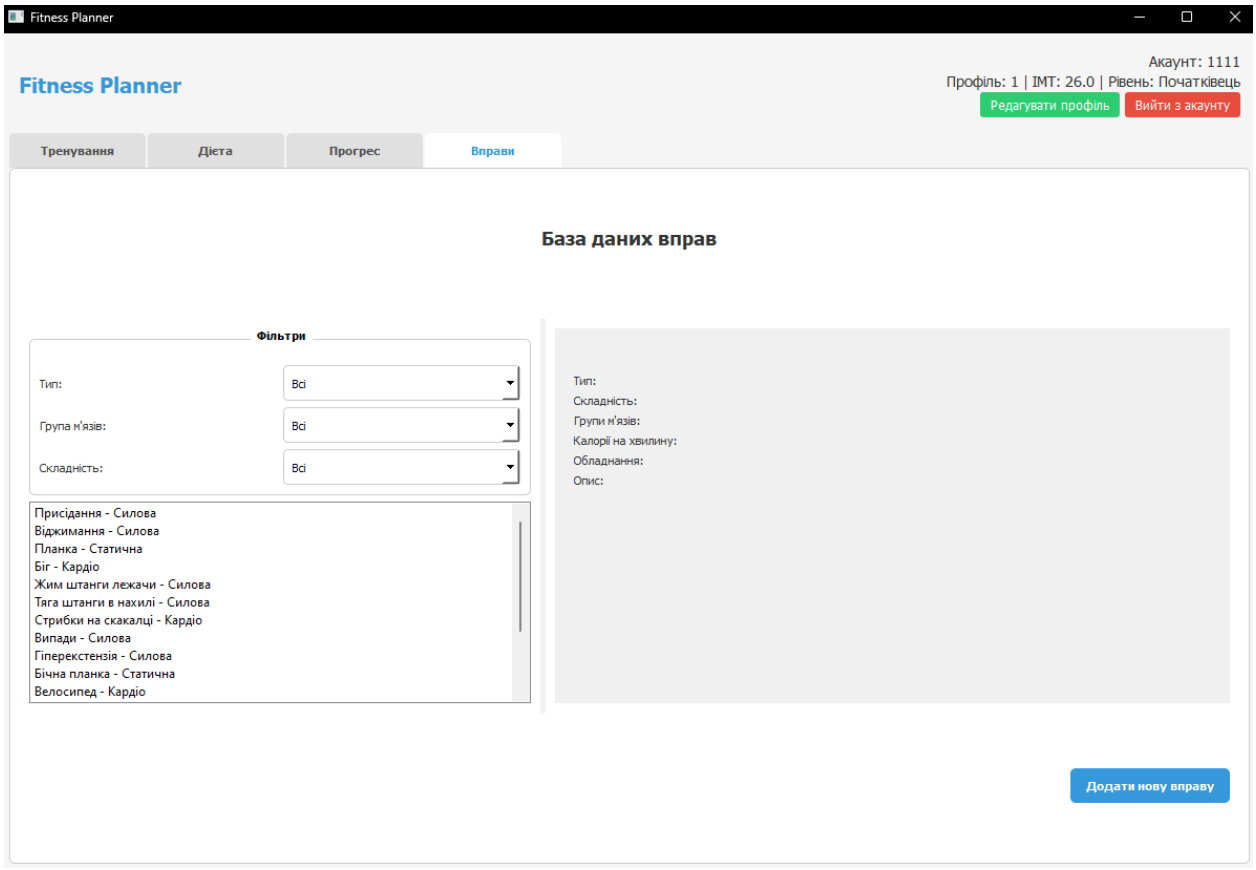

Рисунок Б.7 – Зовнішній вигляд вкладки «Вправи»

8) При необхідності Користувач може редагувати свій профіль у будь-який момент, натиснувши зелену кнопку «Редагувати профіль» справа зверху програмного модулю. Вдкриється нове вікно з усіма параметрами профілю користувача, які можна змінити, як це зображено на рисунку Б.8. Також є можливість повністю видалити профіль користувача.

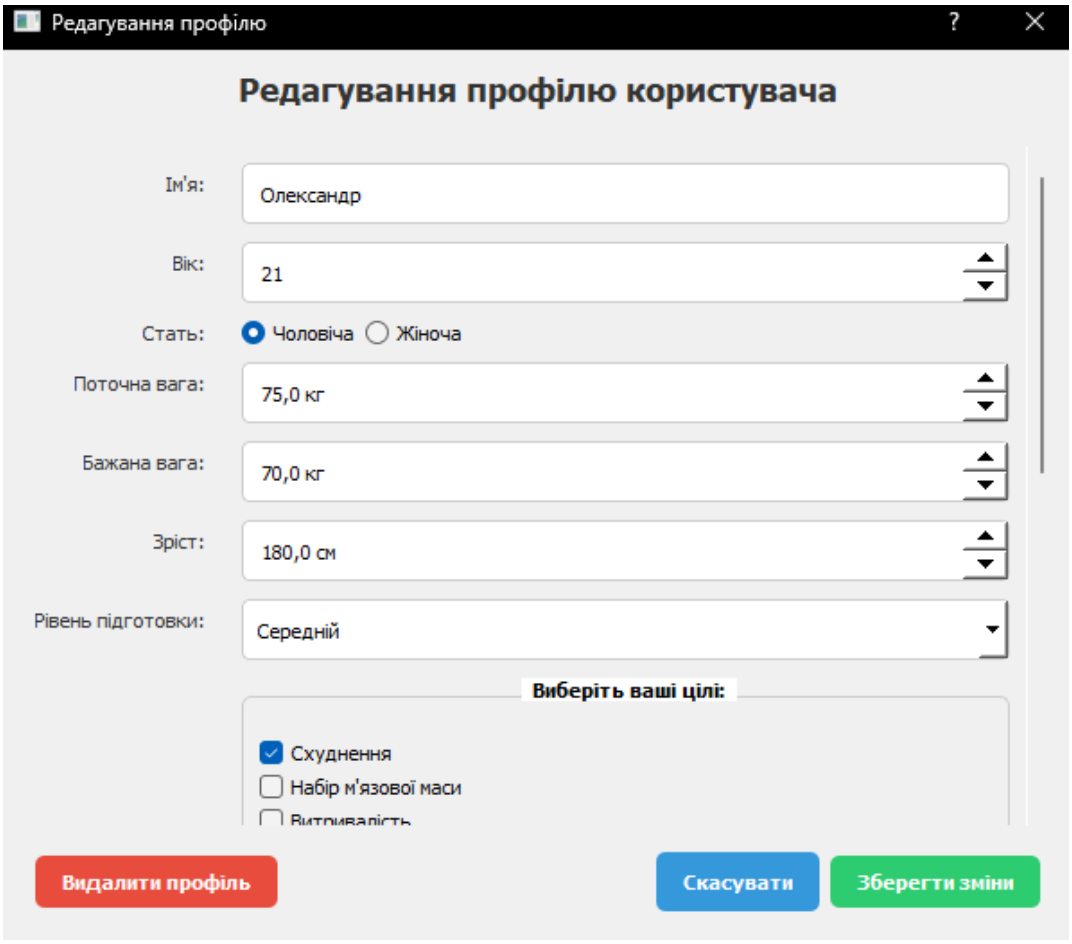

Рисунок Б.8 – Зовнішній вигляд вікна редагування профілю користувача

9) Для виходу з аккаунту користувач натискає кнопку "Вийти з акаунту" у верхньому правому куті програми. Всі дані автоматично зберігаються, після чого програма повертається до вікна авторизації, де можна увійти під іншим акаунтом або завершити роботу.

10) Для завершення роботи з програмним модулем достатньо вийти з нього та вимкнути програму. Всі дані користувача автоматично зберігаються локально у форматі JSON.

