

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Веб-технології»)

на тему:

«Вебплатформа для фітнес-тренувань»

Виконав студент групи ВТ-21-1
ГОДЛЕВСЬКИЙ Олег Олегович

Керівник роботи:
ФАНТ Микола Олександрович

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк
«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ГОДЛЕВСЬКИЙ Олег Олегович**

Керівник роботи: **ФАНТ Микола Олександрович**

Тема роботи: **«Вебплатформа для фітнес-тренувань»**,
затверджена Наказом закладу вищої освіти №61/с від 14.02.2025р.

Вихідні дані для роботи: **методи збору, обробки та відображення тренувальних даних, побудова бібліотеки вправ під різні фітнес-цілі. Інтеграція фітнес-браслетів з вебдодатком для відстеження тренувань і вибору вправ.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Фант М.О.	27.02.2025р.	27.02.2025р.
2	Фант М.О.	27.02.2025р.	27.02.2025р.
3	Фант М.О.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра являє «Вебплатформу для фітнес-тренувань». Пояснювальна записка до кваліфікаційної роботи містить 62 сторінок друкованого тексту, 10 сторінок додатків 45 ілюстрацій та 12 таблиць. Метою роботи є створення вебплатформи для фітнес-трекінгу з інтеграцією Strava та бібліотекою вправ, використовуючи NestJS, NextJS і PostgreSQL.

В роботі визначені головні завдання для розробки додатку, проаналізовано методи розв'язання поставлених задач, обґрунтовано вибір структури додатку, наведено алгоритм його роботи, а також архітектуру та взаємодію його основних компонентів. Також була описана програмна реалізація програми. Крім цього, у роботі було розглянуто основні аспекти забезпечення безпеки даних для користувачів та аторизації. Також приділено увагу зовнішнім сервісам, таким як Strava API, яка дозволяє отримувати дані в режимі реального часу. Проведено тестування функціональності платформи на різних пристроях та визначено що можна покращити та додати щось нове. Загалом, робота платформи підтверджує її стабільність і готовність до використання.

Ключові слова: ВЕБПЛАТФОРМА, NEXT.JS, PRISMA, POSTGRESQL, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА.

					ІПЗ.КР.Б-121-25-ІПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата	Вебплатформа для фітнес-тренувань Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		О. Годлевський						
Керівник		М. ФАНТ					3	
н. контр.						Житомирська політехніка, група ВТ-21-1		
Зав. каф.								

ABSTRACT

The bachelor's qualification thesis presents a «Web Platform for Fitness Training». The explanatory note to the qualification work contains 62 pages of printed text, 10 pages of appendices, 45 illustrations, and 12 tables. The aim of the thesis is to develop a web platform for fitness tracking with Strava integration and an exercise library, using NestJS, NextJS, and PostgreSQL.

The work defines the main tasks for application development, analyzes methods for solving the given problems, justifies the choice of application structure, presents the algorithm of its operation, as well as the architecture and interaction of its main components. The software implementation of the application is also described. In addition, the thesis explores key aspects of user data security and authorization. Attention is also given to external services such as the Strava API, which enables real-time data retrieval. The functionality of the platform was tested on different devices, and areas for improvement and potential new features were identified. Overall, the platform's performance confirms its stability and readiness for use.

Keywords: WEB PLATFORM, NEXT.JS, PRISMA, POSTGRESQL, CLIENT-SERVER ARCHITECTURE.

		Годлевський О.О.			ІІЗ.КР.Б-121-25-ІІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ.....	5
ABSTRACT	4
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБПЛАТФОРМИ ДЛЯ ФІТНЕС-ТРЕНУВАНЬ	9
1.1 Постановка задачі.....	9
1.2 Аналіз аналогів програмного продукту	10
1.3 Вибір архітектури застосунку	14
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	16
Висновок до першого розділу.....	18
РОЗДІЛ 2. ПРОЕКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ	19
2.1. Визначення варіантів використання і структури застосунку	19
2.2 Проектування бази даних системи	24
2.3. Проектування та реалізація алгоритмів роботи системи	30
2.4 Реалізація функціоналу додатку	33
Висновки до другого розділу	35
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ЗАСТОСУНКОМ	37
3.1. Порядок встановлення та налаштування параметрів системи.	37
3.2 Структура інтерфейсу вебплатформи	39
3.3 Тестування роботи програмного комплексу	57
Висновки до третього розділу.....	59
ВИСНОВКИ	60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	61
ДОДАТКИ.....	64

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

DB – база даних, що використовується для зберігання та обробки інформації.

CRUD – набір базових операцій із даними: створення (Create), читання (Read), оновлення (Update) та видалення (Delete).

ORM – об'єктно-реляційне відображення, що спрощує взаємодію з базою даних за допомогою об'єктів програмування.

API – інтерфейс взаємодії між програмними компонентами, що дозволяє обмінюватися даними між клієнтом і сервером.

JWT – JSON Web Token, стандарт для безпечної передачі інформації між сторонами у форматі JSON.

SEO – оптимізація вебресурсу для покращення видимості в пошукових системах.

SSR – рендеринг на сервері перед передачею сторінки у браузер користувача, що покращує продуктивність і SEO.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми: фізична активність та збереження здоров'я є важливою складовою життя сучасної людини. З ростом популярності цифрових технологій та інтернету користувачі починають сприймати вебдодатки, як потужний інструмент для планування, організації та моніторингу своїх тренувань з використанням персонального комп'ютера чи іншого пристрою. Вебдодатки для тренувань надають зручний інтерфейс, що дозволяє користувачеві створювати свої індивідуальні програми на обрану кількість днів, стежити за виконання запланованих навантажень та організовувати тренувальний процес. Результатом роботи має стати виконаний вебдодаток для тренувань, що забезпечить функціональність створення тренувальних планів, складання розкладу тренувань реєстрації та входу користувача та інтеграцію з фітнес-браслетами. Таким чином, розроблення вебдодатку для тренувань є актуальним завданням, що відповідає потребам користувачів у персоналізованому підході до фізичних навантажень. Це особливо важливо в умовах зростання зацікавленості у здоровому способі життя та самостійних заняттях спортом.

Метою даної роботи є розробка функціонального вебдодатку для фітнес-тренувань, який дозволить користувачам планувати, та аналізувати свої тренування.

Для досягнення результату слід вирішити такі **завдання**:

1. провести аналіз існуючих аналогів та визначити їхні плюси та мінуси;
2. розробити структуру платформи, яка буде відповідати клієнт-серверній архітектурі;
3. розробити платформу з функціями створення планів, відображення та взаємодію з бібліотекою вправ;
4. інтегрувати платформу з фітнес-браслетами для збору даних;
5. виконати комплексне тестування платформи;
6. оцінити стабільність роботи;

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	7
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Об’єктом дослідження є реалізація програмного продукту «Вебдодаток для фітнес-тренувань».

Предметом дослідження є методи та інструменти створення вебдодатків для фітнес-тренувань.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБПЛАТФОРМИ ДЛЯ ФІТНЕС-ТРЕНУВАНЬ

1.1 Постановка задачі

У випускній кваліфікаційній роботі потрібно розробити вебплатформу для фітнес-тренувань, яка допоможе користувач тренуватися на основі бібліотеки вправ та можливості створення або генерування планів, також отримання даних щодо своїх тренувань одразу з фітнес-браслету використовуючи сервіс – Strava. Платформа повинна складатися з клієнтської та серверної архітектури.

Розробку платформи слід почати з архітектури, що включає аналіз вимог щодо функціональності, вибір технологічного стеку, проектування структури бази даних. Далі слід зосередитись над користувацьким інтерфейсом, а саме розробку ключових компонентів, меню навігації, функціонал щодо створення планів, відображення вправ.

Для клієнтської частини обрано фреймворк NextJs, він забезпечує зручну маршрутизацію, для UI/UX дизайну обрано Tailwind CSS, для компонентів використовується shadcn/ui. Уся розробка ведеться з TypeScript, що гарантує узгодження типів.

Сервер реалізується через фреймворк NestJs на основі NodeJs, для роботи з базою даних використовується ORM Prisma, яка працює з реляційними базами даних та забезпечує типізацію, керуванням бази даних обрано PostgreSQL.

Перед розгортанням, слід провести тестування та перевірку роботи компонентів та модулів системи, щоб уникнути неоптимізованої платформи, виконати роботу над помилка та знайти слабкі місця у системі.

Також платформа включає функцію синхронізації даних у реальному часі, це дозволить користувачам бачити результат тренувань одразу по закінченню активності, варто зауважити що для цього користувачам потрібно буде використовувати фітнес-браслет, який буде прив'язаний до акаунту в Strava. Окрім цього увага приділялася адаптивності інтерфейсу, задля забезпечення

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	9
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

зручності на різних пристроях, починаючи від смартфонів, закінчуючи ноутбуками та настільними комп'ютерами.

У майбутньому платформа буде підтримуватися та будуть з'являтися нові функції, які мають покращити досвід користувача, щодо використання платформи. Наприклад, планується створення функцій, щодо соціальних викликів, де звичайні користувачі зможуть змагатися між собою, отримувати нагороди за досягнення, відображення рейтингу, що зробить платформу та тренування більш інтерактивними та залучить більше користувачів до платформи.

Наступним етапом щодо розвитку платформи є впровадження штучного інтелекту для аналізу тренувальних планів, недавньої активності для надання персональних порад та рекомендацій. Таке впровадження зробить платформу унікальною та конкурентоспроможною на ринку.

На завершальному етапі планується створення мобільної версії платформи для доступу до додатку через IOS та Android. Це забезпечить розширення аудиторії та дозволить користувачам тренуватися при будь - яких умовах. При реалізації мобільної версії планується створити офлайн-режим, що забезпечить зручність користування.

Результатом виконання має стати платформа для фітнес-тренувань, яка надає користувачеві створення власних планів, перегляд вправ та аналіз фізичних активностей на основі Strava. Дана платформа має формувати звичку до фізичної активності, мотивуючи користувачів до регулярних занять.

1.2 Аналіз аналогів програмного продукту

Розглянемо наявні аналоги, які найбільше подібні до теми кваліфікаційної роботи: MyFitnessPal, Freeletics, Strava. Наведемо переваги та недоліки, та порівняти з майбутньою платформою.

MyFitnessPal – це популярна вебплатформа, призначена для трекінгу харчування, фізичної активності та загального стану здоров'я. Основна його перевага полягає в тому, що він дозволяє відстежувати щоденне споживання калорій, макронутрієнтів і вітамінів, а також фіксувати виконані тренування.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	10
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

База даних продуктів – налічує понад 18 мільйонів найменувань. Крім цього, MyFitnessPal легко синхронізується з іншими популярними додатками та фітнес-пристроями – такими як Fitbit або Garmin – що дозволяє автоматично отримувати інформацію про фізичну активність.

Зручністю користування особливо вирізняється мобільний додаток: він інтуїтивний, швидкий і набагато приємніший у використанні, ніж вебверсія, яка, перевантажена рекламою і викликає роздратування через свій застарілий та незрозумілий інтерфейс. На мою думку сайт збирає досить велику кількість інформації про користувача. Також варто зважати на те, що деякі розширені можливості – як-от перегляд спалених калорій або доступ до тренувальних програм – доступні лише в рамках платної підписки.

У цілому, MyFitnessPal справляє враження потужного інструменту для тих, хто серйозно ставиться до свого здоров'я та способу життя, хоча без певних компромісів у зручності й приватності не обходиться.

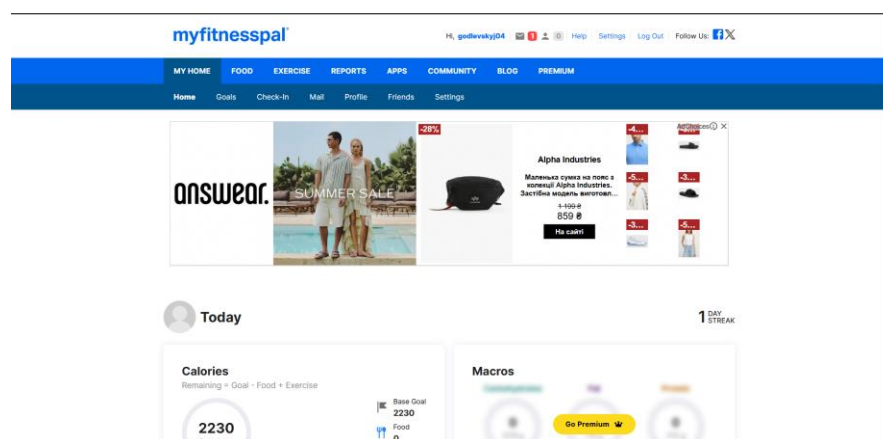


Рис. 1.1 – Головна сторінка MyFitnessPal

Freeletics – це цифровий фітнес-тренер, який використовує штучний інтелект для створення персоналізованих тренувальних програм. Користувач може сам обрати напрям, який його цікавить, чи то розвиток сили, витривалості, швидкості або ж загальної фізичної форми. Також є мобільних додаток, є змога ділитися досягненнями чи тренуваннями з друзями.

Як і в MyFitnessPal є безкоштовна та преміум підписка. Безкоштовна версія містить рекламу, яка часто відволікає. Головний мінус це те, що найголовніші

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	11
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

функції як – персоналізовані плани тренувань доступні лише за преміум підпискою.

Freeletics залишає враження потужного інструменту для тих, хто хоче тренуватися ефективно, хоча для повного доступу до всіх можливостей доведеться скористатися платною версією.

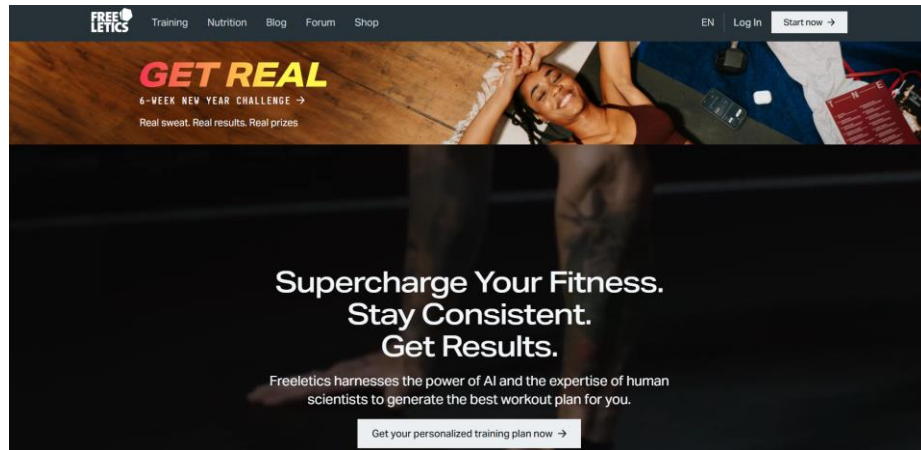


Рис. 1.2 – Головна сторінка Freeletics

Strava – платформа, яка можна сказати є соціальною мережею для любителів спорту. Попри фіксацію власної активності, платформа дозволяє поширювати та обмінюватися активністю з іншими учасниками. Платформа підтримує синхронізацію з популярними фітнес-пристроями, зокрема з Fitbit та Garmin.

Наявність челенджів, можливість залишати коментарі та лайки посправжньому робить платформу схожою на соціальну мережу. Це є перевагою серед інших конкурентів, адже ця атмосфера «змагання» приваблює нових користувачів та не дає іншим занудьгувати.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

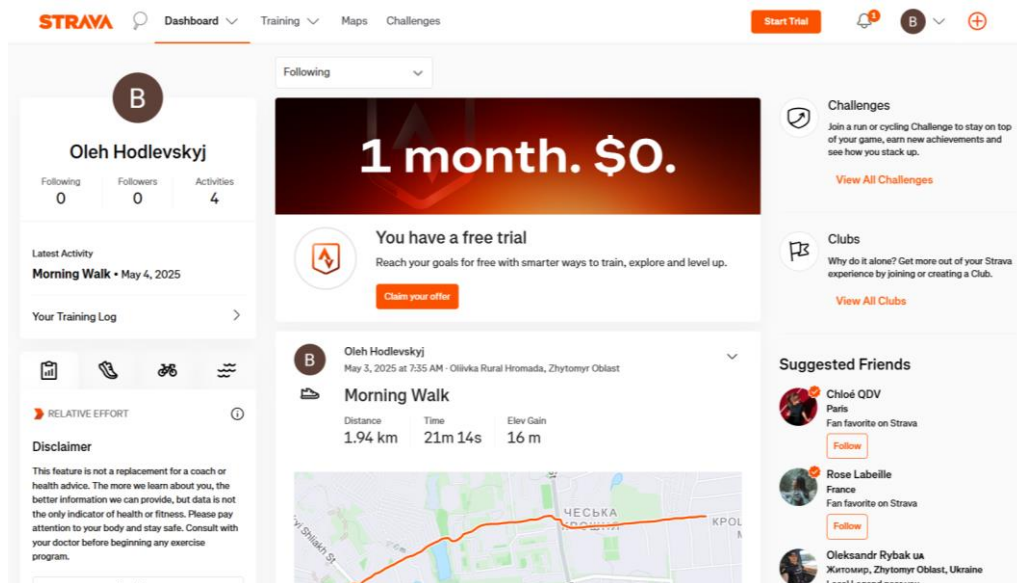


Рис. 1.3 – Головна сторінка Strava

Таблиця 1.1

Порівняльна таблиця наявних рішень на ринку

Критерій	Strava	MyFitnessPal	Freeletics	Розроблений застосунок
Основне призначення	Відстеження фізичної активності	Контроль харчування, калорій	Персоналізовані тренування(зал, вулиця)	Відстеження фізичної активності, персональні тренувальні плани
Тип активності що відстежується	Кардіо-активність, маршрути, темп, дистанція	Харчування, вода, вага, фізична активність	Фізичні вправи на силу, витривалість, гнучкість	Кардіо-активність, маршрути, темп, дистанція
Персоналізація тренувань	-	-	Використовується AI	+
Наявність мобільного додатку	+	+	+	-
Інтеграція з пристроями/сервісами	+	+	-	+
Наявність реклами	+	+	+	-
Безкоштовний функціонал	Обмежений	Обмежений	Обмежений	Необмежений
Зручність інтерфейсу	+	-	-	+

Аналіз існуючих аналогів, дозволив визначити сильні та слабкі сторони застосунків. Основними недоліками є обмежений та платний функціонал, велика

кількість реклами та незрозумілий інтерфейс, який впливає та перше враження від користування.