ДОДАТОК В

Фрагмент лістингу програмного коду

```
class WorkoutGenerator:
    """Клас для генерації планів тренувань та аналізу прогресу"""

    def __init__(self, database: ExerciseDatabase):
        """
        Ініціалізація генератора тренувань

        Args:
            database: База даних вправ
        """
        self.database = database

    def _select_difficulty_level(self, fitness_level: str) -> int:
        """
        Визначення максимальної складності вправ залежно від рівня підготовки

        Args:
            fitness_level: Рівень підготовки користувача (початківець, середній, досвідчений)

        Returns:
            Максимальний рівень складності вправ
        """
        correspondence = {
            "початківець": 2,
            "середній": 4,
            "досвідчений": 5
        }
        return correspondence.get(fitness_level.lower(), 3)

    def _workouts_per_week(self, fitness_level: str, goals: List[str]) -> int:
        """
        Визначення рекомендованої кількості тренувань на тиждень

        Args:
            fitness_level: Рівень підготовки користувача
            goals: Список цілей користувача

        Returns:
            Рекомендована кількість тренувань на тиждень
        """
        base_count = {
            "початківець": 3,
            "середній": 4,
            "досвідчений": 5
        }.get(fitness_level.lower(), 3)

        # Модифікація залежно від цілей
        if "схуднення" in [goal.lower() for goal in goals]:
            base_count += 1
        if "витривалість" in [goal.lower() for goal in goals]:
            base_count += 1

        return min(base_count, 6)  # Максимум 6 тренувань на тиждень

    def _create_workout(self,
                        day_of_week: int,
```

```
            user: User,
            max_difficulty: int,
            focus: str,
            intensity_factor: float = 1.0,
            week_number: int = 1,
            total_weeks: int = 4,
            target_calories: float = None,
            plan_intensity_level: str = "середня") -> Workout:
"""
Створення одного тренування з урахуванням обмежень і обладнання

Args:
    day_of_week: День тижня (0-6)
    user: Об'єкт користувача
    max_difficulty: Максимальна складність вправ
    focus: Тип тренування (верхня частина тіла, нижня частина тіла, і т.д.)
    intensity_factor: Коефіцієнт інтенсивності тренування (0.7-1.3)
    week_number: Номер поточного тижня (для прогресії)
    total_weeks: Загальна кількість тижнів плану
    target_calories: Цільові витрати калорій для цього тренування

Returns:
    Об'єкт тренування
"""
# Визначаємо базові змінні
is_long_plan = total_weeks > 8
is_short_plan = total_weeks <= 4

# Для коротких планів збільшуємо базову інтенсивність
if is_short_plan:
    intensity_factor = max(intensity_factor, 1.1)  # Підвищуємо мінімальну інтенсивність

# Визначення інтенсивності тренування залежно від факторів
if focus == "відпочинок":
    intensity = "низька"
else:
    # Корекція базової інтенсивності на основі загального рівня плану
    if plan_intensity_level.lower() == "висока":
        base_intensity_factor = intensity_factor * 1.1  # Підвищуємо на 10% для високого рівня
    elif plan_intensity_level.lower() == "низька":
        base_intensity_factor = intensity_factor * 0.9  # Знижуємо на 10% для низького рівня
    else:
        base_intensity_factor = intensity_factor  # Не змінюємо для середнього рівня

    # Базове значення інтенсивності з урахуванням факторів
    if base_intensity_factor >= 1.4:
        intensity = "дуже висока"
    elif base_intensity_factor >= 1.2:
        intensity = "висока"
    elif base_intensity_factor >= 1.0:
        intensity = "середня"
    elif base_intensity_factor >= 0.8:
        intensity = "помірна"
    else:
        intensity = "низька"

    # Додаткова корекція на основі конкретних цілей користувача
    if any(goal.lower() == "схуднення" for goal in user.goals) and user.target_weight is not None:
        weight_difference = user.weight - user.target_weight
        if weight_difference > 0:  # Мета - схуднути
            # Чим більша різниця у вазі і коротший термін, тим вища інтенсивність
            urgency_factor = weight_difference / max(1, total_weeks)
```

```
            if urgency_factor > 2.0:
                # Якщо треба схуднути більше ніж на 2 кг на тиждень - екстремальна інтенсивність
                intensity = "дуже висока"  # Примусово встановлюємо дуже високу інтенсивність
            elif urgency_factor > 1.0:
                # Якщо треба схуднути більше ніж на 1 кг на тиждень - висока інтенсивність
                intensity = "висока"  # Примусово встановлюємо високу інтенсивність
            elif urgency_factor > 0.5:
                # Якщо треба схуднути на 0.5-1 кг на тиждень - помірно підвищуємо
                if intensity not in ["висока", "дуже висока"]:  # Не понижуємо, якщо вже висока
                    intensity = "середня"

exercises_for_workout = []
include_cardio = "витривалість" in [goal.lower() for goal in user.goals] or "схуднення" in [goal.lower() for
                                                                                            goal in user.goals]
base_duration_factor = max(0.6, min(1.5, intensity_factor))

if "схуднення" in [goal.lower() for goal in user.goals] and is_short_plan:
    base_duration_factor = max(base_duration_factor, 1.2)  # Збільшуємо мінімальну тривалість

# --- ВИБІР ВПРАВ ЗАЛЕЖНО ВІД ТИПУ ТРЕНУВАННЯ ---
if focus == "верхня частина тіла":
    # Визначаємо групи м'язів для різних тижнів
    if week_number % 3 == 1:  # Тиждень 1, 4, 7...
        muscle_groups = ["груди", "спина", "плечі", "біцепс"]
    elif week_number % 3 == 2:  # Тиждень 2, 5, 8...
        muscle_groups = ["плечі", "трицепс", "біцепс", "груди"]
    else:  # Тиждень 3, 6, 9...
        muscle_groups = ["спина", "трицепс", "плечі", "груди"]

    # Додаємо силові вправи для верхньої частини тіла
    for group in muscle_groups:
        # Змінюємо максимальну складність вправ залежно від тижня
        if is_long_plan:
            week_max_difficulty = max_difficulty - 2 + min(2, (week_number - 1) // 2)
        elif is_short_plan:
            week_max_difficulty = max_difficulty - 1 + min(1, week_number - 1)
        else:
            week_max_difficulty = max_difficulty

        available_exercises = self.database.find_exercises(
            type="силова",
            muscle_group=group,
            max_difficulty=week_max_difficulty
        )

        # Фільтруємо вправи з урахуванням обмежень і обладнання
        available_exercises = self._filter_exercises_by_limitations_equipment(user, available_exercises)

        if available_exercises:
            # Для різноманітності використовуємо різні сіди для однакових груп м'язів у різні тижні
            random.seed(week_number * 1000 + day_of_week * 100 + hash(group) % 100)
            exercise = random.choice(available_exercises)

            # Коригування кількості підходів і повторень залежно від тижня
            if is_long_plan:
                base_sets = 2 + min(2, (week_number - 1) // 2)  # 2-4 підходи
            elif is_short_plan:
                base_sets = 3 + min(1, week_number - 1)  # 3-4 підходи
            else:
                base_sets = 2 + min(2, week_number - 1)  # 2-4 підходи

            sets = max(2, min(4, base_sets))
```

```
            # Кількість повторень залежить від мети
            if "схуднення" in [goal.lower() for goal in user.goals]:
                if is_short_plan:
                    min_reps = 12 + min(6, (week_number - 1) * 2)  # 12-18
                    max_reps = 15 + min(5, (week_number - 1) * 2)  # 15-20
                else:
                    min_reps = 12 + min(6, (week_number - 1) // 2)  # 12-18 поступово
                    max_reps = 15 + min(5, (week_number - 1) // 2)  # 15-20 поступово

                reps = random.randint(min_reps, max_reps)
            elif "набір м'язової маси" in [goal.lower() for goal in user.goals]:
                if is_short_plan:
                    min_reps = max(6, 10 - min(4, (week_number - 1)))
                    max_reps = max(8, 12 - min(4, (week_number - 1)))
                else:
                    min_reps = max(6, 10 - min(4, (week_number - 1) // 2))
                    max_reps = max(8, 12 - min(4, (week_number - 1) // 2))

                reps = random.randint(min_reps, max_reps)
            else:
                min_reps = 8 + min(4, (week_number - 1) // 2)
                max_reps = 12 + min(3, (week_number - 1) // 2)
                reps = random.randint(min_reps, max_reps)

            exercises_for_workout.append((exercise, sets, reps))

    elif focus == "нижня частина тіла":
        # Визначаємо групи м'язів для різних тижнів
        if week_number % 3 == 1:  # Тиждень 1, 4, 7...
            muscle_groups = ["сідниці", "ноги", "ноги", "сідниці"]
        elif week_number % 3 == 2:  # Тиждень 2, 5, 8...
            muscle_groups = ["ноги", "сідниці", "ноги", "сідниці"]
        else:  # Тиждень 3, 6, 9...
            muscle_groups = ["сідниці", "ноги", "сідниці", "ноги"]

        # Додаємо вправи для різних груп м'язів
        for group in muscle_groups:
            if is_long_plan:
                week_max_difficulty = max_difficulty - 2 + min(2, (week_number - 1) // 2)
            elif is_short_plan:
                week_max_difficulty = max_difficulty - 1 + min(1, week_number - 1)
            else:
                week_max_difficulty = max_difficulty

            available_exercises = self.database.find_exercises(
                type="силова",
                muscle_group=group,
                max_difficulty=week_max_difficulty
            )

            # Фільтруємо вправи з урахуванням обмежень і обладнання
            available_exercises = self._filter_exercises_by_limitations_equipment(user, available_exercises)

            if available_exercises:
                random.seed(week_number * 1000 + day_of_week * 100 + hash(group) % 100)
                exercise = random.choice(available_exercises)

                if is_long_plan:
                    base_sets = 2 + min(2, (week_number - 1) // 2)
                elif is_short_plan:
                    base_sets = 3 + min(1, week_number - 1)
```

```
                else:
                    base_sets = 2 + min(2, week_number - 1)

                sets = max(2, min(4, base_sets))

                if "схуднення" in [goal.lower() for goal in user.goals]:
                    if is_short_plan:
                        min_reps = 12 + min(8, (week_number - 1) * 2)
                        max_reps = 15 + min(10, (week_number - 1) * 2)
                    else:
                        min_reps = 12 + min(8, (week_number - 1) // 2)
                        max_reps = 15 + min(10, (week_number - 1) // 2)

                    reps = random.randint(min_reps, max_reps)
                elif "набір м'язової маси" in [goal.lower() for goal in user.goals]:
                    if is_short_plan:
                        min_reps = max(6, 10 - min(4, (week_number - 1)))
                        max_reps = max(8, 12 - min(4, (week_number - 1)))
                    else:
                        min_reps = max(6, 10 - min(4, (week_number - 1) // 2))
                        max_reps = max(8, 12 - min(4, (week_number - 1) // 2))

                    reps = random.randint(min_reps, max_reps)
                else:
                    min_reps = 8 + min(4, (week_number - 1) // 2)
                    max_reps = 12 + min(3, (week_number - 1) // 2)
                    reps = random.randint(min_reps, max_reps)

                exercises_for_workout.append((exercise, sets, reps))

        # Продовжимо метод у другій частині
        return self._create_workout_part2(
            user=user,
            focus=focus,
            max_difficulty=max_difficulty,
            intensity_factor=intensity_factor,
            week_number=week_number,
            total_weeks=total_weeks,
            target_calories=target_calories,
            is_long_plan=is_long_plan,
            is_short_plan=is_short_plan,
            base_duration_factor=base_duration_factor,
            include_cardio=include_cardio,
            day_of_week=day_of_week,
            exercises_for_workout=exercises_for_workout,
            intensity=intensity,
            plan_intensity_level=plan_intensity_level
        )


    def _create_workout_part2(self,
                              user: User,
                              focus: str,
                              max_difficulty: int,
                              intensity_factor: float,
                              week_number: int,
                              total_weeks: int,
                              target_calories: float,
                              is_long_plan: bool,
                              is_short_plan: bool,
                              base_duration_factor: float,
                              include_cardio: bool,
```

```
                day_of_week: int,
                exercises_for_workout: list,
                intensity: str,
                plan_intensity_level: str = "середня") -> Workout:

total_exercise_duration = 0
"""
Друга частина методу _create_workout
"""
if focus == "все тіло":
    # Визначаємо групи м'язів з різним порядком для різних тижнів
    if week_number % 3 == 1:
        muscle_groups = ["груди", "спина", "ноги", "плечі", "прес"]
    elif week_number % 3 == 2:
        muscle_groups = ["ноги", "груди", "спина", "прес", "плечі"]
    else:
        muscle_groups = ["спина", "ноги", "груди", "плечі", "прес"]

    # Додаємо вправи для різних груп м'язів
    for group in muscle_groups:
        if is_long_plan:
            week_max_difficulty = max_difficulty - 2 + min(2, (week_number - 1) // 2)
        elif is_short_plan:
            week_max_difficulty = max_difficulty - 1 + min(1, week_number - 1)
        else:
            week_max_difficulty = max_difficulty

        available_exercises = self.database.find_exercises(
            muscle_group=group,
            max_difficulty=week_max_difficulty
        )

        # Фільтруємо вправи з урахуванням обмежень і обладнання
        available_exercises = self._filter_exercises_by_limitations_equipment(user, available_exercises)

        if available_exercises:
            random.seed(week_number * 1000 + day_of_week * 100 + hash(group) % 100)
            exercise = random.choice(available_exercises)

            if is_long_plan:
                base_sets = 2 + min(2, (week_number - 1) // 2)
            elif is_short_plan:
                base_sets = 2 + min(1, week_number - 1)
            else:
                base_sets = 2 + min(2, week_number - 1)

            sets = max(2, min(3, base_sets))  # Для всього тіла - менше підходів

            if "схуднення" in [goal.lower() for goal in user.goals]:
                if is_short_plan:
                    min_reps = 12 + min(6, (week_number - 1) * 2)
                    max_reps = 15 + min(5, (week_number - 1) * 2)
                else:
                    min_reps = 12 + min(6, (week_number - 1) // 2)
                    max_reps = 15 + min(5, (week_number - 1) // 2)

                reps = random.randint(min_reps, max_reps)
            else:
                min_reps = 10 + min(4, (week_number - 1) // 2)
                max_reps = 12 + min(3, (week_number - 1) // 2)
                reps = random.randint(min_reps, max_reps)
```

```
            exercises_for_workout.append((exercise, sets, reps))

    elif focus == "кардіо":
        # Визначаємо різні види кардіо для різних тижнів
        if week_number % 3 == 1:  # Тиждень 1, 4, 7...
            cardio_types = ["біг", "стрибки", "велосипед"]
        elif week_number % 3 == 2:  # Тиждень 2, 5, 8...
            cardio_types = ["велосипед", "стрибки на скакалці", "біг"]
        else:  # Тиждень 3, 6, 9...
            cardio_types = ["інтервальне кардіо", "біг", "велосипед"]

        # Додаємо кардіо вправи
        cardio_exercises = self.database.find_exercises(
            type="кардіо",
            max_difficulty=max_difficulty
        )

        # ВИПРАВЛЕННЯ: Фільтруємо вправи з урахуванням обмежень і обладнання
        cardio_exercises = self._filter_exercises_by_limitations_equipment(user, cardio_exercises)

        # Якщо після фільтрації немає кардіо вправ, використовуємо будь-які без обладнання
        if not cardio_exercises:
            cardio_exercises = [e for e in self.database.find_exercises(type="кардіо") if not e.equipment]

        # Переконуємося, що маємо хоча б одну вправу
        if not cardio_exercises:
            # Створюємо базову вправу - біг
            for exercise in self.database.exercises:
                if exercise.name.lower() in ["біг", "ходьба", "стрибки"]:
                    cardio_exercises = [exercise]
                    break

        # Вибираємо унікальні вправи
        if cardio_exercises:
            # Визначаємо кількість кардіо вправ залежно від тижня і інтенсивності
            num_exercises = min(len(cardio_exercises), 2 + (week_number - 1) // 2)
            num_exercises = min(num_exercises, 4)  # Не більше 4 різних кардіо вправ

            # Додаємо різноманітні кардіо вправи
            random.seed(week_number * 1000 + day_of_week * 100)
            selected_exercises = random.sample(cardio_exercises, min(num_exercises, len(cardio_exercises)))

            for exercise in selected_exercises:
                # Прогресія інтенсивності через тижні
                if is_long_plan:
                    # Для довгих планів - поступова прогресія
                    base_sets = 1 + min(1, (week_number - 1) // 4)  # 1-2 підходи
                    base_duration = 10 + min(15, (week_number - 1) * 2)  # 10-25 хвилин
                elif is_short_plan:
                    # Для коротких планів - швидша прогресія
                    base_sets = 1
                    base_duration = 15 + min(10, (week_number - 1) * 5)  # 15-25 хвилин
                else:
                    # Для середніх планів
                    base_sets = 1
                    base_duration = 10 + min(15, (week_number - 1) * 3)  # 10-25 хвилин

                sets = max(1, min(2, base_sets))

                # Для схуднення - довші кардіо сесії
                if "схуднення" in [goal.lower() for goal in user.goals]:
                    duration = max(15, min(30, int(base_duration * base_duration_factor)))
```

```
            else:
                duration = max(10, min(25, int(base_duration * base_duration_factor)))

            exercises_for_workout.append((exercise, sets, duration))

    elif focus == "відновлення":
        # Додаємо вправи на розтяжку та відновлення
        recovery_exercises = self.database.find_exercises(
            type="гнучкість",
            max_difficulty=max_difficulty
        ) + self.database.find_exercises(
            type="статична",
            max_difficulty=max_difficulty
        )

        # Фільтруємо вправи з урахуванням обмежень і обладнання
        recovery_exercises = self._filter_exercises_by_limitations_equipment(user, recovery_exercises)

        # Якщо немає вправ після фільтрації, використовуємо базові вправи на розтяжку
        if not recovery_exercises:
            recovery_exercises = [e for e in self.database.find_exercises(type="гнучкість") if not e.equipment]

        if recovery_exercises:
            # Вибираємо кілька вправ на розтяжку
            num_exercises = min(len(recovery_exercises), 3)
            random.seed(week_number * 1000 + day_of_week * 100)
            selected_exercises = random.sample(recovery_exercises, num_exercises)

            for exercise in selected_exercises:
                sets = 1
                if exercise.type.lower() == "статична":
                    # Для статичних вправ використовуємо хвилини
                    duration = 1  # 1 хвилина утримання
                else:
                    # Для інших вправ
                    duration = 3  # 3 хвилини на вправу
                exercises_for_workout.append((exercise, sets, duration))

    # Додаємо кардіо наприкінці силового тренування для схуднення, якщо потрібно
    if include_cardio and focus not in ["кардіо", "відновлення"] and intensity_factor >= 0.9:
        cardio_exercises = self.database.find_exercises(
            type="кардіо",
            max_difficulty=max_difficulty
        )

        # ВИПРАВЛЕННЯ: Фільтруємо вправи з урахуванням обмежень і обладнання
        cardio_exercises = self._filter_exercises_by_limitations_equipment(user, cardio_exercises)

        # Якщо після фільтрації немає кардіо вправ, використовуємо будь-які без обладнання
        if not cardio_exercises:
            cardio_exercises = [e for e in self.database.find_exercises(type="кардіо") if not e.equipment]

        if cardio_exercises:
            random.seed(week_number * 1000 + day_of_week * 100 + 42)  # Інший сід для різноманітності
            exercise = random.choice(cardio_exercises)
            sets = 1

            # Тривалість кардіо залежить від інтенсивності та тижня
            if is_long_plan:
                base_duration = 5 + min(15, (week_number - 1) // 2)  # 5-20 хвилин поступово
            elif is_short_plan:
                base_duration = 10 + min(10, (week_number - 1) * 3)  # 10-20 хвилин швидко
```

```
            else:
                base_duration = 10 + min(10, week_number - 1)  # 10-20 хвилин

            # Для схуднення коригуємо тривалість залежно від інтенсивності
            if "схуднення" in [goal.lower() for goal in user.goals]:
                duration = max(10, min(20, int(base_duration * base_duration_factor * 1.2)))
            else:
                duration = max(5, min(15, int(base_duration * base_duration_factor)))

            exercises_for_workout.append((exercise, sets, duration))

    # Переконуємося, що маємо хоча б одну вправу в тренуванні
    # Переконуємося, що маємо хоча б одну вправу в тренуванні
    if not exercises_for_workout:
        print("УВАГА: Немає вправ для тренування, додаємо базові вправи")

        # Якщо після всіх фільтрацій у нас немає вправ, додаємо базові вправи без обладнання
        basic_exercises = [e for e in self.database.exercises if not e.equipment]

        # Якщо і без обладнання немає вправ - створюємо базові
        if not basic_exercises:
            print("УВАГА: Створюємо базові вправи, бо в базі даних їх немає")
            # Створюємо базову вправу у будь-якому випадку
            from models import Exercise

            if focus == "кардіо":
                # Додаємо базову кардіо вправу
                basic_exercise = Exercise(
                    name="Біг",
                    type="кардіо",
                    difficulty=3,
                    muscle_groups=["ноги", "серцево-судинна система"],
                    calories_per_minute=10.0,
                    description="Базова кардіо вправа",
                    equipment=[]
                )

                exercises_for_workout.append((basic_exercise, 2, 20))  # 2 підходи по 20 хвилин
            else:
                # Додаємо базову силову вправу
                basic_exercise = Exercise(
                    name="Присідання",
                    type="силова",
                    difficulty=2,
                    muscle_groups=["ноги", "сідниці"],
                    calories_per_minute=8.0,
                    description="Базова вправа для нижньої частини тіла",
                    equipment=[]
                )

                exercises_for_workout.append((basic_exercise, 3, 15))  # 3 підходи по 15 повторень

        if basic_exercises:
            exercise = basic_exercises[0]  # Беремо першу доступну базову вправу
            sets = 3
            reps = 15
            exercises_for_workout.append((exercise, sets, reps))
        else:
            # У крайньому випадку, якщо взагалі немає вправ, створюємо заглушку
            dummy_exercise = Exercise(
                name="Базова вправа",
                type="силова" if focus != "кардіо" else "кардіо",
```

```
                difficulty=1,
                muscle_groups=["все тіло"],
                calories_per_minute=5.0,
                description="Базова вправа без обладнання",
                equipment=[]
            )
            exercises_for_workout.append((dummy_exercise, 3, 15))

        # Розрахунок загальної тривалості тренування та калорій
        total_calories = 0
        estimated_duration = 0
        total_exercise_duration = 0  # Ініціалізуємо змінну

        for exercise, sets, reps in exercises_for_workout:
            # Для кардіо reps - це тривалість у хвилинах
            if exercise.type.lower() == "кардіо":
                exercise_duration = sets * reps
                calories = exercise.calories_per_minute * exercise_duration
            else:
                # Для силових вправ рахуємо приблизну тривалість
                exercise_duration = sets * (reps / 10 + 1)  # 10 повторень ≈ 2 хвилини на підхід
                calories = exercise.calories_per_minute * exercise_duration

            estimated_duration += exercise_duration
            total_calories += calories
            total_exercise_duration += exercise_duration  # Додаємо до загальної тривалості

        # Якщо задано цільові калорії, коригуємо тривалість тренування
        if target_calories and total_calories < target_calories:
            # Збільшуємо тривалість для досягнення цільових калорій
            calories_ratio = target_calories / max(1, total_calories)
            estimated_duration = estimated_duration * min(1.5, calories_ratio)  # Не збільшуємо більше ніж в 1.5 рази

            # : Коригування тривалості тренування для коротких планів
        if is_short_plan:
            # Для коротких планів - одразу більша тривалість
            min_duration = 45 + min(15, (week_number - 1) * 5)  # 45-60 хвилин
            max_duration = 60 + min(15, (week_number - 1) * 5)  # 60-75 хвилин
        elif is_long_plan:
            # Для довгих планів - поступове збільшення тривалості
            min_duration = 30 + min(15, (week_number - 1) * 2)  # 30-45 хвилин
            max_duration = 45 + min(15, (week_number - 1) * 2)  # 45-60 хвилин
        else:
            # Для середніх планів
            min_duration = 35 + min(15, (week_number - 1) * 3)  # 35-50 хвилин
            max_duration = 50 + min(15, (week_number - 1) * 3)  # 50-65 хвилин

            # : Забезпечуємо мінімальну тривалість для короткострокових планів схуднення
        if "схуднення" in [goal.lower() for goal in user.goals] and is_short_plan:
            min_duration = max(min_duration, 50)  # Мінімум 50 хвилин

        # Встановлюємо результуючу тривалість тренування
        # Встановлюємо результуючу тривалість тренування
        # Коригуємо підходи та повторення щоб загальна тривалість відповідала розрахованій
        # Для кардіо тренувань перераховуємо тривалість
        if focus == "кардіо" and exercises_for_workout:
            # Перевіряємо співвідношення між розрахованою та бажаною тривалістю
            min_needed_duration = min_duration  # min_duration вже визначено вище

            # Якщо total_exercise_duration значно відрізняється від потрібного
            if total_exercise_duration < min_needed_duration * 0.9 or total_exercise_duration > min_needed_duration *
1.2:
```

```
            ratio = min_needed_duration / max(1, total_exercise_duration)

            # Оновлюємо значення для всіх вправ
            for i in range(len(exercises_for_workout)):
                exercise, sets, reps = exercises_for_workout[i]
                if exercise.type.lower() == "кардіо":
                    # Коригуємо тривалість вправи, зберігаючи розумні значення
                    new_reps = max(5, min(30, int(reps * ratio)))
                    exercises_for_workout[i] = (exercise, sets, new_reps)

            # Перерахунок загальної тривалості
            total_exercise_duration = 0
            for exercise, sets, reps in exercises_for_workout:
                if exercise.type.lower() == "кардіо":
                    total_exercise_duration += sets * reps

        # Встановлюємо результуючу тривалість тренування
        if focus == "кардіо":
            # Для кардіо тренувань тривалість має бути приблизно рівна сумі тривалостей вправ
            duration = max(total_exercise_duration, min_duration)
            duration = min(duration, max_duration)  # Обмежуємо максимальною тривалістю
        else:
            duration = max(min_duration, min(max_duration, int(estimated_duration)))