Ще однією проблемою є відсутність створення персональних планів або його наявність тільки при оплаті. Частина застосунків не пропонує інтеграцію з пристроями чи сервісами, це обмежує користувача у моніторингу прогресу.

Згадуючи всі негативні та позитивні сторони кожного з аналогів, було прийнято рішення розробити власну вебплатформу, яка має забезпечити безкоштовний доступ до всього функціоналу, повинна мати зручний та зрозумілий інтерфейс, підтримувати інтеграцію з іншими пристроями/сервісами, мати можливість створювати для користувача персоналізовані плани для тренувань.

1.3 Вибір архітектури застосунку

Під час розробки вебплатформи для фітнес-тренувань було розглянуто кілька варіантів архітектури, щоб забезпечити стабільну роботу, зручну взаємодію між клієнтом і сервером та реалізацію всіх необхідних функцій – від генерації персональних тренувальних планів до синхронізації з зовнішніми сервісами, як-от Strava. У підсумку було обрано на клієнт-серверну архітектуру з використанням Next.js для фронтенду та NestJS для бекенду. Такий підхід дозволив досягти хорошого балансу між продуктивністю, безпекою та можливістю масштабування. Також було враховано важливість швидкого завантаження сторінок, зручного UX та підтримки інтеграцій із популярними сервісами.

Користувацький інтерфейс реалізовано на основі React із використанням хуків useState та useEffect, що дозволяє працювати з динамічними даними – наприклад, оновлювати статистику вправ чи прогрес тренувань у реальному часі. Компонентний підхід дає змогу легко повторно використовувати готові частини інтерфейсу, як-от картки вправ або поля введення. Для побудови візуальної частини використовуються Tailwind CSS і Shadcn/UI – це дозволило швидко створити адаптивний та приємний інтерфейс, який добре виглядає як на смартфоні, так і на великому екрані. Також є наявність анімацій для покращення

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	14
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

взаємодії з користувачем та інтуїтивна панель керування, яка адаптується під розміри екрана.

На серверній стороні використано NestJS – фреймворк, що побудований на принципах модульності. Кожен модуль у системі виконує свою роль: окремо обробляється автентифікація, керування тренувальними планами, збереження активностей тощо. Це дозволяє легко підтримувати й розширювати проєкт у майбутньому. Бізнес-логіка реалізована в сервісах, де відбувається обробка запитів, генерація планів або взаємодія з базою даних. Контролери, у свою чергу, відповідають за обробку запитів з клієнта, як-от створення нового плану або додавання вправи.

Уся інформація зберігається в базі даних PostgreSQL – вона ідеально підходить для зберігання структурованих, пов’язаних між собою даних (наприклад, коли одна вправа може входити до кількох планів). Для роботи з БД використано Prisma ORM, яка значно полегшує написання запитів, створення схем та міграцій. Це дозволяє зосередитися на логіці додатку, а не на написанні складного SQL-коду.

Також у систему інтегровані зовнішні сервіси – Strava API використовується для автоматичного підвантаження активностей із фітнес-браслетів, а Google OAuth 2.0 дозволяє користувачам авторизуватися через свій Google-акаунт. Щоб забезпечити високу продуктивність, особливо при великій кількості користувачів, API-запити виконуються асинхронно. При цьому всі чутливі дані – API-ключі, токени, паролі – зберігаються в захищених змінних середовища.

У результаті вдалося побудувати гнучку, масштабовану й зручну у використанні архітектуру, яка легко адаптується до нових функцій і навантажень. Це дозволяє у майбутньому безболісно додавати нові можливості – від розширеної аналітики до підтримки інших пристроїв чи фітнес-сервісів.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	15
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

При створенні даної платформи було вирішено використовувати NextJs для клієнтської частини. Основна перевага даної фреймворку є гнучка система маршрутизації, яка дозволяє легко створювати як динамічні так і статичні сторінки без створення якихось складних конфігурацій. ВЗавдяки підтримці серверного рендерингу (SSR) і статичної генерації (SSG), NextJs забезпечує швидке завантаження сторінок, що є важливо для зручності користувачів під час перегляду вправ та планів. Цей вибір також обґрунтований інтеграцією з React, що дозволяє використувати компонентний підхід і оптимізувати продуктивність та підтримку автоматичного кешування, що значно підвищує швидкість роботи платформи на різних пристроях. Отже, NextJs ідеально вписується у нашу платформу для створення адаптивного і швидкого фронтенду.

Як основну мову програмування було обрано TypeScript. Статична типізація знижує рівень помилок, покращує підтримку типів та дозволяє виявляти помилки до запуску програми, що є важливим при роботі з великими обсягами даних. TypeScript підвищує якість коду та спрощує подальшу його підтримку завдяки чіткому визначенні інтерфейсів та типів даних.

На стороні сервера використовуємо NestJs. Це фреймворк на основі NodeJs, який дозволяє будувати структуровану бекенд-систему з чітким розподілом на модулі, сервіси та контролери. Основа переважна NestJs - це його масштабованість, що дає змогу легко розширювати додаток новими функціями та модулями. Підтримка TypeScript у NestJs забезпечує типізацію, дозволяючи виявити помилки на ранніх етапах і прискорювати процес тестування. Даний фреймворк підтримує асинхронну обробку запитів, що оптимізує продуктивність при великій кількості користувачів, це і робить цей фреймворк ідеальною частиною для нашої платформи з інтенсивним обміном даних.

Для бази даних обрано PostgreSQL. Це реляційна СУБД, яка підходить для проектів у яких є велика кількість таблиць та складних зв'язків між ними. База даних забезпечує високу продуктивність і надійність при зберіганні великих

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	16
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

обсягів даних, що є важливим для функціонування платформи, де обробляються дані про деталі активностей та персональні плани. Для зручності використання PostgreSQL використовується Prisma. Це ORM система, котра забезпечує типізовану структуру даних, спрощує створення схем і міграцій, та оптимізує запити, тим самим полегшуючи роботу та підвищує ефективність в управлінні даних.

Для стилізації було використано Tailwind CSS, він обраний через його зручність та швидкість у створенні адаптивних і гнучких дизайнів. Даний інструмент дозволяє швидко налаштувати стилі, без необхідності писати власні, складні CSS структури, що економить час. До Tailwind CSS було додано Shadcn ui. Це бібліотека компонентів, яка надає готові стилізовані компоненти, починаючи від кнопки, закінчуючи формами та модальними вікнами, очевидно, що це значно полегшує та прискорює розробку інтерфейсу та гарантує приємний інтерфейс для користувача. Така комбінація підходить вимогам та забезпечує адаптивність інтерфейсу на різних пристроях.

Технічні вимоги до обладнання включають мінімальні вимоги до стабільної роботи платформи. Підтримуються операційні системи Windows 10 та новіші, macOS 11 або вище, та останні версії Linux (Ubuntu та Fedora). Для оперативної пам'яті достатньо 4 ГБ для зручного використання, але для плавнішої роботи рекомендується використання 8 ГБ оперативної пам'яті і більше. Процесор має бути не нижче Intel Core i3 або еквівалент йому AMD Ryzen 3, для забезпечення достатньої продуктивності під час роботи платформи. Для перегляду потрібно використати браузер останніх версій, наприклад найпопулярніші це Chrome або Firefox. Також потрібно пам'ятати про екран. Бажано мати роздільну здатність екрана не нижче 1366 x 768 для зручного відображення всіх елементів платформи.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	17
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновок до першого розділу

У першому розділі випускної кваліфікаційної роботи було проведено аналіз аналогів, що дозволило визначитися із подальшими кроками щодо функціоналу та стеком для розробки вебплатформи.

Технологічний стек проєкту сформовано на основі сучасних інструментів, які найкраще підходять для розробки вебзастосунку у сфері фітнесу. Для клієнтської частини обрано Next.js – фреймворк, що поєднує переваги серверного рендерингу, що дозволяє досягти продуктивності та швидкого завантаження сторінок. TailwindCSS у поєднанні з UI-бібліотекою ShadCN забезпечують створення адаптивного й привабливого інтерфейсу з мінімальними витратами часу на стилізацію. TypeScript гарантує типобезпеку та знижує кількість помилок у процесі розробки. На сервері використано NestJS – потужний backend-фреймворк, що базується на архітектурі Node.js, який дозволяє будувати масштабовану та підтримувану систему. У якості бази даних використано PostgreSQL, а для зручної роботи з нею – ORM Prisma, що значно спрощує доступ до даних та прискорює розробку. Такий стек технологій дає змогу створити сучасний, зручний у підтримці застосунок, готовий до масштабування та подальшого розвитку.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

РОЗДІЛ 2. ПРОЕКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1. Визначення варіантів використання і структури застосунку

Розроблена система являє собою платформу для тренувань, де є дві ролі: звичайного користувача та адміністратора. Для кращого розуміння взаємодії ролей та функціоналу, було створено діаграму варіантів використання на рисунку 2.1.

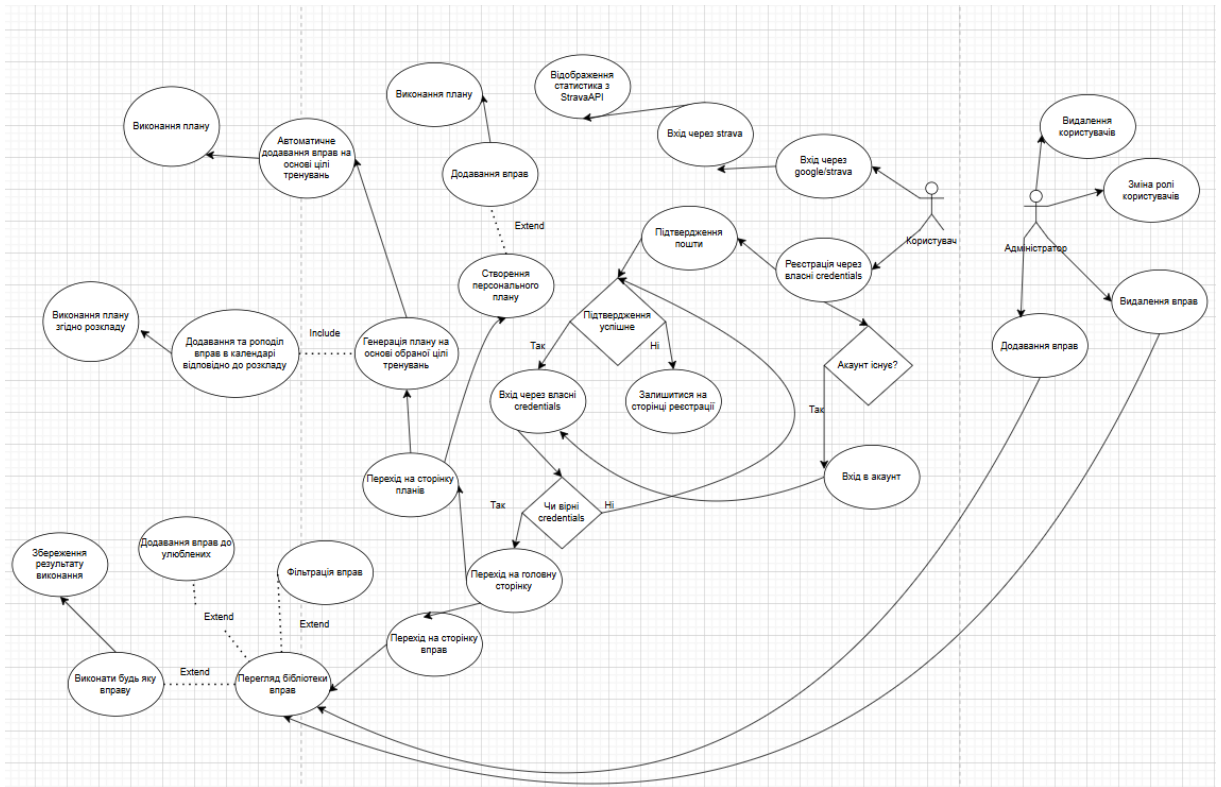


Рисунок 2.1 - Діаграма варіантів використання

Діаграма варіантів використання включає два основні актори:

Користувач взаємодіє із системою, щоб створити чи згенерувати план тренувань, отримати розклад на найближчі дні з розподілом вправ, а також отримати дані з фітнес-браслетів та побачити свою активність.

Адміністратор керує всією бібліотекою вправ та всіма користувачами. Відповідає за оновлення даних щодо вправ та ролями користувачів.

Функціональні вимоги звичайних користувачів:

У користувача є три способи реєстрації на платформі. Перший – це за допомогою електронної пошти, імені та паролю. Другий – це використовуючи

strava. Третій через Google. За замовчуванням користувач присвоюється роль звичайного користувача.

Після входу у систему, користувач має доступ до бібліотеки вправ, сторінки планів, активності, календарю та свого профілю.

На сторінці бібліотеки вправ, користувач може ознайомитися з всіма доступними вправами. Наявне поле пошуку, яке здійснює пошук за назвою вправи, фільтрація відбувається за категорією вправ. Для зручності на сторінці наявна пагінація

Сторінка планів включає відображення власних створених та згенерованих планів. Користувач має можливість видалити та додати план до улюблених, на сторінці наявна пагінація, що робить вигляд та функціонал сторінки ще кращим. При генеруванні плану, до календаря додаються три вправи на кожен день, а сама кількість днів залежить від обраного терміну виконання плану, це вказується в умовах при генеруванні плану. Тому, користувач має змогу безпосередньо виконувати вправи самостійно, незважаючи на розклад або дотримуватися розкладу.

Застосунок дозволяє користувачеві зберегти продукт до улюблених, що дозволяє швидко перевірити дані про продукт, які їх чимось зацікавили. Це дозволяє швидко повертатися до раніше досліджених продуктів та відстежувати власні харчові уподобання з часом.

Якщо користувач вже має обліковий запис в Strava та вирішив зареєструватися на платформу використовуючи саме Strava, то тоді він зможе побачити на сторінці власної активності детальний аналіз активностей. Було вирішено, що найкращими параметрами в аналізі активності буде загальна пройдена дистанція, загальна кількість спалених калорій та скільки сумарно витрачено на всі активності. Для зручності, було сформовано картки з активностями на яких видно чіткі дані щодо кожної активності.

Вимоги адміністраторів:

Вимога адміністратора дуже чіткі. Він має доступ до всіх користувачів та всієї бібліотеки вправ. Надана можливість редагування та видалення будь якої

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	20
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

вправи. Щодо користувачів, то адміністратор може змінювати ролі для користувачів та взагалі їх видаляти. Саме ці функції допомагають підтримувати платформу у зручному та безпечному вигляді для всіх, а можливість додавання, видалення, редагування вправ є дуже швидким рішенням для заповнення, або видалення вправ з бібліотеки, адже це допомагає підтримувати бібліотеку актуальною.

Бізнес-вимоги:

Метою створення вебплатформи для фітнес-тренувань було допомогти людям досягти своїх фітнес-цілей використовуючи підхід з індивідуальними планами. Завдяки інтеграції з Strava, користувачі зможуть легко після активностей переглянути свою статистику та аналіз щодо кожної вправи та згенерувати або самостійно створити персональний план на наступні дні.

Наявність бібліотеки вправ робить платформу по-справжньому унікальною. Завдяки керуванню вправ адміністратором, дана бібліотка завжди залишається актуальною та йде в ногу з потребами та трендами. Також це робить кожен особистий план користувача унікальним через чималий вибір вправ.