        # Встановлюємо цільові рівні витрат калорій залежно від інтенсивності
        target_calories_by_intensity = {
            "дуже висока": (600, 800),  # мін-макс для дуже високої інтенсивності
            "висока": (400, 600),  # мін-макс для високої інтенсивності
            "середня": (200, 400),  # мін-макс для середньої інтенсивності
            "помірна": (100, 200),  # мін-макс для помірної інтенсивності
            "низька": (50, 150)  # мін-макс для низької інтенсивності
        }

        # Визначаємо цільовий діапазон калорій для поточної інтенсивності
        min_calories, max_calories = target_calories_by_intensity.get(intensity.lower(), (200, 400))

        # Якщо очікувані витрати калорій не відповідають цільовому діапазону
        if total_calories < min_calories:
            # Збільшуємо тривалість тренування для досягнення мінімального рівня калорій
            calories_ratio = min_calories / max(1, total_calories)
            new_duration = int(
                duration * min(2.0, calories_ratio))  # Дозволяємо збільшення до 2х для високих інтенсивностей

            # Обмежуємо максимальну тривалість
            new_duration = min(new_duration, 120)  # Максимум 2 години тренування

            # Оновлюємо тривалість тренування
            duration = max(duration, new_duration)

            # Перерахунок калорій для точності
            total_calories = min_calories
        elif total_calories > max_calories:
            # Це рідкісний випадок, коли тренування занадто інтенсивне
            # Ми не зменшуємо тривалість, але можемо зменшити кількість вправ в майбутньому
            total_calories = max_calories

        # Додаємо примітку про калорії залежно від інтенсивності
        workout_note = ""
        if intensity.lower() == "дуже висока":
            workout_note = "Інтенсивне тренування для максимального спалювання калорій. Обережно
контролюйте своє самопочуття."
        elif intensity.lower() == "висока":
            workout_note = "Тренування високої інтенсивності для ефективного спалювання калорій."
```

```
elif intensity.lower() == "середня":
    workout_note = "Збалансоване тренування з помірним спалюванням калорій."

# Забезпечення мінімального рівня калорій залежно від інтенсивності тренування
min_calories_by_intensity = {
    "дуже висока": 600,
    "висока": 400,
    "середня": 300,
    "помірна": 200,
    "низька": 100
}

# Якщо очікувані витрати калорій менші за мінімальний рівень для даної інтенсивності
min_required_calories = min_calories_by_intensity.get(intensity, 200)
if total_calories < min_required_calories:
    # Збільшуємо тривалість тренування для досягнення мінімального рівня калорій
    calories_ratio = min_required_calories / max(1, total_calories)
    adjusted_duration = int(duration * min(1.5, calories_ratio))  # Не збільшуємо більше ніж в 1.5 рази

    # Оновлюємо тривалість тренування
    duration = max(duration, adjusted_duration)

    # Перерахунок калорій (для інформаційних цілей)
    total_calories = min_required_calories

workout_names = {
    "верхня частина тіла": [
        "Тренування Верхня Частина Тіла: Груди і Спина",
        "Тренування Верхня Частина Тіла: Плечі і Руки",
        "Тренування Верхня Частина Тіла: Комплексне",
        "Силове Тренування: Верхня Частина",
        "Тренування Рук і Плечей",
        "Тренування Груди та Спина"
    ],
    "нижня частина тіла": [
        "Тренування Нижня Частина Тіла: Ноги",
        "Тренування Нижня Частина Тіла: Сідниці",
        "Тренування Нижня Частина Тіла: Повний комплекс",
        "Силове Тренування: Нижня Частина",
        "Тренування Ніг",
        "Тренування Сідниць і Ніг"
    ],
    "кардіо": [
        "Тренування Кардіо: Витривалість",
        "Тренування Кардіо: Інтервали",
        "Тренування Кардіо: Комплекс",
        "Інтенсивне Кардіо",
        "Інтервальне Кардіо",
        "Кардіо + Метаболічне Тренування"
    ],
    "все тіло": [
        "Тренування Всього Тіла: Комплекс",
        "Тренування Всього Тіла: Базові Вправи",
        "Тренування Всього Тіла: Круговий Формат",
        "Комплексне Тренування",
        "Функціональне Тренування",
        "Кругове Тренування"
    ],
    "відновлення": [
        "Відновлювальне Тренування",
        "Тренування на Гнучкість",
        "Активне Відновлення",
```

```
            "Розтяжка та Мобільність",
            "Відновлювальна Йога",
            "Тренування на Рухливість"
        ]
    }

    # ВИПРАВЛЕННЯ: Більше варіативності у назвах тренувань
    name_idx = (hash(f"{focus}_{week_number}_{day_of_week}") % len(workout_names.get(focus,
["Тренування"])))
    workout_name = workout_names.get(focus, ["Тренування"])[name_idx]

    # ВИПРАВЛЕННЯ: Додаємо позначку інтенсивності для схуднення при коротких планах
    if "схуднення" in [goal.lower() for goal in user.goals] and is_short_plan:
        workout_name = f"{workout_name} (Інтенсивне)"

    # Перевірка для забезпечення правильної інтенсивності
    if intensity.lower() == "дуже висока" and not any(e[0].type.lower() == "кардіо" for e in
exercises_for_workout):
        # Якщо немає кардіо вправ, додаємо інтенсивну кардіо вправу
        cardio_exercises = [e for e in user.database.exercises if e.type.lower() == "кардіо"]
        if cardio_exercises:
            cardio_exercise = random.choice(cardio_exercises)
            exercises_for_workout.append((cardio_exercise, 2, 15))  # 2 підходи по 15 хвилин
            duration += 30  # Додаємо 30 хвилин до тривалості

    # Розрахунок поточних калорій від вправ
    current_calories = 0
    for exercise, sets, reps in exercises_for_workout:
        if exercise.type.lower() == "кардіо":
            exercise_duration = sets * reps
            exercise_calories = exercise.calories_per_minute * exercise_duration
        else:
            # Для силових вправ рахуємо приблизну тривалість
            exercise_duration = sets * (reps / 10 + 1)
            exercise_calories = exercise.calories_per_minute * exercise_duration
        current_calories += exercise_calories

    # Різниця між цільовими і поточними калоріями
    calories_difference = target_calories - current_calories

    # Коригуємо калорійність вправ, щоб досягти цільового значення
    if abs(calories_difference) > 0.1 * target_calories:  # Якщо різниця більше 10% від цільового
        # Коефіцієнт корекції
        adjust_factor = target_calories / current_calories if current_calories > 0 else 1.0

        # Якщо вправ забагато, залишаємо тільки потрібні
        if calories_difference < 0 and len(exercises_for_workout) > 2:
            # Сортуємо вправи за калорійністю
            exercises_calories = []
            for i, (exercise, sets, reps) in enumerate(exercises_for_workout):
                if exercise.type.lower() == "кардіо":
                    exercise_duration = sets * reps
                    exercise_calories = exercise.calories_per_minute * exercise_duration
                else:
                    exercise_duration = sets * (reps / 10 + 1)
                    exercise_calories = exercise.calories_per_minute * exercise_duration
                exercises_calories.append((i, exercise_calories))

            # Сортуємо за калорійністю (від більшої до меншої)
            exercises_calories.sort(key=lambda x: x[1], reverse=True)

            # Обираємо вправи, щоб досягти цільових калорій
```

```
        selected_exercises = []
        current_sum = 0
        for i, calories in exercises_calories:
            if current_sum + calories <= target_calories * 1.1:  # Допускаємо перевищення на 10%
                selected_exercises.append(exercises_for_workout[i])
                current_sum += calories

            if current_sum >= target_calories * 0.9:  # Якщо досягли 90% від цільового - зупиняємось
                break

        # Оновлюємо список вправ, якщо вибрали достатньо
        if len(selected_exercises) >= 2 and current_sum >= target_calories * 0.8:
            exercises_for_workout = selected_exercises

    # Якщо вправ замало, додаємо або коригуємо існуючі
    elif calories_difference > 0:
        # Якщо є кардіо вправи, збільшуємо їх тривалість
        cardio_exercises = [(i, ex) for i, (ex, sets, reps) in enumerate(exercises_for_workout)
                            if ex.type.lower() == "кардіо"]

        if cardio_exercises:
            # Збільшуємо тривалість кардіо вправ
            for i, exercise in cardio_exercises:
                exercise, sets, reps = exercises_for_workout[i]
                # Збільшуємо тривалість на 30% або до потрібного значення
                additional_time = min(int(reps * 0.3),
                                      int(calories_difference / (exercise.calories_per_minute * sets)))
                new_reps = reps + additional_time
                exercises_for_workout[i] = (exercise, sets, new_reps)
        else:
            # Якщо немає кардіо, додаємо нову кардіо вправу
            from models import Exercise
            cardio = Exercise(
                name="Додаткове кардіо",
                type="кардіо",
                difficulty=3,
                muscle_groups=["серцево-судинна система"],
                calories_per_minute=10.0,
                description="Додаткове кардіо для досягнення цільових калорій",
                equipment=[]
            )

            # Розраховуємо потрібну тривалість
            needed_duration = int(calories_difference / cardio.calories_per_minute)
            if needed_duration >= 5:  # Якщо потрібно хоча б 5 хвилин
                exercises_for_workout.append((cardio, 1, needed_duration))
# Визначаємо цільові калорії для різних рівнів інтенсивності
target_calories_by_intensity = {
    "дуже висока": random.randint(650, 800),
    "висока": random.randint(450, 600),
    "середня": random.randint(350, 450),
    "помірна": random.randint(250, 350),
    "низька": random.randint(150, 250)
}

# Визначаємо цільові калорії для поточної інтенсивності
target_calories = target_calories_by_intensity.get(intensity.lower(), 350)

# Розрахунок поточних калорій від вправ
current_calories = 0
for exercise, sets, reps in exercises_for_workout:
    if exercise.type.lower() == "кардіо":
```

```
            exercise_duration = sets * reps
            exercise_calories = exercise.calories_per_minute * exercise_duration
        else:
            exercise_duration = sets * (reps / 10 + 1)
            exercise_calories = exercise.calories_per_minute * exercise_duration
        current_calories += exercise_calories

    # Різниця між цільовими і поточними калоріями
    calories_difference = target_calories - current_calories

    # Якщо калорій не вистачає - додаємо кардіо
    if calories_difference > 0.2 * target_calories:
        # Додаємо нову кардіо вправу
        from models import Exercise

        # Створюємо кардіо вправу відповідно до інтенсивності
        # Створюємо кардіо вправу відповідно до інтенсивності та доступного обладнання
        high_intensity_cardio_no_equipment = [
            ("Високоінтенсивний інтервальний біг", 14, []),
            ("Інтенсивні стрибки", 15, []),
            ("Спринтерські інтервали", 16, []),
            ("Бурпі", 18, [])
        ]

        high_intensity_cardio_with_equipment = [
            ("Інтенсивна робота на еліптичному тренажері", 14, ["тренажери"]),
            ("Інтенсивні стрибки на скакалці", 15, ["скакалка"]),
            ("Інтенсивне тренування на велотренажері", 14, ["тренажери"])
        ]

        medium_intensity_cardio_no_equipment = [
            ("Біг підтюпцем", 12, []),
            ("Аеробіка", 11, [])
        ]

        medium_intensity_cardio_with_equipment = [
            ("Стрибки на скакалці", 13, ["скакалка"]),
            ("Велосипед з середньою інтенсивністю", 10, ["тренажери"]),
            ("Степ-аеробіка", 12, ["степ-платформа"])
        ]

        low_intensity_cardio_no_equipment = [
            ("Ходьба в швидкому темпі", 8, []),
            ("Легкий біг", 10, [])
        ]

        low_intensity_cardio_with_equipment = [
            ("Велосипед з низькою інтенсивністю", 8, ["тренажери"]),
            ("Еліптичний тренажер", 9, ["тренажери"]),
            ("Легка аеробіка зі скакалкою", 8, ["скакалка"])
        ]

        # Фільтруємо вправи за доступним обладнанням
        available_equipment_lower = [eq.lower() for eq in user.available_equipment]

        # Доступні вправи високої інтенсивності (спочатку додаємо вправи без обладнання)
        available_high_intensity = high_intensity_cardio_no_equipment.copy()
        for ex in high_intensity_cardio_with_equipment:
            if all(eq.lower() in available_equipment_lower for eq in ex[2]):
                available_high_intensity.append(ex)

        # Доступні вправи середньої інтенсивності
```

```
        available_medium_intensity = medium_intensity_cardio_no_equipment.copy()
        for ex in medium_intensity_cardio_with_equipment:
            if all(eq.lower() in available_equipment_lower for eq in ex[2]):
                available_medium_intensity.append(ex)

        # Доступні вправи низької інтенсивності
        available_low_intensity = low_intensity_cardio_no_equipment.copy()
        for ex in low_intensity_cardio_with_equipment:
            if all(eq.lower() in available_equipment_lower for eq in ex[2]):
                available_low_intensity.append(ex)

        # Вибір вправи відповідно до інтенсивності
        if intensity.lower() == "дуже висока":
            cardio_choice = random.choice(available_high_intensity)
        elif intensity.lower() == "висока":
            # Додаємо кілька вправ з високою інтенсивністю до середніх
            if available_high_intensity:
                cardio_choice = random.choice(available_medium_intensity + [available_high_intensity[0]])
            else:
                cardio_choice = random.choice(available_medium_intensity)
        elif intensity.lower() == "середня":
            cardio_choice = random.choice(available_medium_intensity)
        else:
            cardio_choice = random.choice(available_low_intensity)

        cardio_name, cal_per_min, required_equipment = cardio_choice

        cardio = Exercise(
            name=cardio_name,
            type="кардіо",
            difficulty=4,
            muscle_groups=["ноги", "серцево-судинна система"],
            calories_per_minute=cal_per_min,
            description="Кардіо тренування для досягнення цільової калорійності",
            equipment=required_equipment
        )

        # Розраховуємо потрібну тривалість (не менше 5 хв і не більше 45 хв)
        needed_duration = max(5, min(45, int(calories_difference / cal_per_min)))

        # Додаємо вправу
        exercises_for_workout.append((cardio, 1, needed_duration))




    return Workout(


        name=workout_name,
        exercises=exercises_for_workout,
        duration=duration,
        intensity=intensity,
        day_of_week=day_of_week
    )

def generate_plan(self,
                  user: User,
                  start_date: datetime.date,
                  end_date: datetime.date,
                  custom_workouts_per_week: int = None) -> WorkoutPlan:
    """
```

```
        Генерація плану тренувань з адаптивним навантаженням
        та лише одним днем відпочинку на тиждень.

        Чим довший план, тим менша початкова інтенсивність для досягнення
        таргетних значень з калорій до кінця плану.

        Args:
            user: Об'єкт користувача
            start_date: Дата початку плану
            end_date: Дата закінчення плану
            custom_workouts_per_week: Користувацьке значення кількості тренувань на тиждень (не враховується)

        Returns:
            Об'єкт плану тренувань
        """
        # Переконуємося, що початкова і кінцева дати вказано правильно
        if start_date > end_date:
            start_date, end_date = end_date, start_date

        # Розрахунок тривалості плану в тижнях
        days_duration = (end_date - start_date).days + 1
        weeks_duration = days_duration // 7 + (1 if days_duration % 7 > 0 else 0)

        # Базова складність залежно від рівня підготовки
        max_difficulty = self._select_difficulty_level(user.fitness_level)

        # --- РОЗРАХУНОК ЗАГАЛЬНОЇ МЕТИ ПЛАНУ ПО КАЛОРІЯМ ---
        # Базальний рівень метаболізму - скільки калорій потрібно в день для підтримки ваги
        daily_calorie_needs = user.basal_metabolic_rate() * 1.2  # Базовий рівень з мінімальною активністю

        target_calories_per_day = 0
        weight_goal_description = None
        plan_intensity = "середня"  # Базова інтенсивність плану

        # Обчислення необхідного дефіциту/профіциту калорій залежно від мети
        if "схуднення" in [goal.lower() for goal in user.goals]:
            # 1 кг жиру = приблизно 7700 ккал
            if user.target_weight is not None and user.weight > user.target_weight:
                weight_difference = user.weight - user.target_weight  # кг для схуднення
                total_calories_deficit = weight_difference * 7700  # загальний дефіцит

                # Розподіляємо дефіцит на всі дні плану
                daily_deficit = total_calories_deficit / days_duration
                weekly_weight_loss = weight_difference / weeks_duration

                # Визначаємо інтенсивність плану на основі тижневої втрати ваги
                if weekly_weight_loss > 2.0:
                    plan_intensity = "дуже висока"
                    daily_deficit_cap = 1500  # Максимальний дефіцит для дуже високої інтенсивності
                elif weekly_weight_loss > 1.0:
                    plan_intensity = "висока"
                    daily_deficit_cap = 1000  # Максимальний дефіцит для високої інтенсивності
                elif weekly_weight_loss > 0.5:
                    plan_intensity = "середня"
                    daily_deficit_cap = 750  # Максимальний дефіцит для середньої інтенсивності
                else:
                    plan_intensity = "низька"
                    daily_deficit_cap = 500  # Максимальний дефіцит для низької інтенсивності

                # Обмежуємо дефіцит для безпеки (не більше визначеного ліміту на день)
                if daily_deficit > daily_deficit_cap:
```

```
                # Якщо план закороткий для безпечного схуднення
                daily_deficit = daily_deficit_cap
                actual_weight_loss = (daily_deficit * days_duration) / 7700
                actual_weekly_loss = (daily_deficit * 7) / 7700

                weight_goal_description = f"УВАГА! Ваша ціль надто амбіційна — {weight_difference:.1f} кг за {weeks_duration} тижнів " \
                                          f"(це {weekly_weight_loss:.1f} кг/тиждень). Обмежено до {actual_weight_loss:.1f} кг " \
                                          f"({actual_weekly_loss:.1f} кг/тиждень). " \
                                          f"Рекомендований термін: {int(weight_difference)} тижнів."
            else:
                weekly_rate = weight_difference / weeks_duration
                if weekly_rate > 0.75:
                    weight_goal_description = f"Ціль: зниження ваги на {weight_difference:.1f} кг за {weeks_duration} тижнів " \
                                              f"({weekly_rate:.1f} кг/тиждень). Це інтенсивний темп схуднення."
                else:
                    weight_goal_description = f"Ціль: здорове зниження ваги на {weight_difference:.1f} кг за {weeks_duration} тижнів " \
                                              f"({weekly_rate:.1f} кг/тиждень)."

                # Визначаємо інтенсивність залежно від дефіциту
                if daily_deficit < 300:
                    plan_intensity = "низька"
                    target_calories_per_day = 300  # Мінімальні витрати калорій
                elif daily_deficit < 500:
                    plan_intensity = "помірна"
                    target_calories_per_day = 400
                elif daily_deficit < 750:
                    plan_intensity = "середня"
                    target_calories_per_day = 500
                else:
                    plan_intensity = "висока"
                    target_calories_per_day = 600  # Високі витрати калорій для значного дефіциту
            else:
                # Стандартний дефіцит без конкретної цілі
                daily_deficit = 500  # Стандартний дефіцит для схуднення
                weight_goal_description = f"Ціль: зниження ваги із дефіцитом {daily_deficit} ккал/день"
                plan_intensity = "середня"
                target_calories_per_day = daily_deficit


        elif "набір м'язової маси" in [goal.lower() for goal in user.goals]:
            # Для набору маси - створюємо профіцит калорій
            if user.target_weight is not None and user.weight < user.target_weight:
                weight_difference = user.target_weight - user.weight  # кг для набору
                # Для набору 1 кг м'язової маси потрібно більше калорій (і більше часу)
                total_calories_surplus = weight_difference * 9000  # приблизна формула
                # Розподіляємо профіцит на всі дні плану
                daily_surplus = total_calories_surplus / days_duration
                # Обмежуємо профіцит для якісного набору (не більше 500 ккал/день)
                if daily_surplus > 500:
                    daily_surplus = 500
                    actual_weight_gain = (daily_surplus * days_duration) / 9000
                    weekly_weight_gain = (daily_surplus * 7) / 9000

                weight_goal_description = f"УВАГА! Ваша ціль надто амбіційна — набір {weight_difference:.1f} кг за {weeks_duration} тижнів " \
                                          f"(це {weight_difference / weeks_duration:.1f} кг/тиждень). Обмежено до {actual_weight_gain:.1f} кг " \
```

```
                                f"({weekly_weight_gain:.1f} кг/тиждень). Рекомендований термін:
{int(weight_difference * 2)} тижнів " \
                                f"для якісного набору м'язової маси без надлишку жиру."

            else:
                weekly_rate = weight_difference / weeks_duration
                if weekly_rate > 0.4:
                    weight_goal_description = f"Ціль: набір м'язової маси на {weight_difference:.1f} кг за
{weeks_duration} тижнів " \
                                      f"({weekly_rate:.1f} кг/тиждень). Це інтенсивний темп набору маси."

                else:
                    weight_goal_description = f"Ціль: здоровий набір м'язової маси на {weight_difference:.1f} кг за
{weeks_duration} тижнів " \
                                  f"({weekly_rate:.1f} кг/тиждень)."

            # Визначаємо інтенсивність залежно від профіциту
            if daily_surplus < 200:
                plan_intensity = "помірна"
            elif daily_surplus < 350:
                plan_intensity = "середня"
            else:
                plan_intensity = "висока"

            # Цільові витрати калорій на тренуваннях - чим більше профіцит, тим вище інтенсивність
            target_calories_per_day = 300 + daily_surplus * 0.5
        else:
            # Стандартний профіцит без конкретної цілі
            daily_surplus = 300
            weight_goal_description = f"Ціль: набір м'язової маси з профіцитом {daily_surplus} ккал/день"
            plan_intensity = "середня"
            target_calories_per_day = 400  # Стандартне значення

    else:
        # Для інших цілей - підтримка або загальне здоров'я
        if "витривалість" in [goal.lower() for goal in user.goals]:
            weight_goal_description = "Ціль: підвищення витривалості та загального фітнесу"
            plan_intensity = "середня"
            target_calories_per_day = 450
        else:
            weight_goal_description = "Ціль: підтримка форми та загальне здоров'я"
            plan_intensity = "помірна"
            target_calories_per_day = 350

        if weeks_duration <= 2:
            # Для дуже коротких планів підвищуємо інтенсивність
            if plan_intensity == "низька":
                plan_intensity = "помірна"
                target_calories_per_day *= 1.2
            elif plan_intensity == "помірна":
                plan_intensity = "середня"
                target_calories_per_day *= 1.15
            elif plan_intensity == "середня":
                plan_intensity = "висока"
                target_calories_per_day *= 1.1

            # Додамо попередження в опис
            weight_goal_description = f"{weight_goal_description} (Короткостроковий план - підвищена
інтенсивність)"

    workouts = []
```

```
# Використовуємо поточний час для унікальності кожного плану
current_timestamp = int(datetime.datetime.now().timestamp())
unique_seed = current_timestamp + weeks_duration * 1000 + len(user.name) * 10

# Заздалегідь визначаємо день відпочинку для кожного тижня
rest_days_by_week = {}
for week in range(weeks_duration):
    random.seed(unique_seed + week * 100 + user.age)
    rest_days_by_week[week] = random.randint(0, 6)  # Випадковий день тижня (0-6)

# Визначаємо наявне обладнання
available_equipment = [eq.lower() for eq in user.available_equipment]
has_weights = any(eq in available_equipment for eq in ["гантелі", "штанга"])
has_machines = "тренажери" in available_equipment
has_bodyweight_equipment = any(eq in available_equipment for eq in
                               ["турнік/бруси", "скакалка", "фітнес-резинки"])

# Визначаємо обмеження здоров'я
limitations = [lim.lower() for lim in user.limitations]
has_back_issues = "проблеми зі спиною" in limitations
has_knee_issues = "проблеми з коліном" in limitations
has_shoulder_issues = "проблеми з плечем" in limitations

# Визначення шаблонів тренувань зі змінним розподілом по тижнях
workout_templates = {
    "схуднення": {
        "з_тренажерами": [
            # тиждень 1
            {0: "кардіо", 1: "все тіло", 2: "кардіо", 3: "нижня частина тіла",
             4: "кардіо", 5: "верхня частина тіла", 6: "відпочинок"},
            # тиждень 2
            {0: "все тіло", 1: "кардіо", 2: "нижня частина тіла", 3: "кардіо",
             4: "верхня частина тіла", 5: "кардіо", 6: "відпочинок"},
            # тиждень 3
            {0: "нижня частина тіла", 1: "верхня частина тіла", 2: "кардіо",
             3: "все тіло", 4: "кардіо", 5: "нижня частина тіла", 6: "відпочинок"},
            # тиждень 4
            {0: "кардіо", 1: "верхня частина тіла", 2: "кардіо",
             3: "нижня частина тіла", 4: "все тіло", 5: "кардіо", 6: "відпочинок"}
        ],
        "з_гантелями": [
            # тиждень 1
            {0: "кардіо", 1: "верхня частина тіла", 2: "кардіо",
             3: "нижня частина тіла", 4: "кардіо", 5: "все тіло", 6: "відпочинок"},
            # тиждень 2
            {0: "верхня частина тіла", 1: "кардіо", 2: "нижня частина тіла",
             3: "кардіо", 4: "все тіло", 5: "кардіо", 6: "відпочинок"},
            # тиждень 3
            {0: "кардіо", 1: "все тіло", 2: "кардіо",
             3: "верхня частина тіла", 4: "нижня частина тіла", 5: "кардіо", 6: "відпочинок"}
        ],
        "з_власною_вагою": [
            # тиждень 1
            {0: "кардіо", 1: "все тіло", 2: "кардіо",
             3: "все тіло", 4: "кардіо", 5: "все тіло", 6: "відпочинок"},
            # тиждень 2
            {0: "все тіло", 1: "кардіо", 2: "все тіло",
             3: "кардіо", 4: "все тіло", 5: "кардіо", 6: "відпочинок"}
        ]
    },
    "набір_маси": {
        "з_тренажерами": [
```

```
            # тиждень 1
            {0: "верхня частина тіла", 1: "нижня частина тіла", 2: "верхня частина тіла",
             3: "нижня частина тіла", 4: "все тіло", 5: "кардіо", 6: "відпочинок"},
            # тиждень 2
            {0: "нижня частина тіла", 1: "верхня частина тіла", 2: "нижня частина тіла",
             3: "верхня частина тіла", 4: "кардіо", 5: "все тіло", 6: "відпочинок"},
            # тиждень 3
            {0: "верхня частина тіла", 1: "нижня частина тіла", 2: "все тіло",
             3: "верхня частина тіла", 4: "нижня частина тіла", 5: "кардіо", 6: "відпочинок"}
        ],
        "без_тренажерів": [
            # тиждень 1
            {0: "все тіло", 1: "все тіло", 2: "кардіо",
             3: "все тіло", 4: "все тіло", 5: "кардіо", 6: "відпочинок"},
            # тиждень 2
            {0: "все тіло", 1: "кардіо", 2: "все тіло",
             3: "все тіло", 4: "кардіо", 5: "все тіло", 6: "відпочинок"}
        ]
    },
    "загальне": [
        # тиждень 1
        {0: "все тіло", 1: "кардіо", 2: "все тіло",
         3: "кардіо", 4: "все тіло", 5: "кардіо", 6: "відпочинок"},
        # тиждень 2
        {0: "кардіо", 1: "все тіло", 2: "кардіо",
         3: "все тіло", 4: "кардіо", 5: "все тіло", 6: "відпочинок"}
    ]
}
```