Функціональні вимоги:

Аутентифікація в системі реалізована через окремий модуль – AuthModule. Користувачі можуть входити в систему використовуючи Google, Strava або електронну пошту та пароль. Безпеку забезпечують JwtAuthGuard і AdminGuard, а система автоматично призначає роль для нового користувача – USER, лише адміністратор зможе змінити статус ролі.

Уся інформація зберігається в PostgreSQL, для доступу до бази даних використовується Prisma ORM. Тут зберігаються всі дані, починаючи від користувача закінчуючи інформацією про улюблені плани.

Персоналізовані плани тренувань формуються за допомогою PlanExerciseService. Цей сервіс генерує тренування, орієнтуючись на індивідуальні цілі користувача – схуднення, набір м'язової маси або підтримку активного способу життя.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	21
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Інтеграція зі Strava API здійснюється через ActivitiesModule. Завдяки цьому модулю активності синхронізуються автоматично.

Особисті налаштування користувача, зокрема улюблені плани, зберігаються та синхронізуються через FavoritesModule.

Нефункціональні вимоги:

Платформа має працювати швидко та без затримок. Наприклад, завантаження планів або синхронізація з Strava не повинні тривати довше трьох секунд. Початковий екран має відкриватися максимум за п'ять секунд, а відповіді від Strava API – приходити приблизно за дві. Система повинна витримувати навантаження щонайменше 500 користувачів одночасно – і при цьому залишатися стабільною та швидкою.

Щодо надійності, то тут планка теж висока: доступність сервісу – не менше 99,5% часу. Іншими словами, платформа повинна працювати практично без збоїв протягом усього місяця. Якщо ж раптом щось піде не так, система повинна самостійно відновитися максимум за п'ять хвилин.

Інтерфейс має бути простим і зрозумілим, як і звичайний користувач, так і адміністратор повинні розібратися з основними функціями не довше ніж за 15 хвилин. Дизайн адаптується під різні пристрої.

Щодо безпеки – персональні дані (такі як email та пароль) надійно захищені. Паролі шифруються та при використанні oauth пошта не зберігається в таблиці бази даних.

Архітектура платформи дозволяє легко масштабуватися: можна збільшувати потужності при зростанні кількості користувачів або додавати нові функції (наприклад, нові вправи) без впливу на вже наявний функціонал.

Платформа працює з усіма сучасними браузерами – Chrome, Firefox, Safari (версії з 2023 року й новіші). На стороні користувача для стабільної роботи достатньо мати хоча б 2 ГБ оперативної пам'яті.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	22
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Архітектура системи:

У основі цього рішення лежить клієнт-серверна архітектура, побудована на таких вебтехнологіях – серверна частина реалізована на NestJS, а клієнтська – на NextJS. Архітектура включає кілька окремих модулів, що дозволяє зробити систему гнучкою та зручною для масштабування.

Клієнтська частина – створений на React із використанням Tailwind CSS для адаптивного та зручного інтерфейсу, для стилізації використано shadcnui. Завдяки цьому додаток добре працює у різних браузерях і на різних пристроях.

Серверна частина має модульну структуру, де виділені такі частини, як AuthModule для авторизації, PlanModule для роботи з планами, ExerciseModule для вправ, ActivitiesModule та AdminModule. Для автентифікації використовується JWT, а також інтеграція зі сторонніми сервісами, наприклад Google та Strava. Вся інфраструктура налаштована так, щоб не потребувати складного ручного розгортання.

Для розширення можливостей система інтегрується із зовнішніми API. Зокрема, Strava API дозволяє отримувати дані про активності користувачів – відстань, час тренувань, калорії. А через CloudinaryModule реалізована зручна робота з медіа: завантаження та обробка зображень у профілі користувача.

Клієнтська архітектура.

Клієнтська частина вебплатформи реалізована з урахуванням принципів модульної архітектури та чіткого розділення відповідальностей. За відображення інтерфейсу відповідає React. Для стилізації використовується Tailwind CSS та shadcnui, що забезпечує адаптивність, а використовуючи фреймворк NextJs можна зручна та швидко організувати навігацію між основними розділами.

Бізнес-логіка винесена в окремо, де застосовуються кастомні React-хуки та Context API для керування глобальним станом.

Обмін даними із сервером організований у вигляді асинхронних запитів до API, побудованого на NestJS (через axios та fetch). Крім того, реалізована інтеграція з Strava API для автоматичної синхронізації активностей. Такий підхід

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	23
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

дозволяє легко масштабувати додаток, ефективно проводити тестування та розробляти окремі модулі.

2.2 Проектування бази даних системи

Для зберігання інформації в застосунку було обрано PostgreSQL, використовуючи Prisma ORM. PostgreSQL є реляційною SQL базою даних, що працює з таблицями.

База даних складається з 10 таблиць та 4 списків:

Таблиця 2.1.

Призначення колекцій бази даних

Назва колекції	Призначення
User	Таблиця для зберігання інформації про користувачів
Role	Визначення ролей для користувачів
GoalType	Цілі для тренування
WorkoutDay	таблиця для днів у планах тренувань
WorkoutPlanExercise	таблиця для зв'язку вправ із планами та днями
WorkoutPlan	таблиця для планів тренувань
ExerciseType	Тип вправи
PlanStatus	Статус планів
Activity	Таблиця для активностей, синхронізованих зі Strava
ExerciseCategory	Таблиця для категорій вправ
TrackedExercise	Таблиця для відстежуваних вправ
Exercise	Таблиця для вправ
WorkoutPlanCompletion	Таблиця для відстеження завершення планів
FavoritePlan	Таблиця для улюблених планів

Структуру бази даних представлено на рисунку 2.2, де відображено 10 таблиць із зв'язками. Вся система вдало продумана та оптимізована для безпечного зберігання всіх можливих даних.

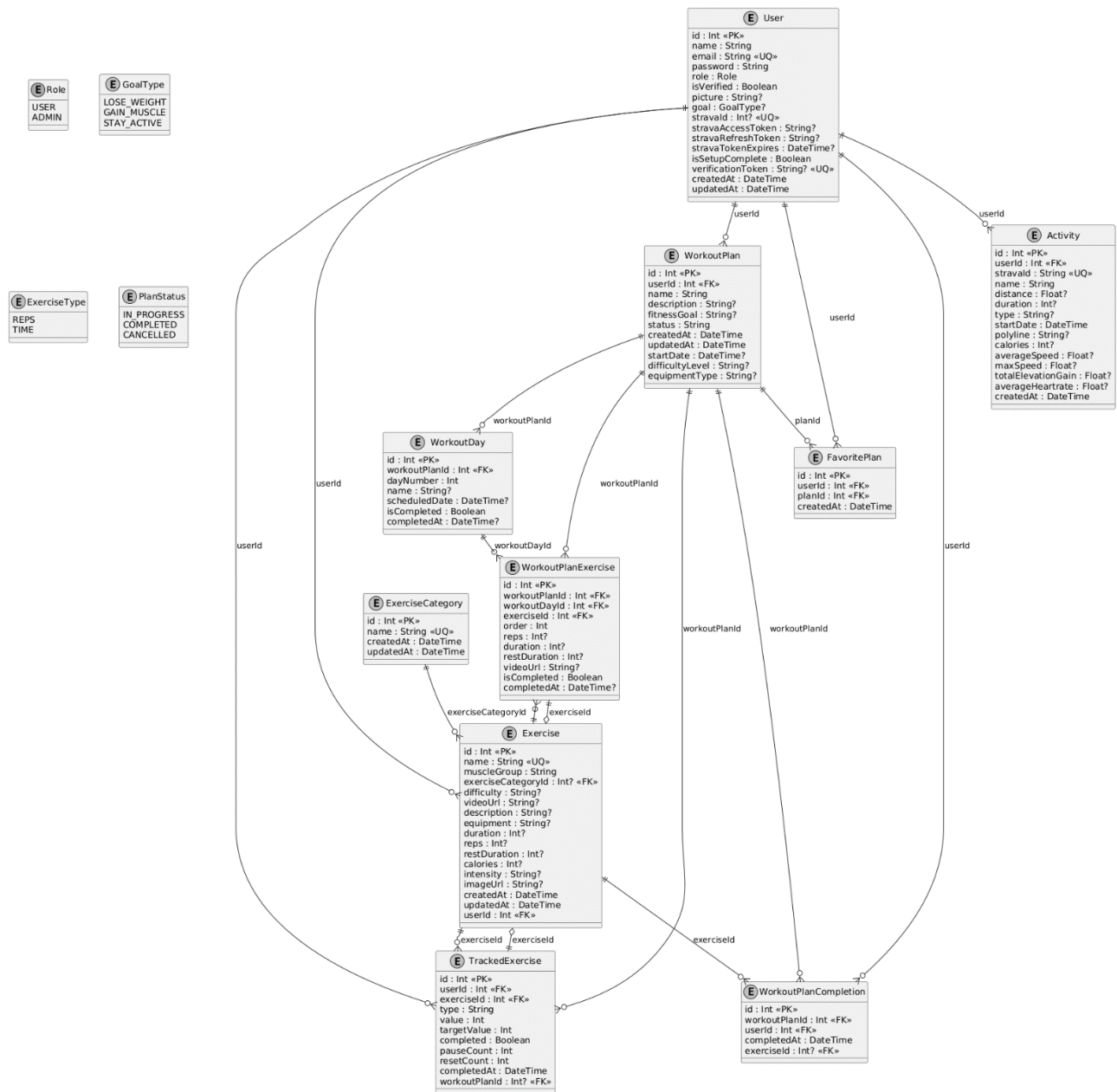


Рис. 2.2 - Діаграма бази даних

Таблиця 2.2

Таблиця Users

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
name	string	Ім'я користувача
email	string	Електронна пошта
password	string	Пароль користувача
role	enum	Роль користувача
picture	string	Зображення користувача
goal	enum	Мета тренувань
stravaId	int	Унікальний ідентифікатор Strava
stravaAccessToken	string	Access Token Strava

Продовження таблиці 2.2

Поле	Тип	Призначення
stravaRefreshToken	string	Refresh Token Strava
stravaTokenExpires	DateTime	Час закінчення токена
createdAt	DateTime	Дата створення акаунту
updatedAt	DateTime	Дата оновлення акаунту

Дана таблиця зберігає всю основну інформацію про кожного користувача платформи. Тут є все: ім'я, email, пароль, роль, фото профілю, ціль тренувань, а також дані для інтеграції зі Strava – ID, токени та їхній термін дії. Крім того, зберігаються дати створення й останнього оновлення профілю.

Таблиця 2.3

Таблиця WorkoutPlan

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор плану
userId	int	Ідентифікатор користувача
name	string	Назва плану
description	string	Опис плану
fitnessGoal	string	Фітнес мета
status	string	Статус виконання
createdAt	DateTime	Дата створення
updatedAt	DateTime	Дата оновлення
startDate	DateTime	Запланований початок
difficultyLevel	string	Рівень складності
equipmentType	string	Тип обладнання

Таблиця WorkoutPlan відповідає за саму структуру тренувального процесу. У ній зберігається, хто створив план, яка в нього назва, опис, яка мета, коли він починається, на якому він етапі, яка складність і чи потрібне якесь обладнання.

Таблиця 2.4

Таблиця WorkoutDay

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
workoutPlanId	int	ID плану
dayNumber	int	Номер дня
name	string	Назва дня
scheduleDate	DateTime	Запланована дата
isCompleted	boolean	Статус виконання
completedAt	DateTime	Дата завершення

Тут зберігається розподіл тренувань по днях. Для кожного дня вказано, до якого плану він належить, який у нього номер у черзі, чи є у нього назва, коли планується виконання і чи завершене воно насправді.

Таблиця 2.5

Таблиця WorkoutPlanExercise

Поле	Тип	Призначення
id	int	Ідентифікатор вправи у плані
workoutPlanId	int	ID плану
workoutDayId	int	ID дня
exerciseId	int	ID вправи
order	int	Порядок виконання
reps	int	Кількість повторень
duration	int	Тривалість вправи
restDuration	int	Тривалість відпочинку
videoUrl	string	Посилання на відео
isCompleted	boolean	Статус виконання
completedAt	DateTime	Дата завершення

Це таблиця, яка об'єднує конкретні вправи з певним планом і днем. Тут вказано, в якій послідовності виконуються вправи, скільки повторень потрібно зробити, скільки часу триває вправа, скільки слід відпочивати і навіть – чи є відеоінструкція. Також відображено, чи завершена вправа та коли саме.

Таблиця 2.6

Таблиця Activity

Поле	Тип	Призначення
id	int	Ідентифікатор активності
userId	int	ID користувача
stravaId	string	Ідентифікатор Strava
name	string	Назва активності
distance	float	Дистанція
duration	int	Тривалість
type	string	Тип активності
startDate	DateTime	Дата початку
Polyline	string	Карта
Calories	int	Калорії
averageSpeed	float	Середня швидкість
maxSpeed	float	Максимальна швидкість
totalElevationGain	float	Загальний підйом
averageHeartRate	float	Середнє серцебиття
createdAt	DateTime	Дата створення

Activity зберігає інформацію про активності, які синхронізуються зі Strava. Тут можна побачити назву активності, тривалість, дистанцію, тип, дату,

маршрут, кількість витрачених калорій, пульс, швидкість тощо. Дуже зручно для аналітики та трекінгу результатів.

Таблиця 2.7

Таблиця ExerciseCategory

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
name	string	Назва категорії
exercises	Exercises[]	Масив вправ
createdAt	DateTime	Створення
updatedAt	DateTime	Оновлення

Завдяки цій таблиці вправи діляться на категорії – наприклад, для ніг, спини чи рук. Це значно спрощує пошук потрібної вправи або складання власного плану.

Таблиця 2.8

Таблиця TrackedExercise

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
userId	string	Унікальний ідентифікатор користувача
User	User	Зв'язок з таблицею User
exerciseId	Int	ID вправа
exercise	Exercise	Зв'язок з таблицею Exercise
Type	string	Тип відстежувального показника
value	String	Значення поточного показника
targetValue	Int	Значення для досягнення користувачем
completed	Int	Статус завершення вправи
pauseCount	Boolean	Кількість пауз
resetCount	Int	Кількість скидань
completedAt	Int	Статус завершення
WorkoutPlan	DateTime	Зв'язок з таблицею WorkoutPlan
workoutPlanId	WorkoutPlan	Ідентифікатор плану тренувань

Це своєрідний щоденник прогресу: яка вправа, який результат зараз, яка ціль, скільки було пауз чи скидань, чи досягнуто мети. Також вказано, чи пов'язана ця вправа з конкретним тренувальним планом.

Таблиця 2.9

Таблиця Exercise

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
name	string	Назва вправи
muscleGroup	string	Тип м'язів

Продовження таблиці 2.9

Поле	Тип	Призначення
exerciseCategory	ExerciseCategory	Категорія вправи
exerciseCategoryId	int	ID категорії
Difficulty	string	Показник складності
videoUrl	String	Відео для виконання
description	string	Опис вправи
Equipment	string	Тип інструментів для виконання
Duration	int	Тривалість виконання
Reps	Int	Кількість повторень
restDuration	Int	Кількість відпочинку
calories	Int	Кількість калорій
intensity	String	Інтенсивність
imageUrl	String	Зображення вправи
createdAt	DateTime	Дата створення
updatedAt	DateTime	Дата оновлення
workoutPlans	WorkoutPlanExercise[]	Зв'язок з WorkoutPlanExercise
completions	WorkoutPlanCompletion[]	Зв'язок з WorkoutPlanCompletion
TrackedExercise	TrackedExercise[]	Зв'язок з TrackedExercise
userId	Int	ID користувача
user	Exercise	Зв'язок з User

Одна з головних таблиць: тут зберігаються всі вправи, доступні на платформі. Назва, цільова м'язова група, категорія, опис, рівень складності, відео, обладнання, тривалість, калорії, інтенсивність – усе в одному місці.

Таблиця 2.10

Таблиця WorkoutPlanCompletion

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
workoutPlanId	int	Унікальний ідентифікатор плану
workoutPlan	WorkoutPlan	Зв'язок з WorkoutPlan
userId	Int	ID користувача
user	User	Зв'язок з користувачем
completedAt	DateTime	Завершення вправи
Exercise	Exercise	Зв'язок з Exercise
exerciseId	Int	Унікальний ідентифікатор вправи

Фіксує момент, коли користувач офіційно завершує тренувальний план. У ній зберігається, який саме план завершено, ким і коли. За потреби – ще й яка саме вправа завершила весь процес.

Таблиця 2.11

Таблиця FavoritePlan

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
userId	int	ID користувача

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	29
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Поле	Тип	Призначення
planId	Int	ID плану
createdAt	DateTime	Дата додавання до улюблених
User	User	Зв'язок з User
plan	WorkoutPlan	Зв'язок з WorkoutPlan

У цій таблиці фіксуються всі плани, які користувач позначив як улюблені. Тут видно, хто додав який план до обраного і коли саме це відбулося.

2.3. Проектування та реалізація алгоритмів роботи системи

В даному розділі слід розглянути UML діаграми, дані діаграми є ключовим аспектом у розробці програмного забезпечення. Використовуючи UML діаграми, ми можемо описувати конкретні варіанти, можливості роботи платформи та виявити недоліки чи можливості щодо покращення роботи.