ДОДАТОК Г

ГРАФІЧНА ЧАСТИНА

«ПРОГРАМНИЙ МОДУЛЬ ІНДИВІДУАЛЬНОГО ПЛАНУВАННЯ СПОРТИВНИХ ТРЕНУВАНЬ»

Виконав: студент 4 курсу, групи 2КН-21б
Спеціальності 122 – Комп'ютерні науки
(шифр і назва напрямку підготовки, спеціальності)
Олександр ВЛАСОК
(прізвище та ініціали)
Керівник: д.т.н., проф. каф. КН
Ярослав ІВАНЧУК
(прізвище та ініціали)

«03» червня 2025 р.

Вінниця ВНТУ – 2025 рік

Рисунок Г.1 – Загальна архітектура програмного модуля індивідуального планування спортивних тренувань

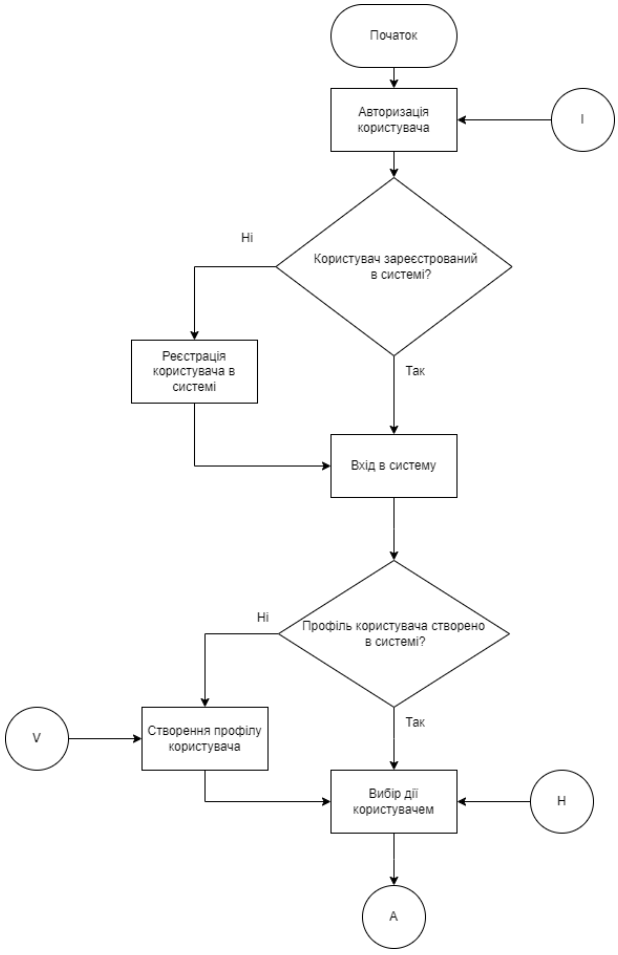

Рисунок Г.2 – Схема загального алгоритму роботи програмного модуля індивідуального планування спортивних тренувань

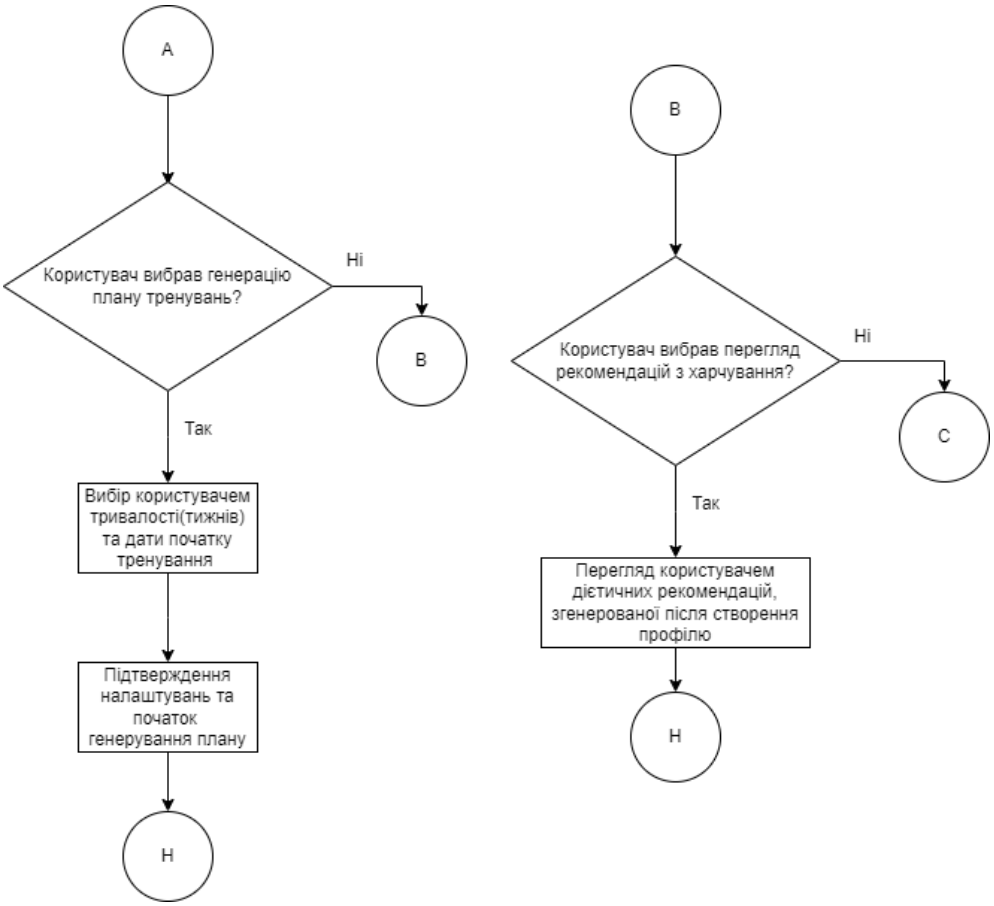

Рисунок Г.2 – Продовження

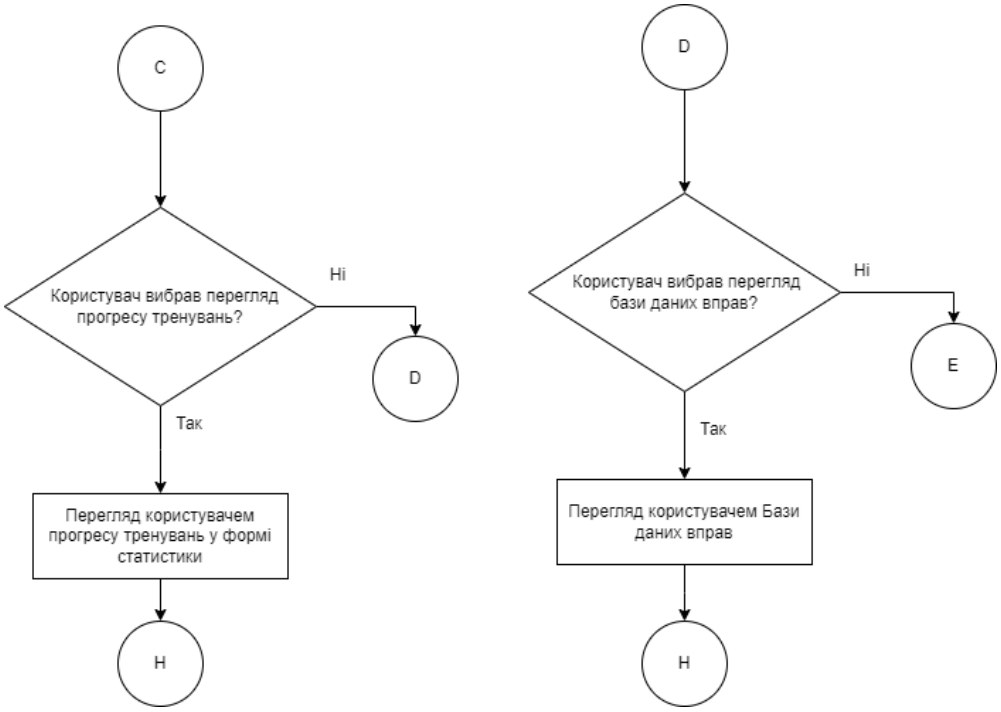

Рисунок Г.2 – Продовження

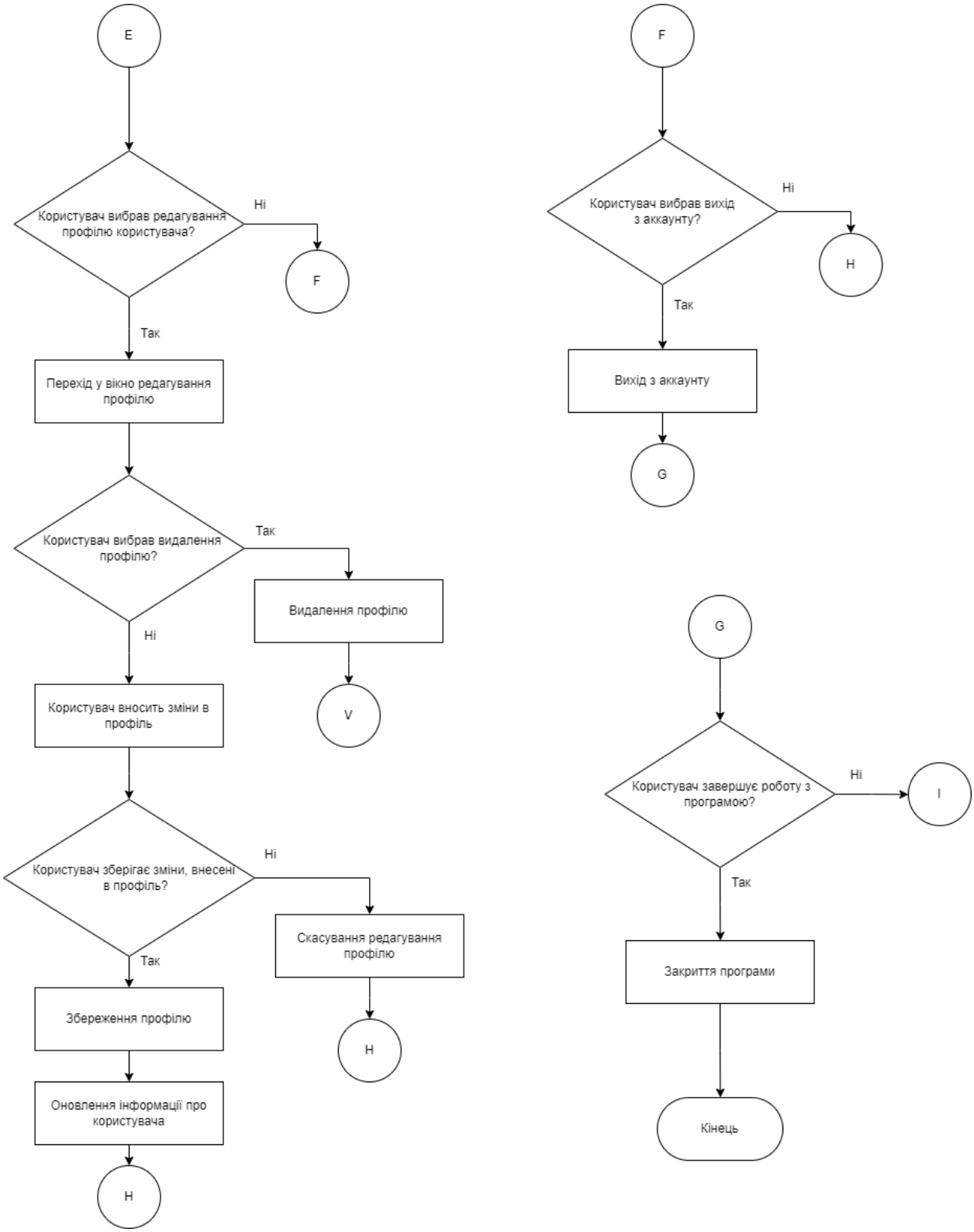

Рисунок Г.2 – Продовження

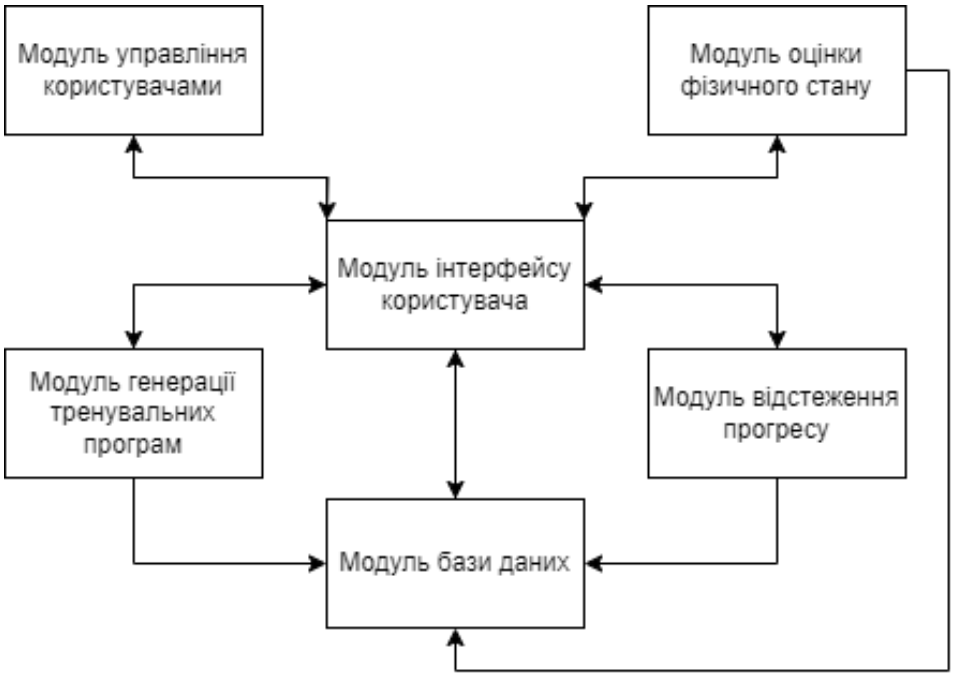

Рисунок Г.3 – Структурна схема програмного модуля індивідуального планування спортивних тренувань