На рисунку 2.3 зображена діаграма активності щодо додавання вправ до власноруч створеного плану та взаємодії адміністратора та користувача у даній функції.

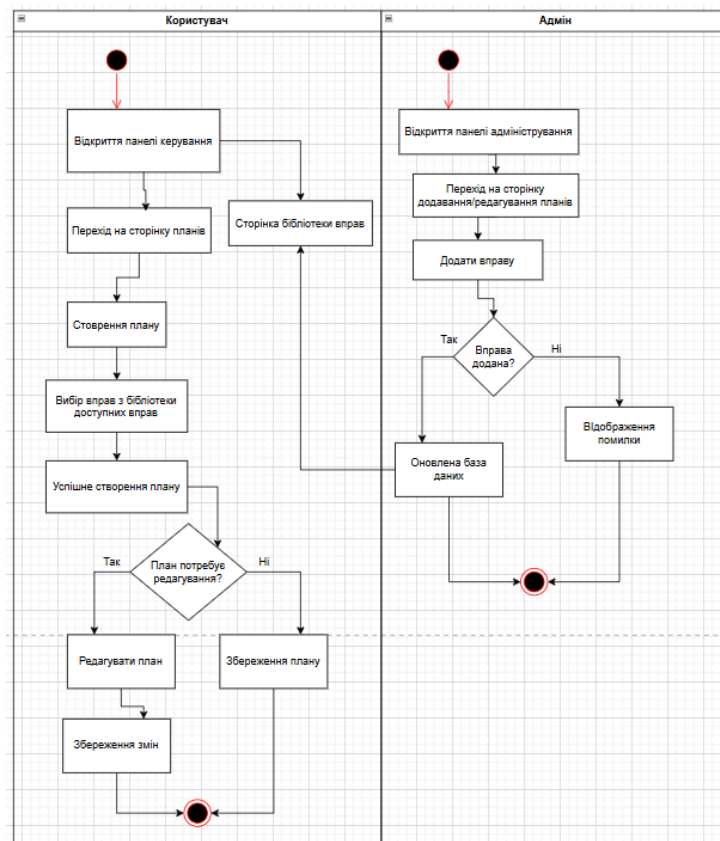


Рис. 2.3 - Діаграма активності додавання вправ до плану

Дана діаграма активності показує, як у системі взаємодіють дві ролі – звичайний користувач і адміністратор – у процесах створення та адміністрування тренувальних планів і вправ.

Для користувача все починається з відкриття панелі керування. Далі він обирає, куди перейти – на сторінку з планами чи в бібліотеку вправ. Якщо вирішує створити новий план, він обирає з доступного списку вправ і на їх основі формує свій індивідуальний план тренувань. Після цього система перевіряє, чи потрібне редагування. Якщо потрібно щось виправити – користувач переходить до редагування. Якщо все готово, або редагування вже виконано, він просто зберігає план і завершує роботу.

Адміністратор починає зі своєї панелі. Його головне завдання – керувати вправами: додавати нові або редагувати наявні. Для цього він переходить на відповідну сторінку, вводить дані про нову вправу, і система перевіряє, чи все заповнено правильно. Якщо так – вправа зберігається в базі. Якщо є помилки, адміністратор отримує повідомлення і може їх виправити. Після цього і його частина роботи завершується.

У висновку, діаграма демонструє, як користувач створює власні тренувальні плани, а адміністратор – підтримує та оновлює бібліотеку вправ. Обидва процеси працюють паралельно й доповнюють одне одного, забезпечуючи повну функціональність фітнес-додатку.

Важливим етапом кожної розробки, а особливо такої платформи є логіка реєстрації/входу в систему та наявні можливості це зробити. Для кращого розуміння роботи даного модуля, було створено діаграму активності для реєстрації користувачів. На рис. 2.4 показано саме цей процес.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	31
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

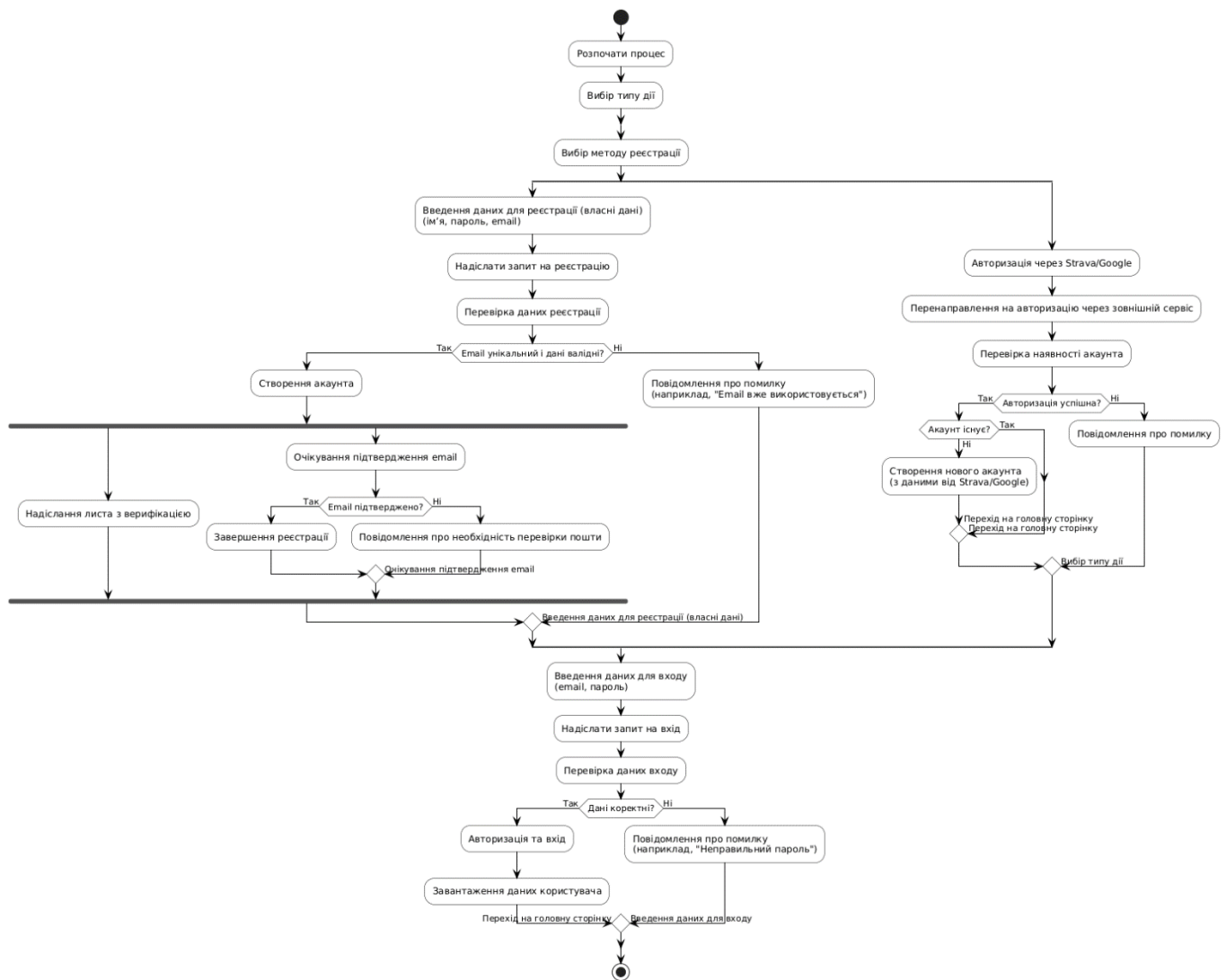


Рис. 2.4 - Діаграма активності реєстрації користувачів

Усе починається з того, що перед користувачем є вибір щодо реєстрації та входу у систему. Якщо обрано реєстрацію, користувач може або ввести свої дані вручну, або скористатися одним із сервісів – Google чи Strava.

У разі звичайної реєстрації користувач заповнює форму з ім'ям, електронною поштою та паролем. Система перевіряє, чи не зайнята вже введена пошта. Якщо така адреса вже використовується, виводиться повідомлення про помилку. Якщо все добре – створюється обліковий запис, і на вказану пошту надсилається лист із підтвердженням. У випадку не підтвердження пошти, користувач буде залишатися на сторінці реєстрації. Якщо підтвердження проходить успішно – реєстрацію завершено, і можна входити в систему. У разі, якщо підтвердження не надійшло, користувач побачить відповідне повідомлення

і матиме змогу перевірити свою скриньку ще раз. Коли пошта підтверджена користувач зможе успішно увійти у систему.

Якщо ж було обрано вхід або реєстрацію через Google чи Strava, користувач перенаправляється до відповідного сервісу для авторизації. Після цього система перевіряє, чи існує вже такий акаунт. Якщо так – відбувається вхід. Якщо ж це перше підключення, система створює новий обліковий запис на основі отриманих даних, і користувач потрапляє на головну сторінку.

Коли користувач обирає звичайний вхід, він вводить email і пароль. Після надсилання форми система перевіряє правильність даних. Якщо вони вірні – користувач авторизується, завантажуються його дані, і він переходить до головної сторінки. У разі помилки, наприклад, неправильного пароля, система повідомляє про це і дозволяє спробувати ще раз.

Загалом, діаграма демонструє всі можливі шляхи входу та реєстрації, враховуючи перевірки, повідомлення про помилки, підтвердження пошти та інтеграцію з зовнішніми сервісами.

2.4 Реалізація функціоналу додатку

Система містить великий спектр модулів: аутентифікацію користувачів, робота з активностями, логіка щодо роботи з вправами, можливості адміністратора щодо управління користувачами та вправами, робота з планами та користувачами, збереження зображень. Кожен модуль розроблено з урахуванням модульної архітектури NestJs, що полегшує підтримку модулів та легшу імплементацію нових. Для детального огляду було обрано розглянути роботу модуля, який пов'язаний з аутентифікацією користувача, бо він надає доступ до всіх інших функцій платформи. Даний модуль є ключовим елементом безпеки системи, він забезпечує надійне користування і захист даних. Також гнучкість даного модуля дозволяє інтегрувати додаткові методи автентифікації в подальшому.

Архітектура програмного забезпечення зображена на рис.2.5.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	33
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

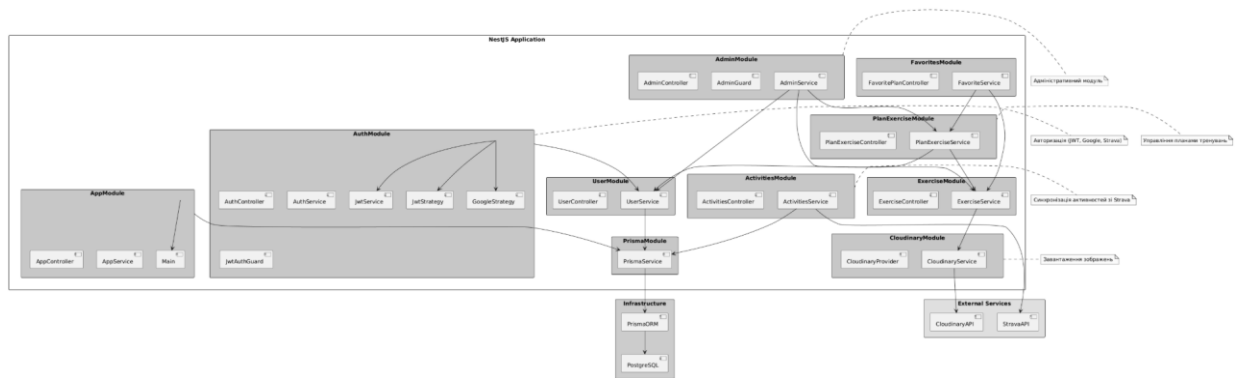


Рис. 2.5 - Діаграма компонентів системи

У процесі реєстрації нового користувача спочатку перевіряється, чи вже існує користувач із таким email. Якщо такої пошти не існує – створюється новий запис, пароль хешується, а також формується унікальний токен та підтвердження email. Після цього надсилається лист із кнопкою підтвердження, для цього було використано сервіс nodemailer. Усе це реалізовано в методі register() сервісу аутентифікації. Також варто чітко зрозуміти логіку входу у систему. Алгоритм реєстрації наведено в додатку А. Алгоритм логіну наведено в додатку А.

Дана функція приймає три параметри: email, password, res. У параметр res записується відповідь HTTP, у який згодом записується токени. На початку метод шукає користувача в базі даних за email, використовуючи Prisma (this.prisma.user.findUnique). Якщо користувача не знайдено, одразу викидається помилка з повідомленням "Invalid credentials" і статусом 401 (UNAUTHORIZED).

Якщо ж користувача знайдено, далі перевіряється, чи введений пароль збігається з тим, що збережений у базі. Для цього використовується бібліотека bcrypt, а саме метод compare. Якщо паролі не збігаються – знову викидається помилка зі статусом 401.

Після цього метод перевіряє, чи підтверджено електронну пошту користувача (поле isVerified). Якщо пошта ще не підтверджена, викидається помилка зі статусом 403 (FORBIDDEN) і повідомленням, що спочатку потрібно підтвердити email.

Якщо всі перевірки пройдені успішно, запускається генерація токенів використовуючи метод `generateToken(user)`, а потім ці токени записуються в куки через метод `setCookies(res, tokens)`. Це дозволяє зберігати авторизацію для наступних запитів користувача. У кінці метод повертає просте повідомлення: `{ message: 'Login successful' }`.

Додатково, згенеровані токени (`access token` і `refresh token`) мають обмежений термін дії, що підвищує безпеку системи, а `refresh token` дозволяє користувачу отримувати новий `access token` без повторного введення даних. Модуль аутентифікації також інтегрується з іншими частинами платформи, наприклад, із модулем активностей, щоб забезпечити доступ до даних лише авторизованим користувачам. Процес логіну підтримує обробку помилок у реальному часі, що покращує користувацький досвід, а токени зберігаються в куки. У майбутньому планується додати двофакторну автентифікацію для підвищення рівня безпеки. Така архітектура модуля дозволяє адаптувати його до зростаючих потреб платформи та нових вимог щодо захисту даних.

Тестування модуля проходило вручну, де перевірялося всі ключові сценарії, включаючи успішний вхід, обробку помилок і верифікацію електронної пошти. Тестування включало симуляцію при різних умовах, таких як введення некоректних даних чи перевищення терміну дії токена. Результати тестування використовувалися для вдосконалення логіки та підвищення стабільності модуля.

Висновки до розділу 2

Під час роботи над другим розділом було створено діаграму системи, а також діаграми активності, які демонструють, як влаштована платформа та взаємодію користувача та адміністратора. Було проаналізовано і спроектовано основні таблиці бази даних, які зберігають інформацію про користувачів, тренувальні плани, вправи та активності. Також, розроблено діаграму варіантів використання, яка описує взаємодію у системі між різними компонентами. Було детально проаналізовано роботу методів `register()` і `login()`. Вони реалізують

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	35
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

реєстрацію з підтвердженням email та вхід через звичайні облікові дані. Також реалізовано обробку помилок і захист від несанкціонованого доступу. Можна виділити ключові модулі – AuthModule, PlanModule, ExerciseModule, ActivitiesModule та AdminModule – які забезпечують основний функціонал: створення персональних планів тренувань, аналіз фізичної активності, керування вправами та зручну адміністративну панель. Архітектура додатку побудована таким чином, щоб забезпечити масштабованість і можливість легкого розширення функціоналу в майбутньому.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ЗАСТОСУНКОМ

3.1. Порядок встановлення та налаштування параметрів системи.

Система побудована за принципом клієнт-серверної архітектури, що дозволяє розділити обов'язки між фронтендом і бекендом. Користувачі працюють із вебзастосунком, створеним на базі Next.js – фреймворку, який забезпечує швидке завантаження сторінок і оптимізацію для пошукових систем. Це важливо для користувачів, які шукають фітнес-платформи через Google чи інші пошукові системи. Вебзастосунок надсилає запити до серверної частини, яка реалізована на NestJS – фреймворку на Node.js. NestJS використовує модульну структуру, через це розробники можуть швидко додавати новий функціонал, підключати сторонні сервіси чи масштабувати систему. Для роботи з базою даних використовується Prisma ORM, вона спрощує написання запитів і керування структурою даних.

Усі важливі дані – профілі користувачів, плани тренувань, списки вправ, записані активності – зберігаються в реляційній базі даних PostgreSQL. Цей інструмент обрано через його стабільність, гнучкість і зручність при роботі з великими обсягами структурованої інформації. Prisma дозволяє створювати моделі даних, автоматизує міграції, спрощує читання і запис до бази, а також допомагає підтримувати чисту архітектуру проєкту.

Для розширення функціональності система підключається до зовнішніх API. Завдяки використанню Strava API відбувається автоматичне отримання активностей користувача з фітнес-браслетів – платформа імпортує маршрути, калорії, час і відстань. Google OAuth 2.0 використовується для авторизації через обліковий запис Google, що значно спрощує вхід. А Cloudinary API дає змогу працювати з профільними зображеннями – завантаження, зберігання. Такий набір інтеграцій робить платформу зручною, гнучкою та технічно підготовленою до масштабування.

Розробка починається з встановлення всіх необхідних залежностей для Next.js і NestJS. Параметри підключення до бази даних та API ключі зберігаються в .env-файлі, що дозволяє налаштувати систему під конкретне середовище та

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	37
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

зберегти конфіденційність інформації. З боку клієнта обов’язково вказується адреса серверного API – через змінну середовища NEXT_PUBLIC_API_BASE_URL. Після цього можна запускати сервер `npm run start` та клієнт `npm run dev` у паралельному режимі для зручного тестування.

Щоб платформа працювала стабільно, важливо регулярно оновлювати бібліотеки через `npm` або `yarn`, слідкувати за логами серверу та аналізувати швидкість відповіді API. Це дозволяє оперативно виявляти і виправляти проблеми, наприклад, затримки при взаємодії з зовнішніми сервісами. Вся ця інфраструктура розроблена так, щоб користувач міг безперешкодно користуватись платформою, зосереджуючись на власних фітнес-цілях. При цьому система залишається відкритою до змін – за потреби можна додати персональні рекомендації, розширену статистику чи нові інтеграції з іншими фітнес-сервісами.

У подальшому планується покращити користувацький досвід, планується інтеграція з системами сповіщення, наприклад Firebase Cloud Messaging, для відправки нагадувань про тренування чи якихось оновлень щодо вправи чи загалом платформи. Це вимагатиме додаткової конфігурації серверної частини для обробки push-сповіщень і забезпечення їх доставки.

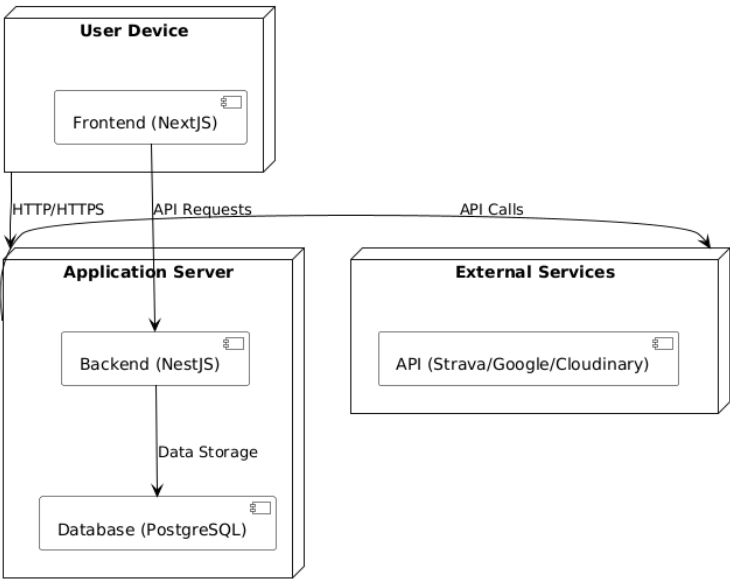


Рисунок 3.1 - Діаграма розгортання застосунку

3.2 Структура інтерфейсу вебплатформи

Коли користувач потрапляє на наш сайт, то він бачить цільову сторінку, де наявні способи входження у систему. Користувач може обрати вхід за допомогою власних даних, використати Google або Strava для швидкого входу, це підвищує рівень безпеки завдяки інтеграції з надійними зовнішніми сервісами. Дана сторінка зроблена інтуїтивно зрозуміло, акцент робиться на відображенні способів для входу рис.3.2.

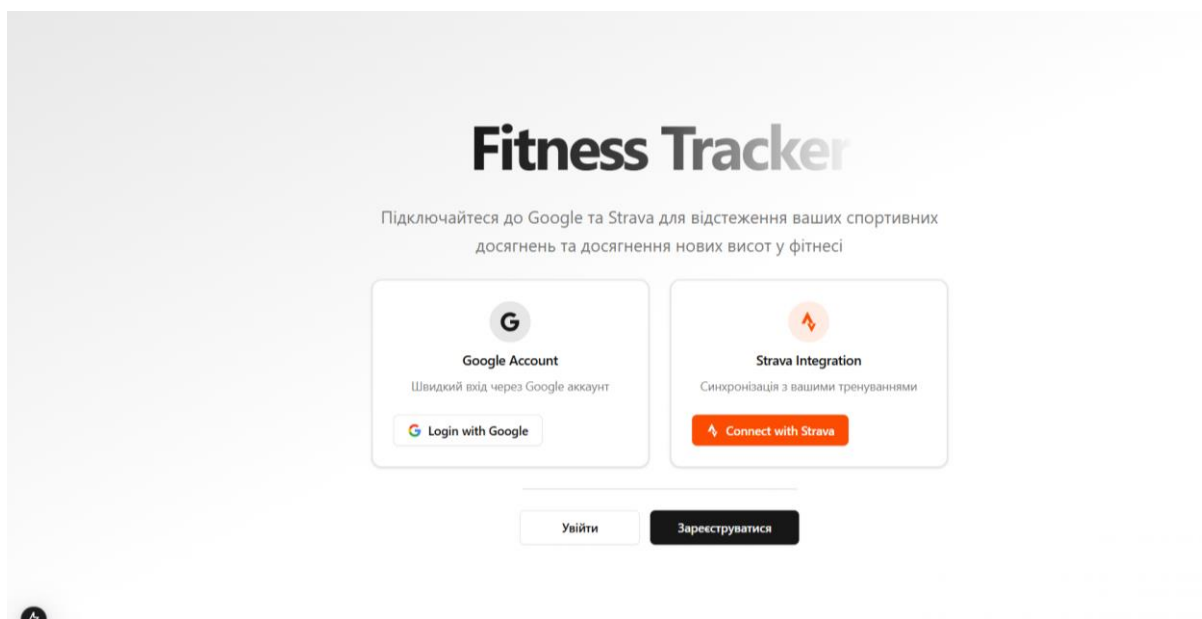


Рисунок 3.2 – Цільова сторінка застосунку

Обравши пункт реєстрація, потрапляємо на сторінку з формою для реєстрації. Форма включає поля для введення імені, електронної пошти та пароля, також вона оснащена механізмами валідації, які повідомляють користувача про помилки рис.3.3.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

Створити акаунт

Вхід через Google

Підключитися через Strava

Або продовжи з

Ім'я

Олег

Пошта

godlevskyj04@gmail.com

Пароль

.....

Створити акаунт

Вже маєте акаунт? [Увійти](#)

Рисунок 3.3 – Форма реєстрації

Після реєстрації користувачу на пошту приходить лист з підтвердженням вказаної пошти. Це робиться для того, щоб перевірити автентичність користувача та захистити платформу від спаму ботів, якщо користувач не підтвердить власну пошту, то не зможе увійти у систему. Даний процес посилює безпеку, та зумовлює користувачі використовувати дійсні електронні пошти, це для користувача дуже добре, адже це в подальшому допоможе з комунікацією, для відновлення паролю рис.3.4.

Перевірте вашу пошту

Ми надіслали лист для перевірки вашої пошти

godlevskyj04@gmail.com.

Будь ласка, перевірте свою поштову скриньку та натисніть посилання, щоб підтвердити свій обліковий запис.

Не отримав підтвердження? [Спробувати знову](#)

Рисунок 3.4 – Повідомлення про надісланий лист на пошту
Лист успішно прийшов на вказану пошту рис.3.5.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

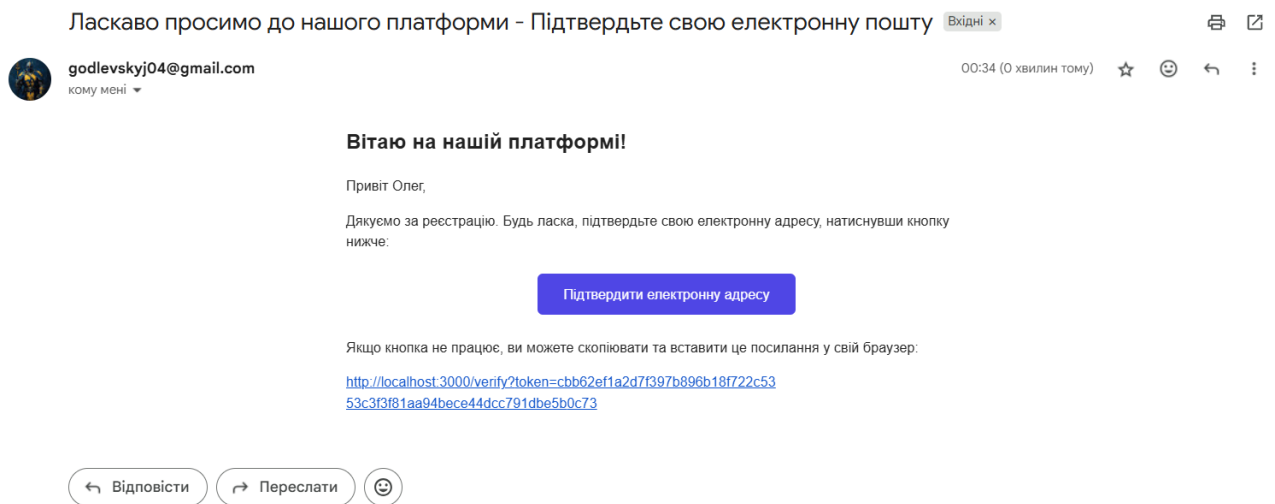


Рисунок 3.5 – Лист на пошті

Після натискання на кнопку «Підтвердити електронну адресу», користувач перенаправляє на платформу з таким повідомленням рис.3.6.

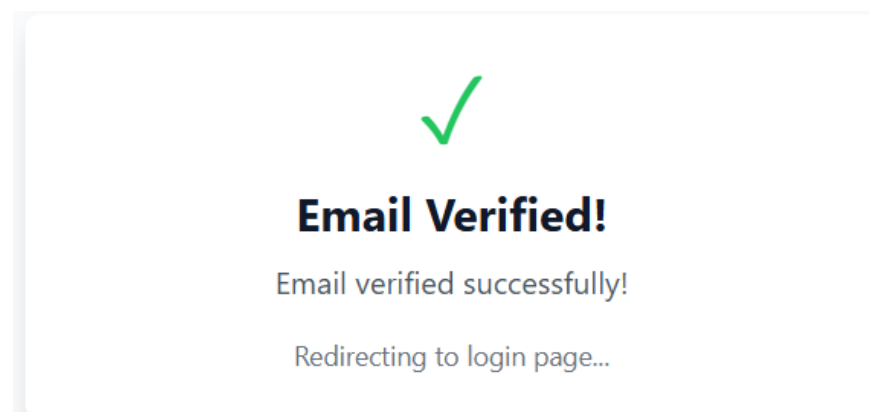


Рисунок 3.6 – Перенаправлення на сторінку входу
Далі перенаправляє на сторінку входу рис.3.7.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	41
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Реєстрація' (Don't have an account? Registration)."/>

Рисунок 3.7 – Форма для входу

Після успішного входу, користувача бачить панель керування, де і зможу здійснювати навігацію між сторінками. У панелі керування є такі розділи як: вправи, плани, активності, календар та профіль. Дизайн виконано у мінімалістичному стилі рис.3.8.

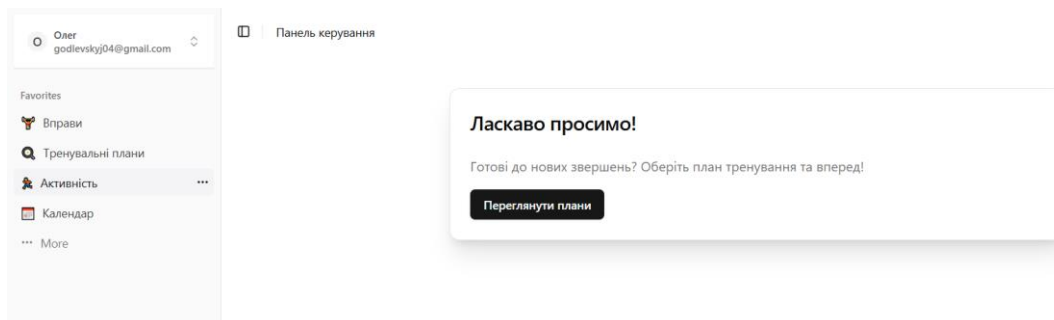


Рисунок 3.8 – Користувач увійшов у систему

Перейдемо на сторінку з вправами. Коли користувач потрапляє на дану сторінку, то одразу бачить статистику вправ розподілених по категоріях. Також наведе поле для пошуку та сама фільтрація по категоріям рис.3.9.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	42
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

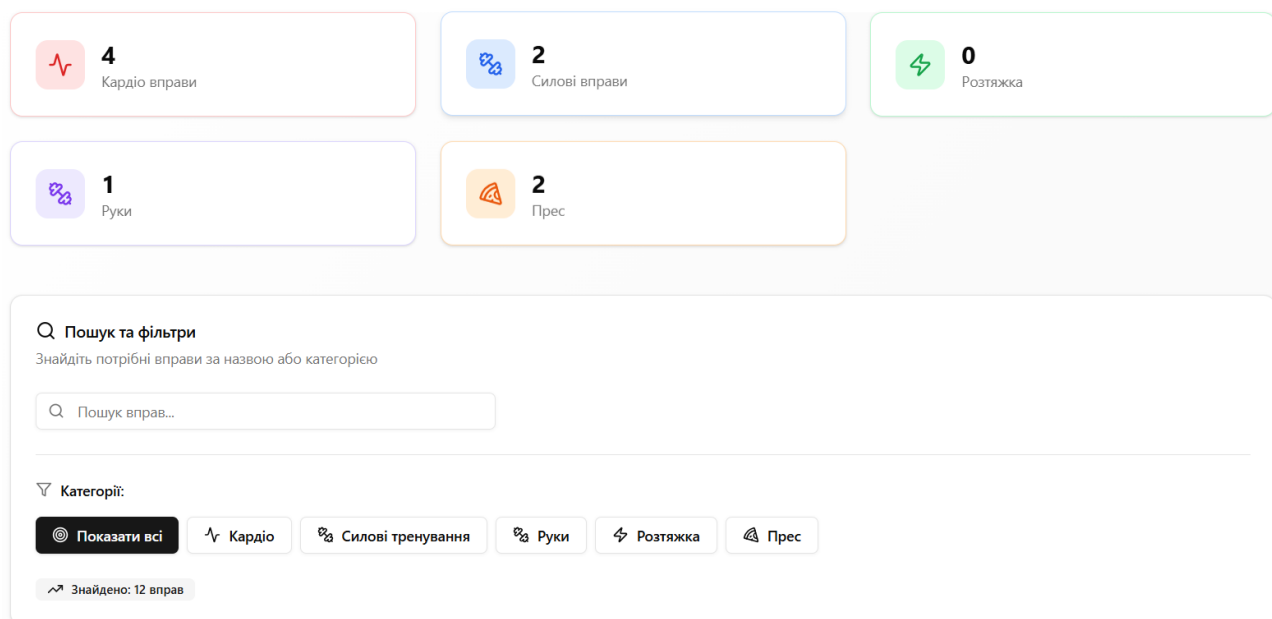


Рисунок 3.9 – Фільтрація та пошук вправ

Нижче користувач побачить картки з вправами. На картках відображено актуальну інформацію щодо кожної вправи – це сама назва, опис, позначка, яка вправа до якої категорії належить, та які м'язи задіяні в цій вправі рис.3.10.