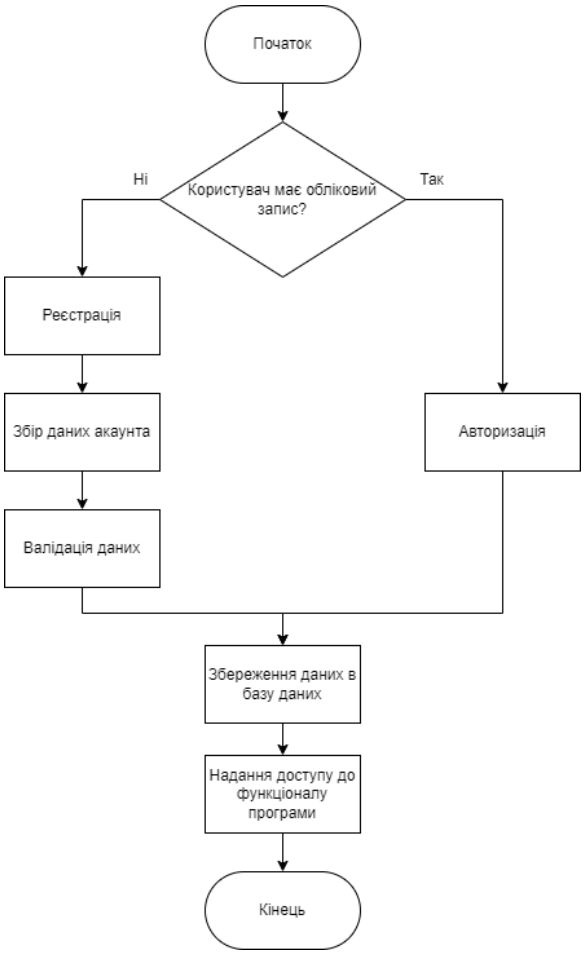

Рисунок Г.4 – Схема алгоритму роботи модуля управління користувачами

Рисунок Г.5 – Схема алгоритму роботи модуля оцінки фізичного стану

Рисунок Г.6 – Схема алгоритму роботи модуля генерації тренувальних програм

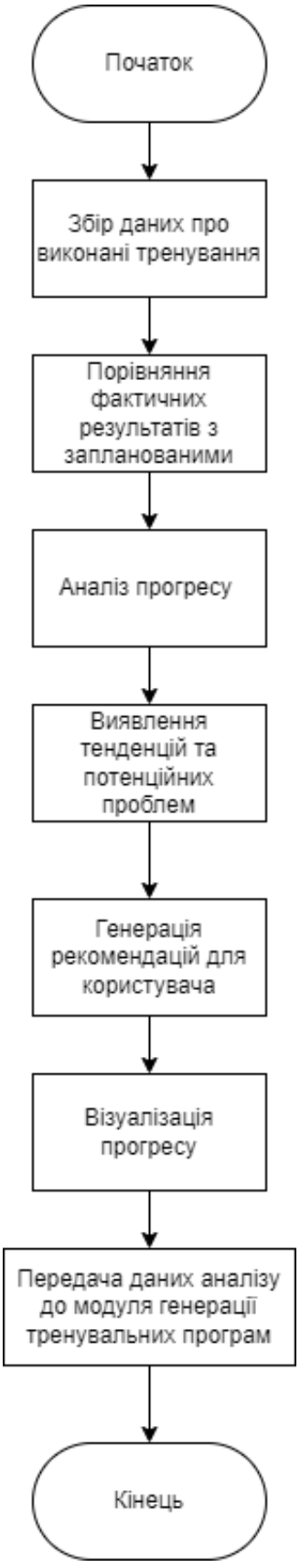

Рисунок Г.7 – Схема алгоритму роботи модуля відстеження прогресу

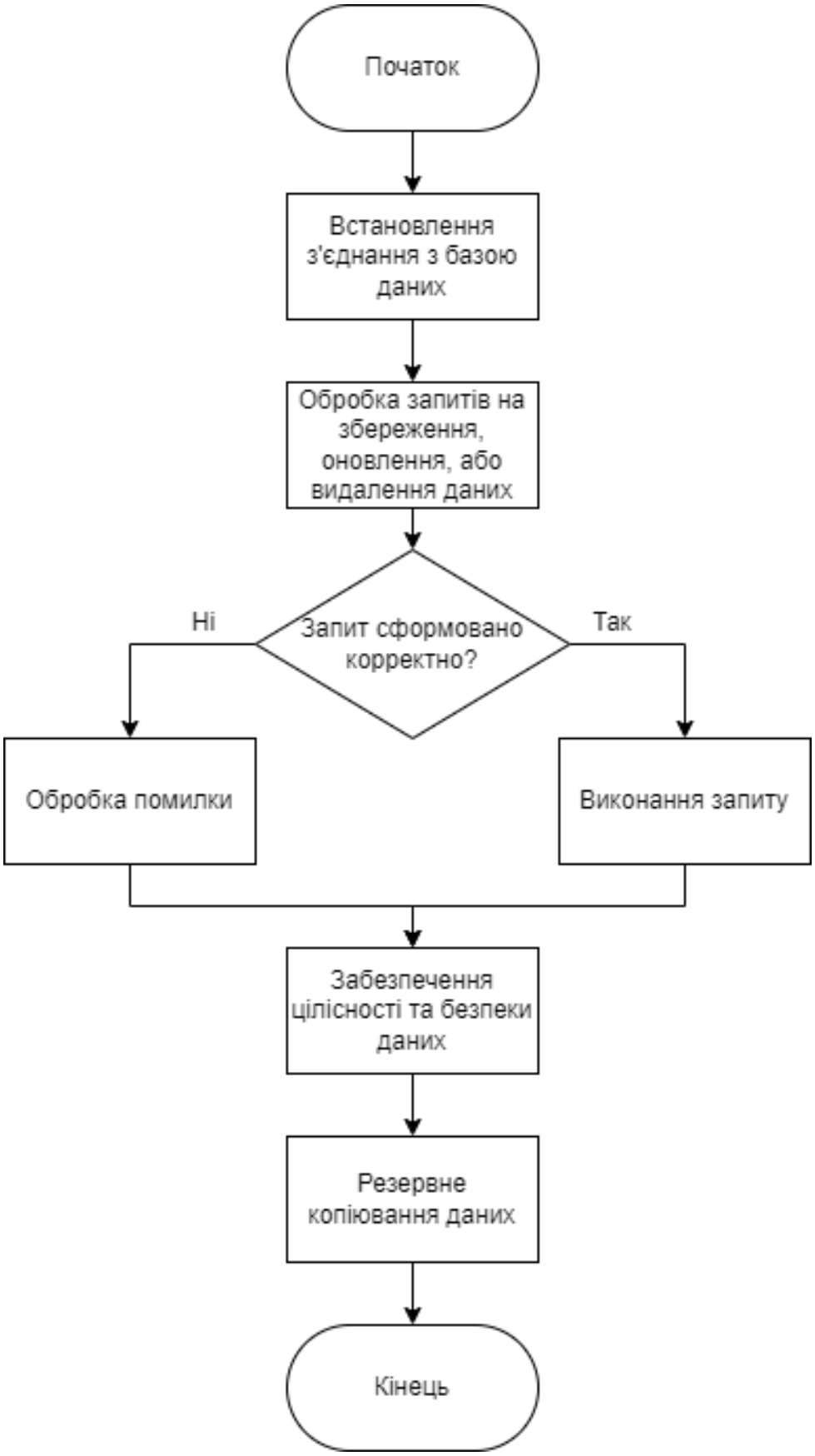

Рисунок Г.8 – Схема алгоритму роботи модуля бази даних

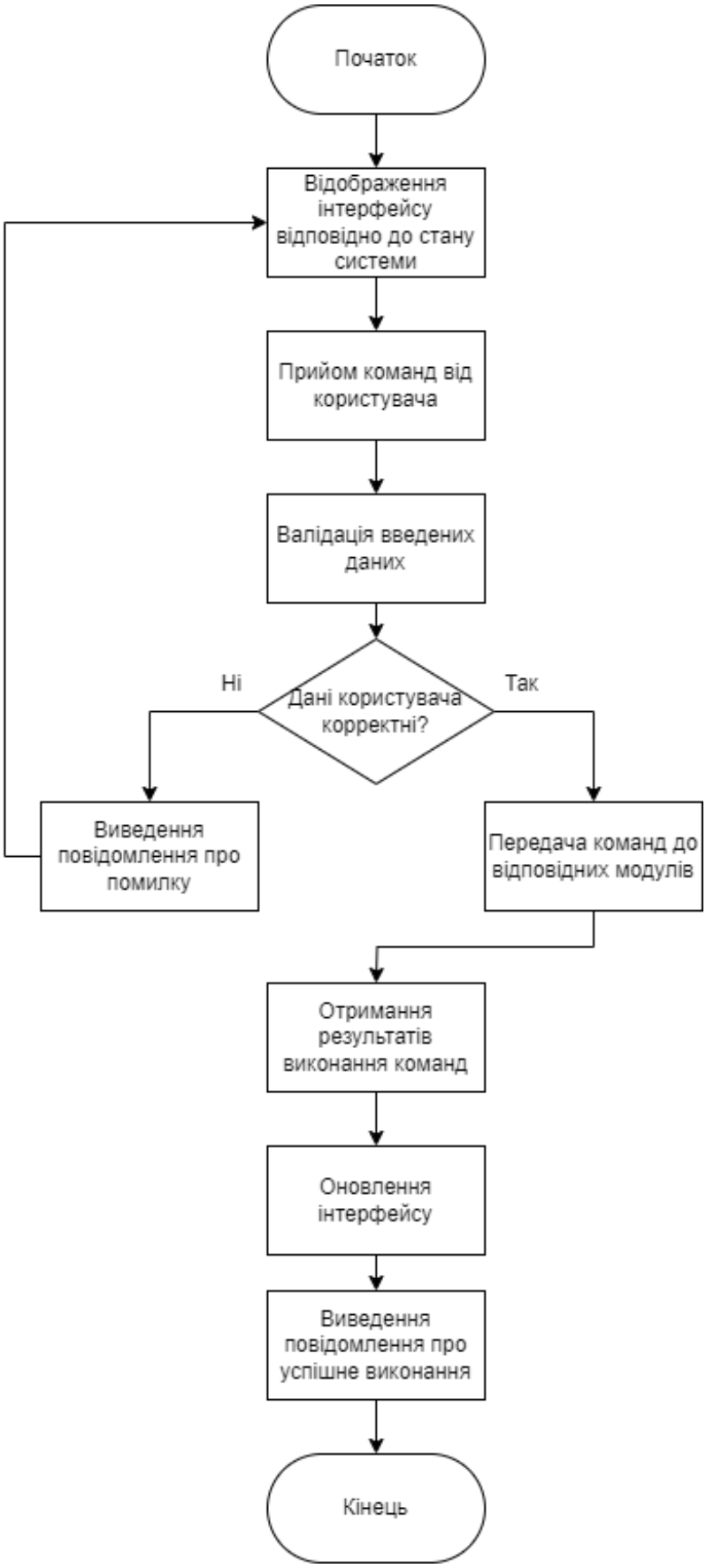

Рисунок Г.9 – Схема алгоритму роботи модуля інтерфейсу користувача

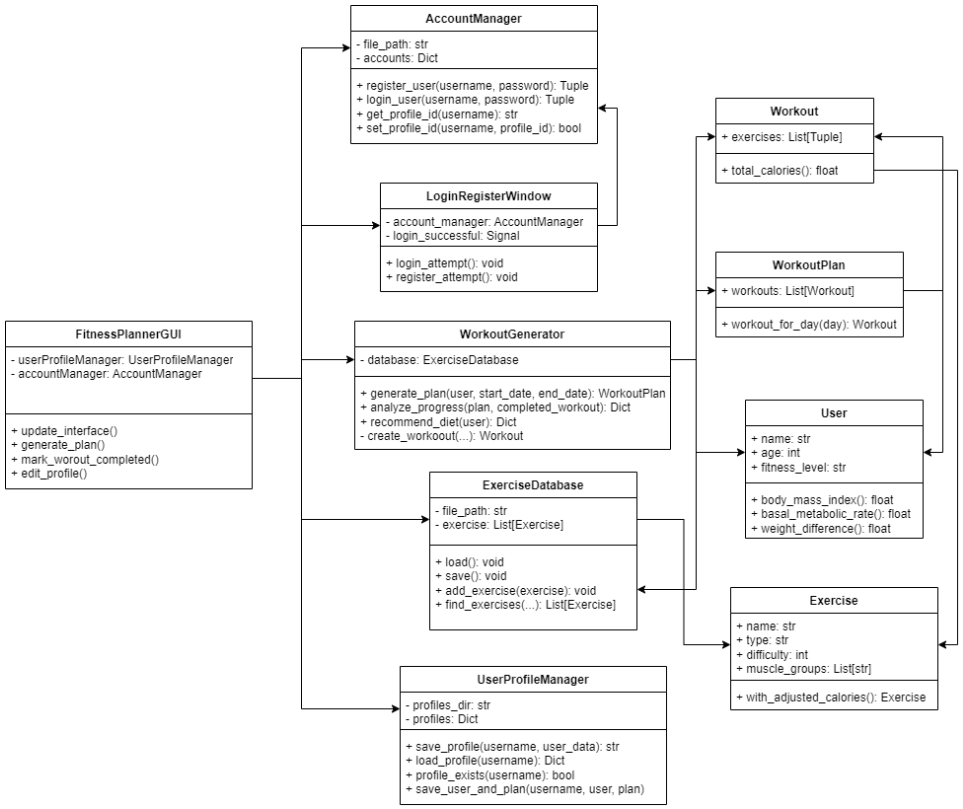

Рисунок Г.10 – UML-діаграма класів системи індивідуального планування спортивних тренувань

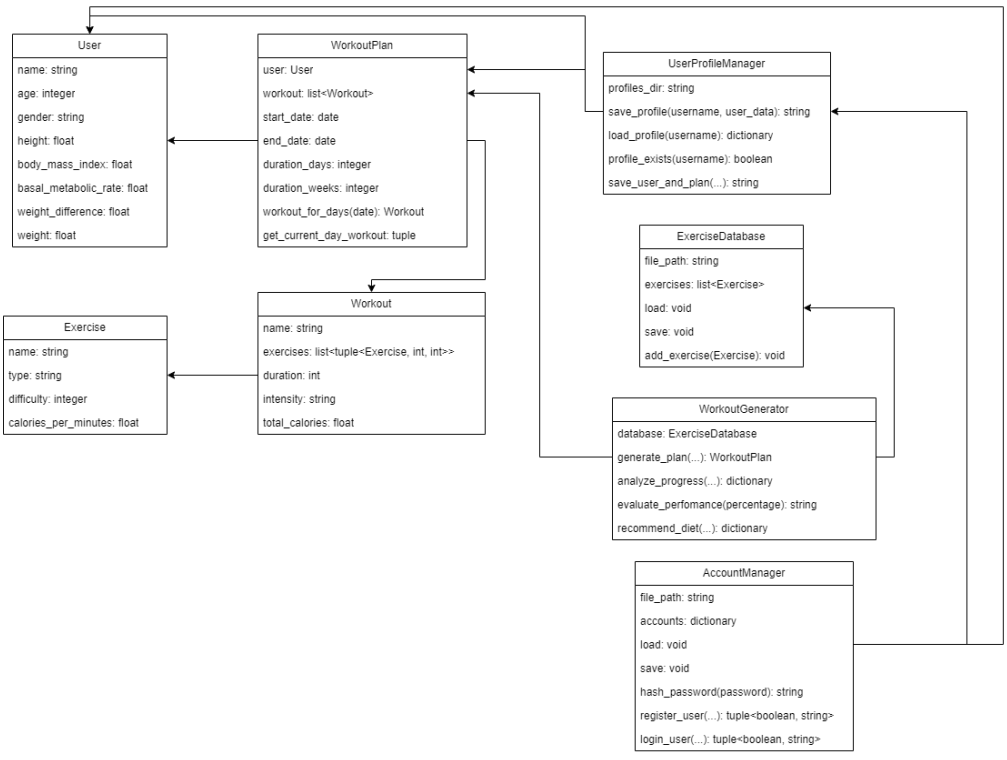

Рисунок Г.11 – Схема Об'єктно-орієнтованої моделі даних предметної області "Індивідуальне планування спортивних тренувань"

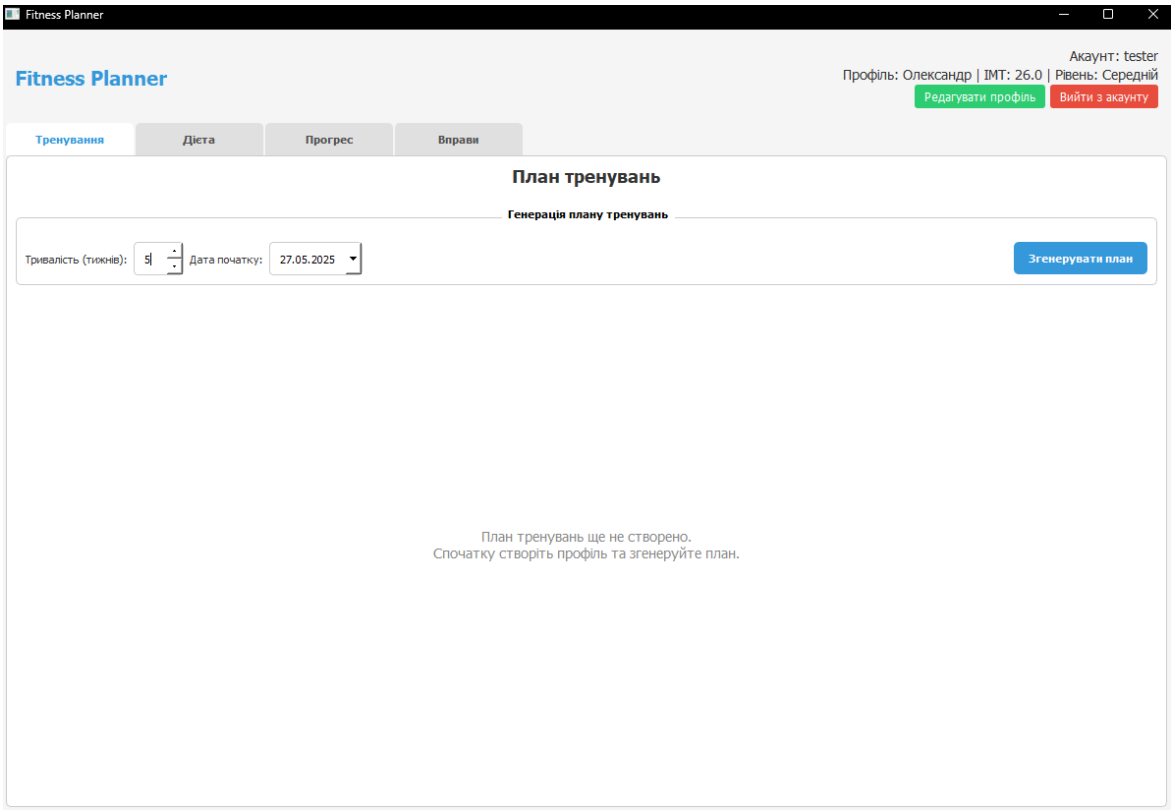

Рисунок Г.12 – Загальний вигляд інтерфейсу вкладки «Тренування» до генерації індивідуального плану тренування