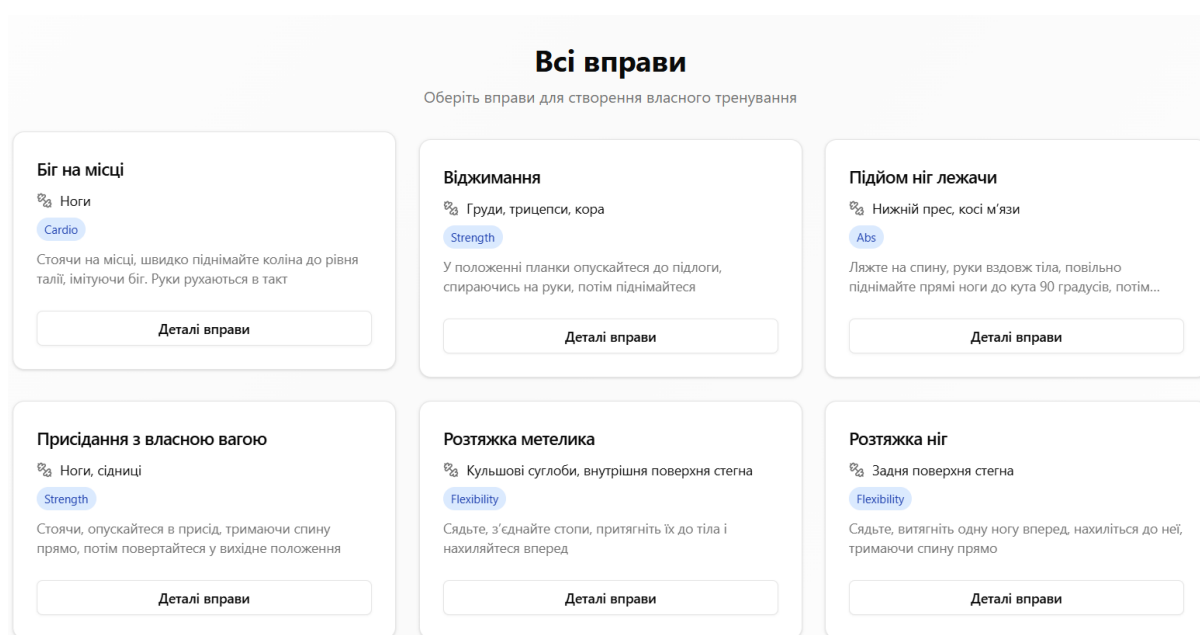


Рисунок 3.10 – Бібліотека вправ

При переході на сторінку конкретної вправи користувач бачить складність та групу м'язів, які працюють у цій вправі. Користувач може обрати тип виконання вправи, чи то буде виконання на час, чи з використанням повторів. //

На сторінці наявний відео- урок, який допомагає краще зрозуміти техніку виконання вправи рис.3.11.

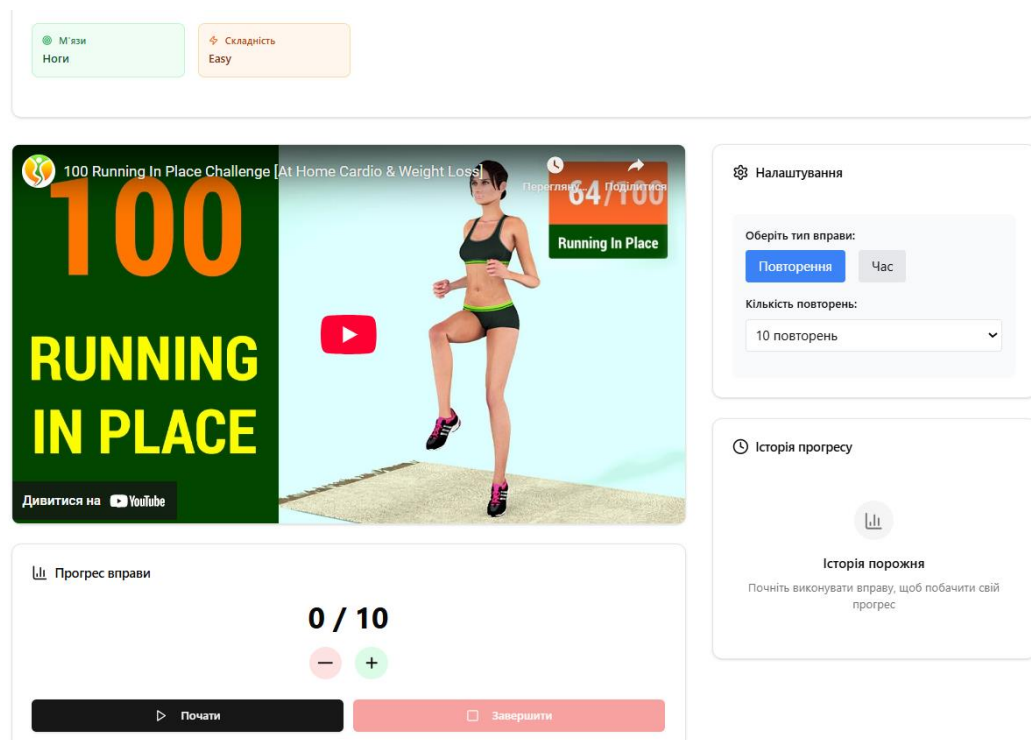


Рисунок 3.11 – Сторінка вправи

Оберемо для виконання тип «Час». Під час виконання вправи із згаданим типом, користувач має змогу поставити таймер на паузу або його взагалі скинути. Наявний прогрес-бар, який візуалізує залишок часу рис.3.12

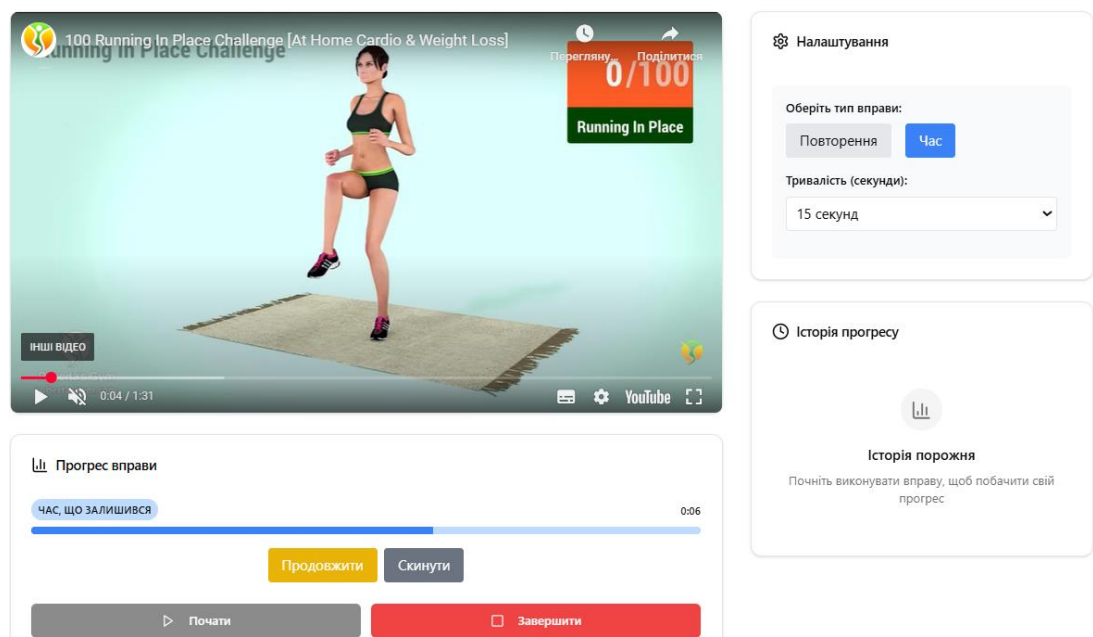


Рисунок 3.12 – Виконання вправи використовуючи тип «Час»

Після закінчення часу бачить відповідне повідомлення та запис щодо завершення вправи в історію прогресу виконання вправи рис.3.13

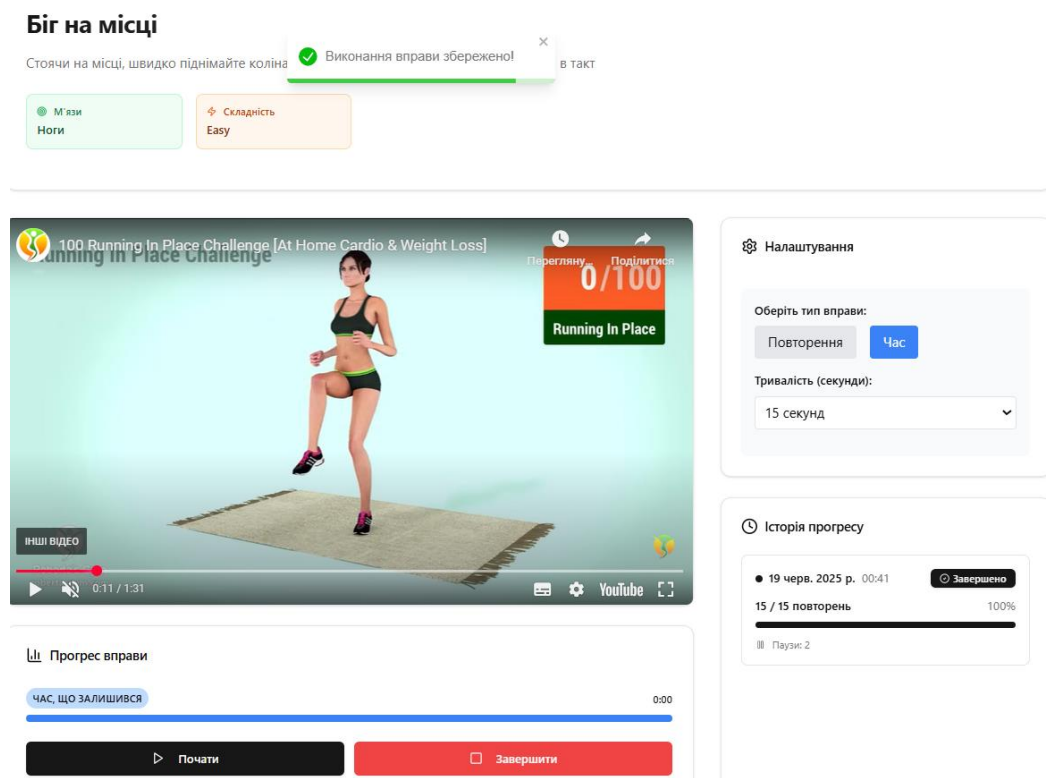


Рисунок 3.13 – Успішне завершення вправи

Ідентично фіксується виконання вправи за повторами. Для цього використовується лічильник повторів з ручним коригуванням рис.3.14.

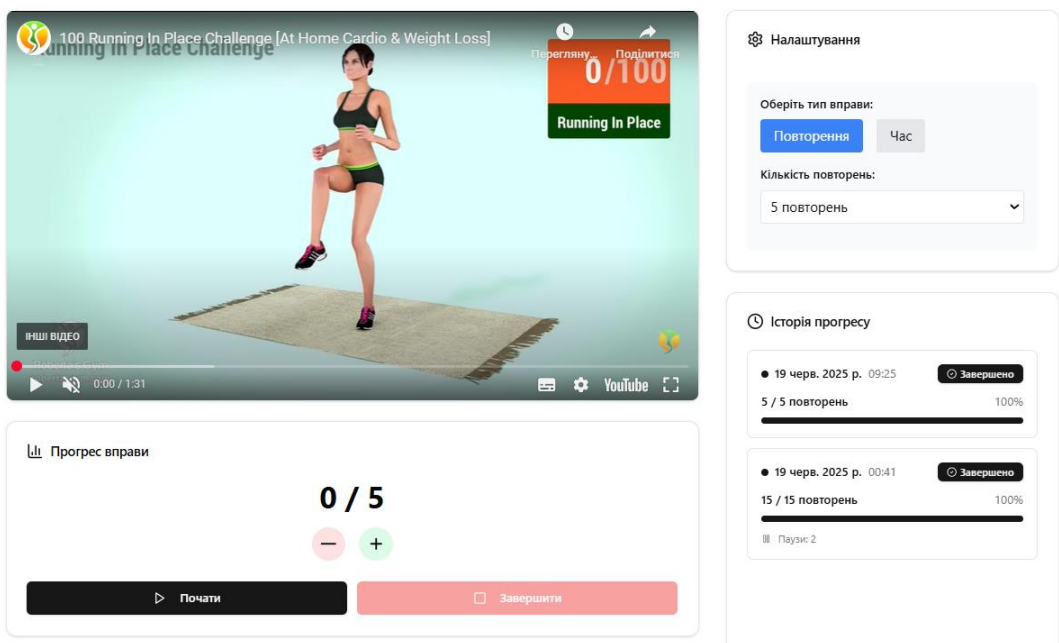


Рисунок 3.14 – Успішне завершення вправи за повторами

Фільтрація відбувається зручно за категоріями. Наявні такі категорії: кардіо, силові вправи, руки, розтяжка, прес рис.3.15.

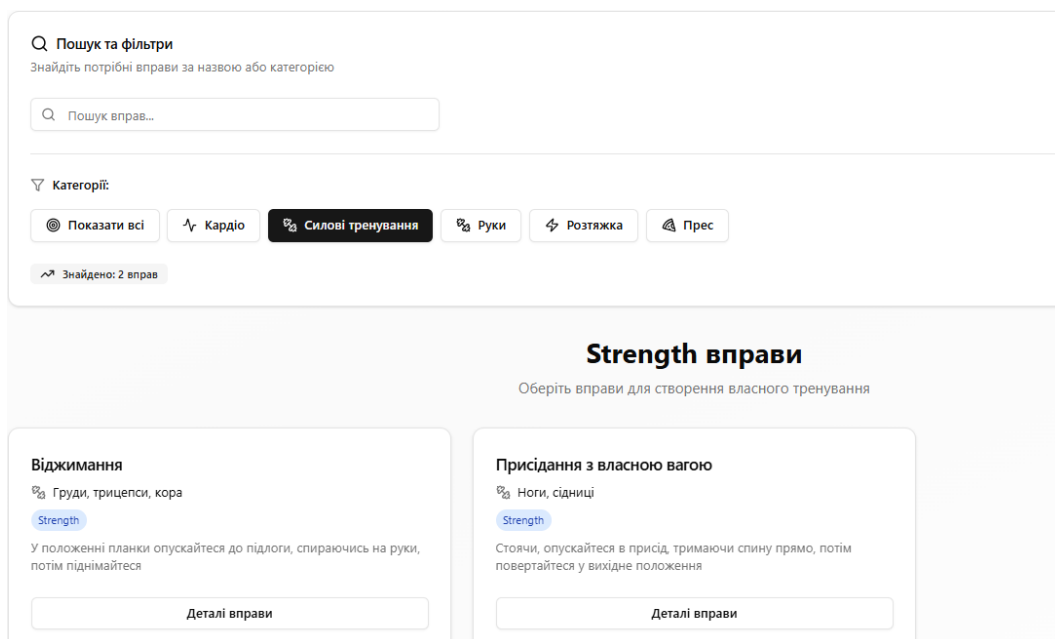


Рисунок 3.15 – Фільтрація вправ

Переходячи на сторінку планів. У користувача є можливість створити власний план або згенерувати план на основі цілі тренування. Це може бути схуднення, підтримка активності або ж набір м'язової маси. Дана сторінка вкдючає перегляд існуючих планів, з можливістю їх редагування та видалення рис.3.16.

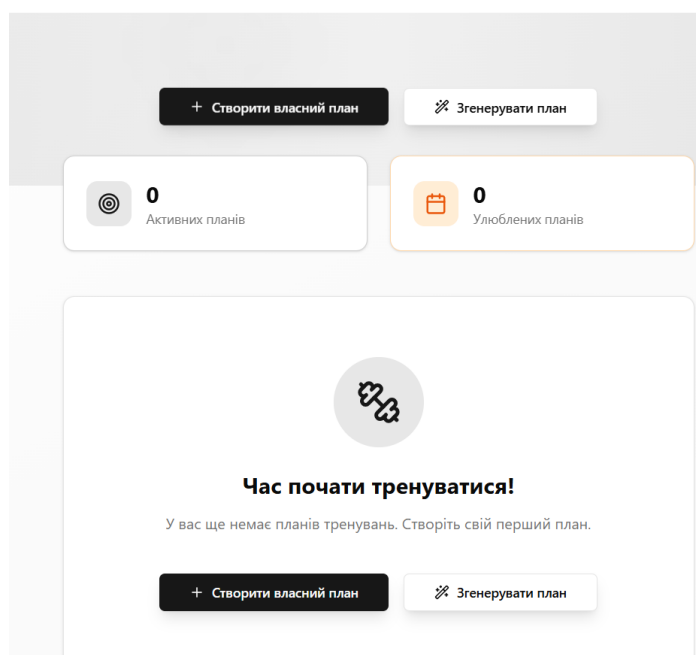


Рисунок 3.16 – Сторінка планів

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	46
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Щоб створити власний план користувач повинен дати назву плану, вказати опис та мету тренувань рис.3.17.

Створити новий план тренувань

Назва

тест

Опис

тестОпис

Мета

Схуднення

Скасувати

Створити план

Рисунок 3.17 – Модальне вікно для створення плану
Після створення плану, план видно на сторінці рис.3.18.

Створити власний план

Згенерувати план

1
Активних планів

0
Улюблених планів

тест

LOSE_WEIGHT

тестОпис

Переглянути деталі

Попередня

1

Наступна

Рисунок 3.18 – План створено

При переході на сторінку плану, користувач має можливість додавати вправ до плану, редагувати сам план, а саме назва та опис. Редагування відбувається через модальне вікно рис.3.19.

тест

Редагувати план

Виконати план

тестОпис

Статус: IN_PROGRESS

Додати вправу

Вправи у плані

Вправ ще немає.

Рисунок 3.19 – Сторінка плану

Коли користувач додає вправи до плану, то з'являється форма з елементами щодо вправи. Користувач повинен вибрати вправу з всіх існуючих, вказати її порядок виконання, вказати кількість повторень або тривалість виконання та кількість відпочинку в секундах рис.3.20.

Додати вправу до плану

×

Вправа

Скелелаз

Порядок

1

Повторення (reps)

5

Тривалість (секунди)

0

Відпочинок (секунди)

5

Скасувати

Додати

Рисунок 3.20 – Додавання вправи до плану

Вправу успішно додано до плану. Після додавання користувач бачить оновлений список вправ, з можливістю редагування вправ та самого плану рис.3.21.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	48
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

тест

[Редагувати план](#)[Виконати план](#)

тестОпис

Статус: IN_PROGRESS

[Додати вправу](#)

Вправи у плані

Скелелаз

Порядок: 1 | Повторення: 5 | Відпочинок: 5 сек

[Редагувати](#)[Видалити](#)

Рисунок 3.21 – Додавання вправи до плану

Додамо ще пару вправ для кращого прикладу та розуміння функціоналу рис.3.22.

тест

[Редагувати план](#)[Виконати план](#)

тестОпис

Статус: IN_PROGRESS

[Додати вправу](#)

Вправи у плані

Скелелаз

Порядок: 1 | Повторення: 5 | Відпочинок: 5 сек

[Редагувати](#)[Видалити](#)

Стрибки зі скакалкою

Порядок: 2 | Тривалість: 10 сек | Відпочинок: 10 сек

[Редагувати](#)[Видалити](#)

Віджимання

Порядок: 3 | Повторення: 5 | Відпочинок: 12 сек

[Редагувати](#)[Видалити](#)

Рисунок 3.22 – Поточні вправи

Також користувач з легкістю може видаляти, змінювати порядок, кількість повторів та тривалість виконання самої вправи рис.3.23.

тест

[Редагувати план](#)[Виконати план](#)

тестОпис

Статус: IN_PROGRESS

[Додати вправу](#)

Вправи у плані

Віджимання

Порядок: 1 | Повторення: 15 | Відпочинок: 12 сек

[Редагувати](#)[Видалити](#)

Стрибки зі скакалкою

Порядок: 2 | Тривалість: 20 сек | Відпочинок: 5 сек

[Редагувати](#)[Видалити](#)

Розтяжка спини

Порядок: 3 | Тривалість: 10 сек | Відпочинок: 4 сек

[Редагувати](#)[Видалити](#)

Рисунок 3.23 – Змінені вправи

На сторінці виконання плану, користувач бачить назву вправи її опис, відео для правильного виконання та лічильник з рахуванням кількості повторів рис.3.24.

тест

Крок 1 з 3

Віджимання

У положенні планки опускайтеся до підлоги, спираючись на руки, потім піднімайтеся



0 / 15

[Почати](#)[Скинути](#)

Рисунок 3.24 – Виконання плану

Після завершення вправи відбувається відпочинок із вказаними секундами рис.3.25.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

тест

Крок 1 з 3

Відпочинок: 11 сек

Рисунок 3.25 – Відпочинок після вправи

Ідентично також є лічильник на рахунок часу що залишився, де користувач в будь який момент виконання самаї вправи поставити на паузу або взагалі скинути лічильник, у такому випадку відрахунок почнеться заново рис.3.26.

тест

Крок 2 з 3

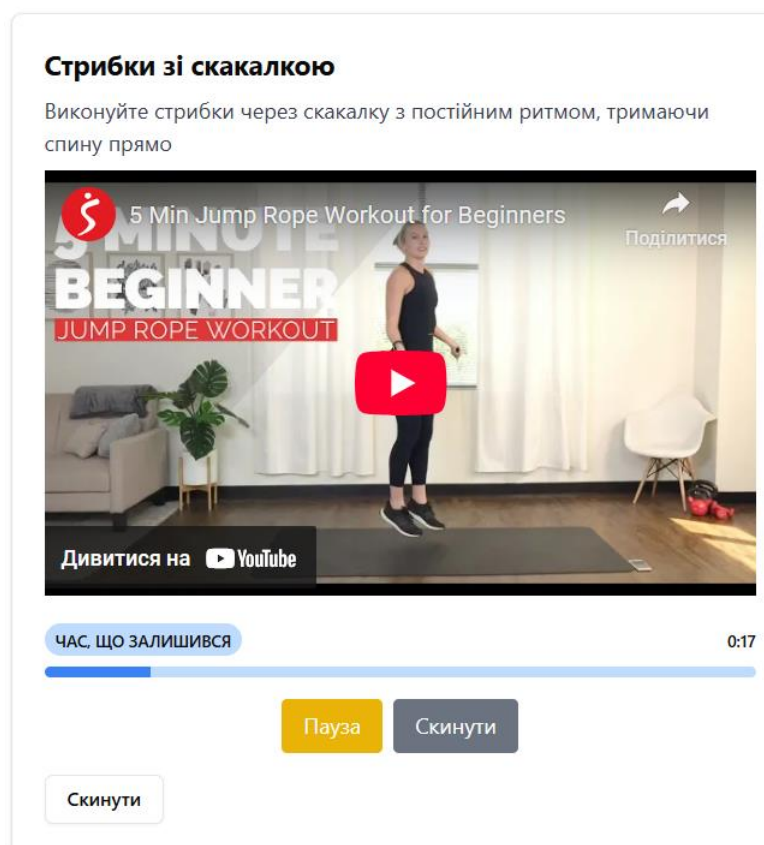


Рисунок 3.26 – Лічильник виконання

Завершення плану відображає саме повідомлення та кнопку, яка повертає користувача на сторінку плану рис.3.27.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	51
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

План завершено!

Зберегти результат

Рисунок 3.27 – Повідомлення про завершення плану
Після виконання плану, він позначається як виконаний рис.3.28.

тест

Редагувати план

Виконати план

тестОпис

Статус: COMPLETED

Додати вправу

Вправи у плані

Віджимання

Порядок: 1 | Повторення: 15 | Відпочинок: 12 сек

Редагувати

Видалити

Стрибки зі скалкою

Порядок: 2 | Тривалість: 20 сек | Відпочинок: 5 сек

Редагувати

Видалити

Розтяжка спини

Порядок: 3 | Тривалість: 10 сек | Відпочинок: 4 сек

Редагувати

Видалити

Рисунок 3.28 – Зміна статусу виконання плану

Якщо користувач не хоче додавати та довго міркувати яку вправу додати серед переліку усіх існуючих з бібліотеки, то є можливість згенерувати план на основі цілі тренувань на визначену кількість днів. Ці вправи які будуть додані до плану, також будуть відображатися в календарі. Тобто якщо користувач обрав цілі для тренування – «Підтримка активності», то на основі даної мети для користувача будуть автоматично додані вправи на основі категорій та груп м’язів, які найкраще підходять для вказаної цілі рис.3.29-3.31.

Створення персонального плану тренувань

Параметри плану

Введіть ваші фізичні параметри та побажання для створення оптимального плану тренувань

Ціль тренувань

Підтримка активності

Кількість днів на тиждень

2 днів

Створити план тренувань

Рисунок 3.29 – Створення плану

Plan for STAY_ACTIVE

Редагувати план

Виконати план

Статус: IN_PROGRESS

Додати вправу

Вправи у плані

Розтяжка спини

Порядок: 1 | Повторення: 10 | Відпочинок: 60 сек

Редагувати

Видалити

Розтяжка ніг

Порядок: 1 | Повторення: 10 | Відпочинок: 60 сек

Редагувати

Видалити

Присідання з власною вагою

Порядок: 2 | Повторення: 10 | Відпочинок: 60 сек

Редагувати

Видалити

arms

Порядок: 2 | Повторення: 10 | Відпочинок: 60 сек

Редагувати

Видалити

Стрибки з підніманням колін до грудей

Порядок: 3 | Тривалість: 30 сек | Відпочинок: 60 сек

Редагувати

Видалити

Розтяжка спини

Порядок: 3 | Повторення: 10 | Відпочинок: 60 сек

Редагувати

Видалити

Рисунок 3.30 – Згенерований план

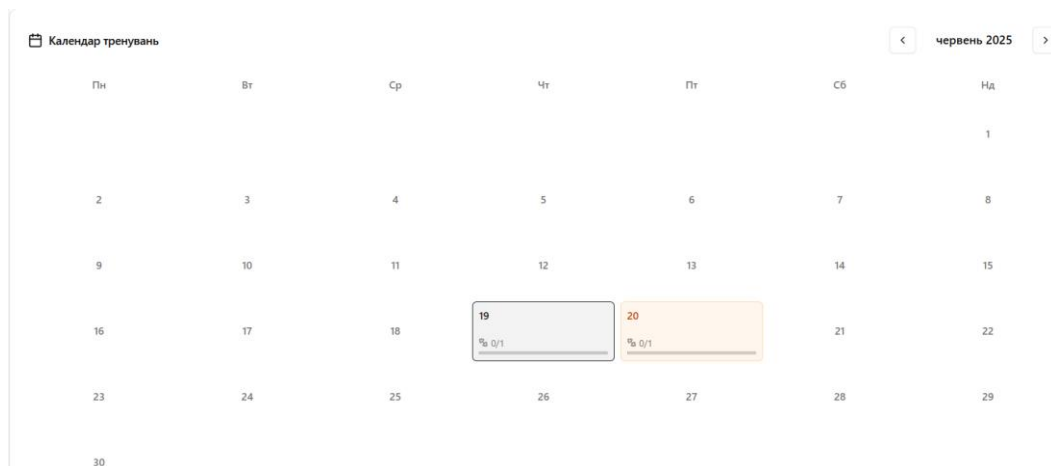


Рисунок 3.31 – Календар з вправами

На кожен день для користувача буде задано три вправи на виконання, календ відображає вправи, які потрібно виконати рис.3.32.

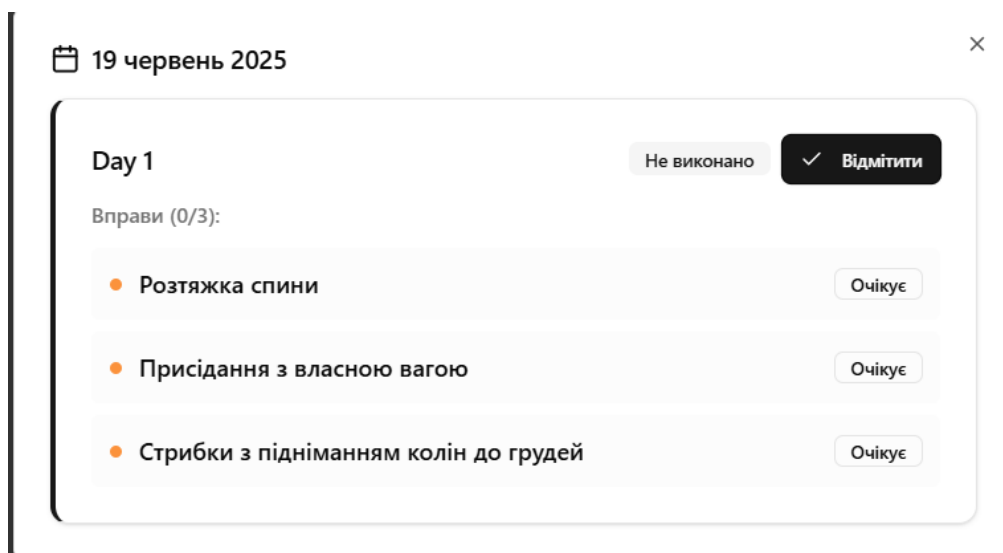


Рисунок 3.32 – Щоденний розклад

Після виконання вправи день з вправами позначається як виконаний рис.3.33.

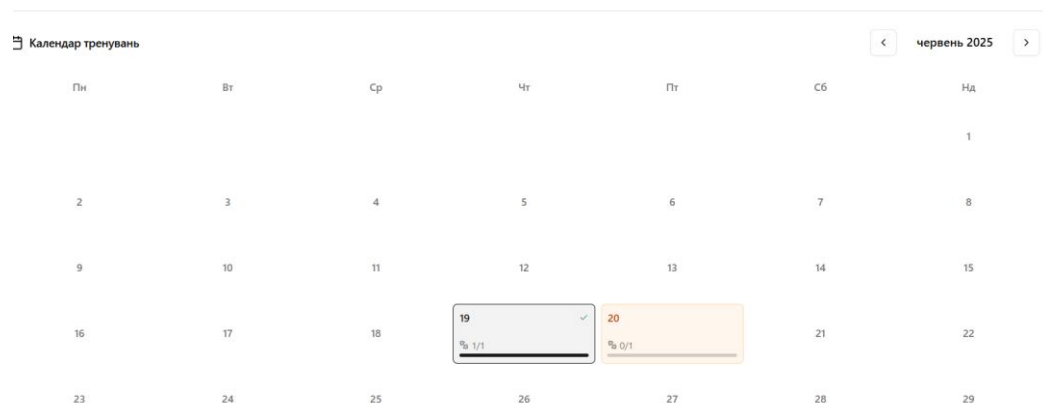


Рисунок 3.33 – Виконаний день з вправами

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	54
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Впроваджено отримання даних з фітнес-браслетів користувачів за допомогою StravaAPI. Для того, щоб користувач побачив свої дані на сторінці активності, він повинен бути вже зареєстрований в Strava. Користувач отримує детальну аналітику, що відображається на сторінці активності, яка надає огляд таких показників, як дистанція, виконаний час, кількість калорій, мапу проходження активності. В аналітика також входить діаграми в яких розподілено тип активності, також є діаграма з аналізом відстані за типом діяльності. Це дозволяє користувачу аналізувати свій прогрес, порівнювати результати з попередніми спробами рис.3.34-3.36.

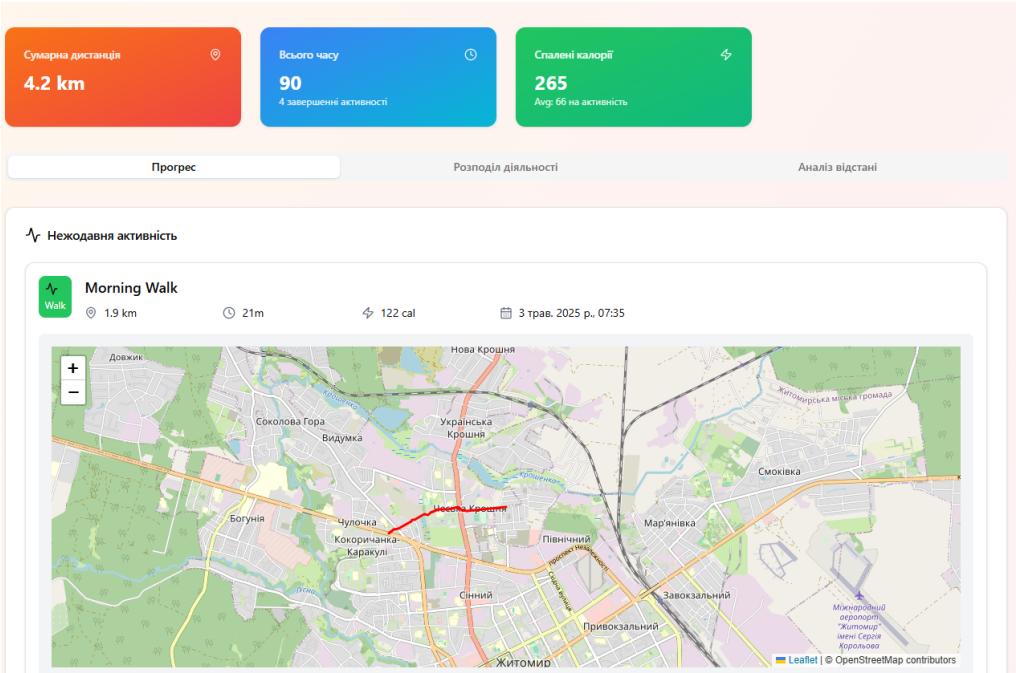


Рисунок 3.34 – Детальна інформація про активність

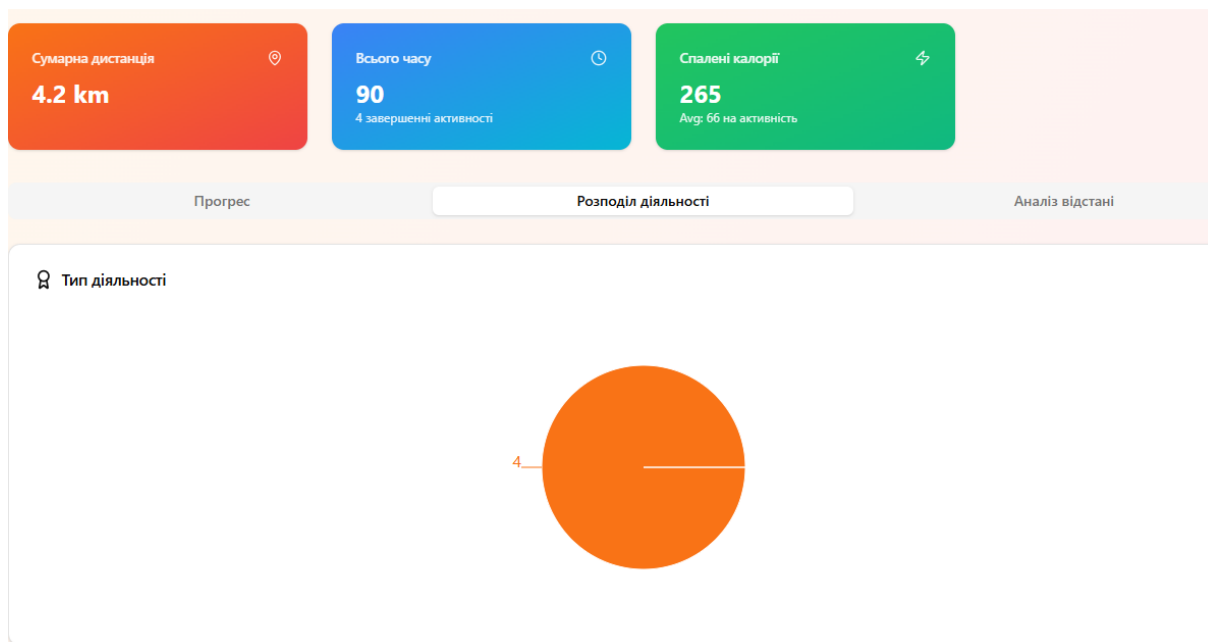


Рисунок 3.35 – Діаграма «Тип діяльності»

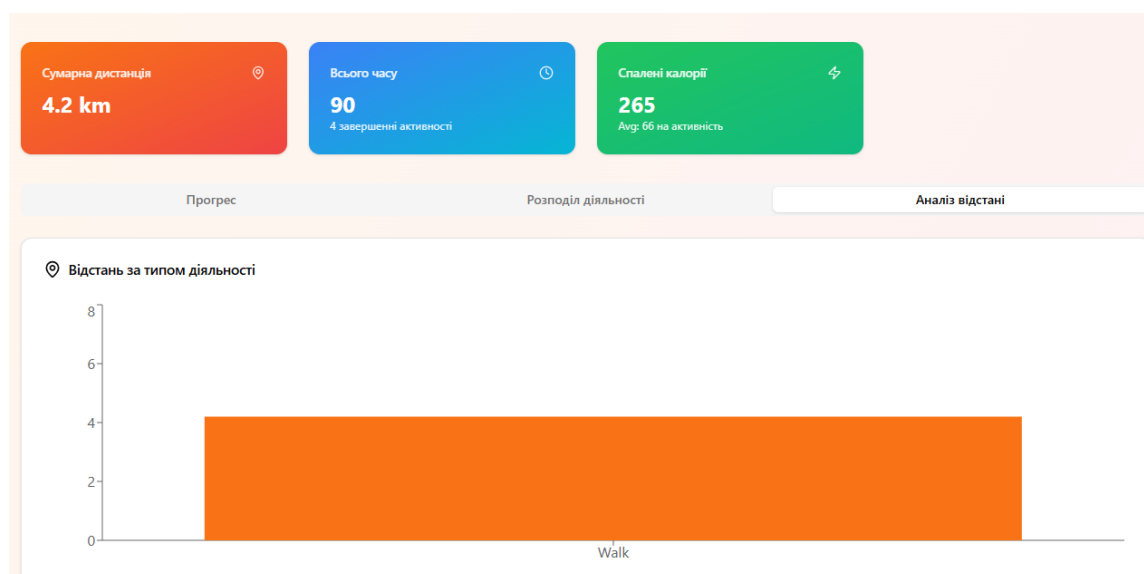


Рисунок 3.36 – Діаграма «Відстань за типом діяльності»

Також на сторінці Профілю користувач може змінювати власне ім'я та зображення, бачити статистику щодо виконаних вправ та кількість власних планів, переглядати свої останні плани рис.3.37.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

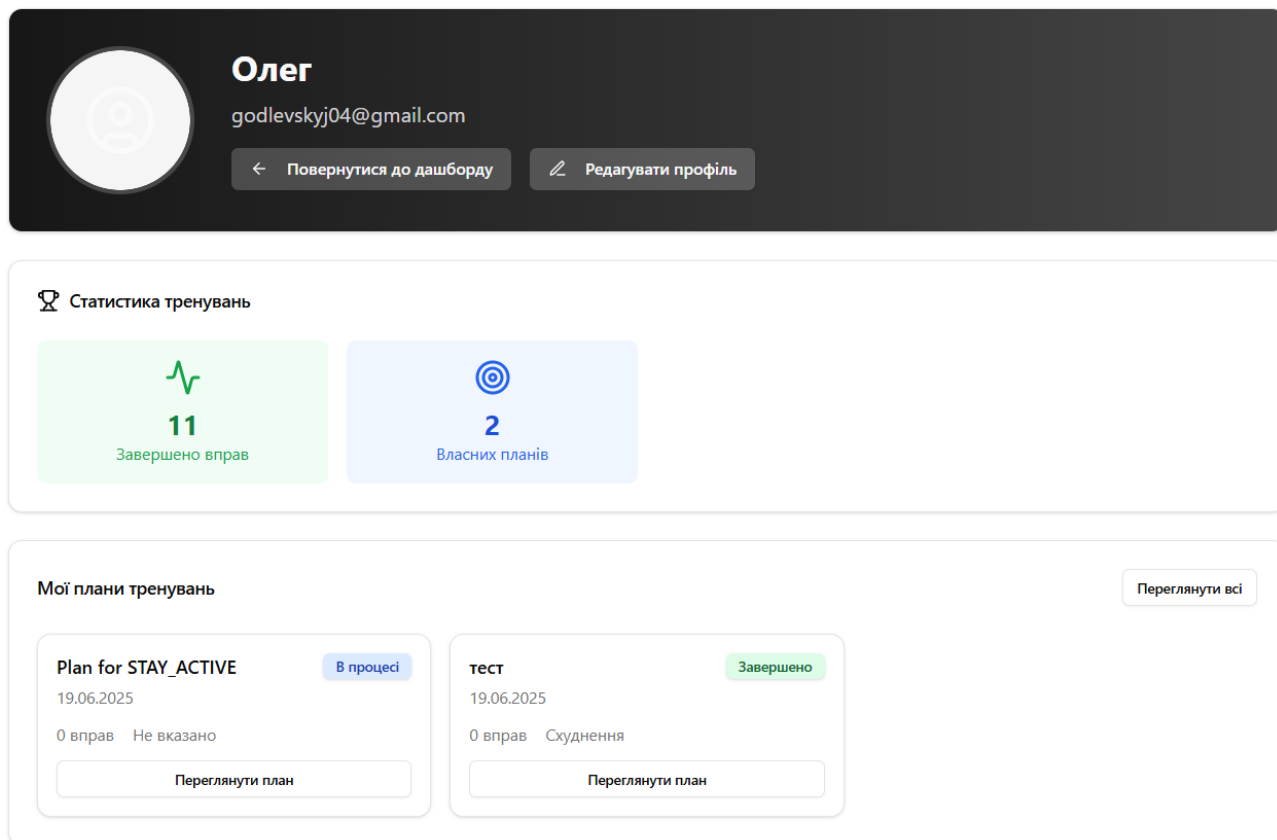


Рисунок 3.37 – Сторінка профілю

3.3 Тестування роботи програмного комплексу

Тестування вебплатформи є критично важливим етапом розробки для забезпечення стабільної роботи та високої якості користувацького досвіду. Виконано детально комплексне тестування для всіх функцій системи.

Тест 1: Автентифікація через електронну пошту

Передумови: користувач не авторизований у системі;

Кроки: введення електронної пошти "user@gmail.com" та пароля "Password2025", натискання кнопки "Увійти";

Очікуваний результат: користувач успішно авторизується та потрапляє на панель керування;

Фактичний результат: користувач успішно авторизується та потрапляє на панель керування.

Тест 2: Синхронізація з Strava

Передумови: користувач на сторінці активності;

Кроки: виконати вправу з фітнес - браслетом

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	57
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Очікуваний результат: синхронізація даних та відображення на сторінці активності.;

Фактичний результат: система змогла синхронізувати дані та відобразити їх на сторінці активності.

Тест 3: Фільтрація вправ

Передумови: користувач на сторінці бібліотеки вправ;

Кроки: вибір категорії для фільтрації, наприклад – Силові тренування;

Очікуваний результат: відображається тільки ті вправи, які входять у дану категорію;

Фактичний результат: відображаються вправи з категорії – Силові вправи.

Тест 4: Пошук вправ по назві

Передумови: користувач знаходиться на сторінці бібліотеки вправ;

Кроки: вводить назву вправи;

Очікуваний результат: відображення вправ по назві, у випадку не знаходження вправи, відображення неіснуючої вправи.

Фактичний результат: вправа відображається по назві.

Тест 5: Створення плану

Передумови: користувач знаходиться на сторінці планів;

Кроки: натиснути на кнопку «Створити власний план» та вести дані щодо даного плану.

Очікуваний результат: План збережеться на відобразиться на сторінці планів.

Фактичний результат: план створено та відображено на сторінці планів.

Тест 6: Видалення плану

Передумови: користувач знаходиться на сторінці планів;

Кроки: натиснути на кнопку з іконкою Кошик та підтвердити намір видалити план.

Очікуваний результат: план буде видалено.

Фактичний результат: план видалився та зник з сторінки планів.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	58
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Тест 7: Зміна зображення у профілі

Передумови: користувач знаходиться на сторінці профілю;

Кроки: натиснути на кнопку «Редагуват профіль», де з'явиться форма зі зміною імені та зображення, потрібно вибрати файл та натиснути на кнопку «Зберегти зміни»;

Очікуваний результат: зображення зміниться на обране

Фактичний результат: зображення змінено.

Висновки до розділу 3

У даному розділі описано процес встановлення та налаштування системи, яка побудована на клієнт-серверній архітектурі з використанням Next.js для фронтенду та NestJS для бекенду. Для розуміння структури було створено діаграму розгортання, що демонструє взаємодію основних компонентів системи.

Описано компоненти інтерфейсу: головну сторінку, форми реєстрації та входу, розділи бібліотеки вправ, планів, сторінки виконання тренувань та профілю користувача. Розглянуто кожну функцію, яка є на цих сторінках – зокрема, можливість фільтрувати вправи, створювати персональні плани, відстежувати прогрес та синхронізувати дані з фітнес-браслетами чи сервісом Strava.

Також описано етап тестування системи: було перевірено автентифікацію, синхронізацію з API, роботу фільтрів, створення та видалення тренувальних планів, а також редагування профілю.

У підсумку вдалося реалізувати стабільну, масштабовану та зручну для користувачів платформу, яка дозволяє ефективно стежити за фізичною активністю, зберігає дані надійно та відповідає усім поставленим вимогам щодо функціональності, інтеграцій і безпеки.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	59
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Під час роботи над бакалаврською кваліфікаційною роботою було реалізовано вебплатформу для відстеження фізичних вправ і аналізу прогресу користувачів. На початку було проведено аналіз теми та порівняння з аналогами. Це дало змогу чітко визначити вимоги до майбутньої системи та визначити основні вимоги.

В основі платформи лежить клієнт-серверна архітектура: фронтенд реалізовано на Next.js, бекенд – на NestJS, а для зберігання даних використано PostgreSQL. Також додано інтеграцію з зовнішніми сервісами, такими як Strava API. Завдяки цьому платформа відзначається масштабованістю та безпекою.

У рамках розробки створено діаграми розгортання та описано структуру інтерфейсу, що включає сторінки авторизації, бібліотеку вправ, планування і виконання тренувань, а також сторінку профілю користувача. Після реалізації було проведено повноцінне тестування ключових функцій: перевірка авторизації, синхронізації з API, фільтрації, створення і редагування планів, а також оновлення даних профілю – усе це підтвердило стабільну роботу системи.

У результаті вдалося створити повноцінну платформу, яка допомагає користувачам стежити за власними тренуваннями, аналізувати прогрес і синхронізувати дані з фітнес-пристроями. В подальшому можна покращити та додати новий функціонал – соціальні функції, підтримку різних мов для зручності користування.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pressman R.S. Software Engineering: A Practitioner's Approach / R.S. Pressman. – Boston: McGraw-Hill Education, 2019. – 950 p.
2. Sommerville I. Software Engineering / I. Sommerville. – London: Pearson, 2015. – 820 p.
3. Bass L., Clements P., Kazman R. Software Architecture in Practice / L. Bass, P. Clements, R. Kazman. – Boston: Addison-Wesley, 2012. – 600 p.
4. Flanagan D. JavaScript: The Definitive Guide / D. Flanagan. – O'Reilly Media, 2020. – 700 p.
5. Resig J. Pro JavaScript Techniques / J. Resig. – New York: Apress, 2006. – 300p.
6. Black R. Managing the Testing Process / R. Black. – New York: Wiley, 2009. – 380 p.
7. Martin R.C. Clean Code: A Handbook of Agile Software Craftsmanship / R.C. Martin. – Upper Saddle River: Prentice Hall, 2008. – 464 p.
8. Myers G.J. The Art of Software Testing / G.J. Myers. – New York: Wiley, 2011. – 245 p.
9. Kaner C., Bach J., Pettichord B. Lessons Learned in Software Testing / C. Kaner, J. Bach, B. Pettichord. – New York: Wiley, 2001. – 320p.
10. Fowler M. Patterns of Enterprise Application Architecture / M. Fowler. – Boston: Addison – Wesley, 2002. – 560 p.
11. Hunt A., Thomas D. The Pragmatic Programmer / A. Hunt, D. Thomas. – Boston: Addison – Wesley, 1999. – 352 p.
12. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software / E. Evans. – Boston: Addison – Wesley, 2003. – 560 p.

		Годлевський О.О.			ІІЗ.КР.Б-121-25-ІІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

13. Freeman E., Robson E., Bates B., Sierra K. Head First Design Patterns / E. Freeman et al. – O'Reilly Media, 2004. – 675 p.
14. Beizer B. Software Testing Techniques / B. Beizer. – New York: Van Nostrand Reinhold, 1990. – 550p.
15. Craig R.D., Jaskiel S.P. Systematic Software Testing / R.D. Craig, S.P. Jaskiel. – Boston: Artech House, 2002. – 400p.
16. IEEE Standards Association. IEEE 829-2008: Software and System Test Documentation / IEEE. – New York: IEEE, 2008. – 150 p.
17. Hinchliffe D. Building Node.js Application / D.Hinchliffe. – Packt Publishing, 2019. – 350 p.
18. Chacon S., Straub B. Pro Git / S.Chacon, B. Straub. – New York: Apress , 2014. – 450 p.
19. Osmani A. Learning JavaScript Design Patterns / A.Osmani. – O'Reilly Media, 2012. – 250 p.
20. Coplien J.O., Schmidt D.C. Pattern Languages of Program Design / J.O. Coplien, D.C. Schmidt. – Boston: Addison-Wesley, 1995. – 570 p.
21. Fogel K. Producing Open Source Software / K. Fogel. – O'Reilly Media, 2005. – 300 p.
22. Vercel Documentation. [Електронний ресурс]. – Режим доступу: <https://vercel.com/docs>
23. Nodemailer Documentation [Електронний ресурс]. – Режим доступу: <https://nodemailer.com/>
24. NextJs Official Documentation [Електронний ресурс]. – Режим доступу: <https://nextjs.org/docs>

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

25. NestJs Official Documentation [Електронний ресурс]. – Режим доступу:
<https://docs.nestjs.com/>
26. PostgreSQL Official Documentation [Електронний ресурс]. – Режим доступу: <http://postgresql.org/docs/>
27. Strava Official Documentation [Електронний ресурс]. – Режим доступу:
<https://developers.strava.com/docs/>
28. TypeScript Official Documentation [Електронний ресурс]. – Режим доступу: <https://www.typescriptlang.org/docs/>
29. Cloudinary Official Documentation [Електронний ресурс]. – Режим доступу: <https://cloudinary.com/documentation>
30. Prisma Official Documentation – [Електронний ресурс]. – Режим доступу:
<https://www.prisma.io/docs>
31. Json Documentation – [Електронний ресурс]. – Режим доступу:
<https://jwt.io/introduction>
32. Годлевський О. О., Фант М. О. Безпечна аутентифікація та управління користувачами у вебдодатках на основі Nest.js та Next.js. Тези XV Міжнародної науково-технічної конференції «Інформаційно-комп'ютерні технології», 28-29 березня 2025 року. Житомир : Державний університет «Житомирська політехніка», 2025. С. 149–150.

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

ДОДАТКИ

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

Лістинг фрагменту коду auth.service.ts

```

async login(email: string, password: string, res: Response) {
    const user = await this.prisma.user.findUnique({ where: { email } });
    if (!user) {
        throw new HttpException('Invalid credentials',
HttpStatus.UNAUTHORIZED);
    }

    const isPasswordValid = await bcrypt.compare(password, user.password);
    if (!isPasswordValid) {
        throw new HttpException('Invalid credentials',
HttpStatus.UNAUTHORIZED);
    }

    // Перевірка, чи користувач верифікований
    if (!user.isVerified) {
        throw new HttpException(
            'Please verify your email before logging in',
            HttpStatus.FORBIDDEN,
        );
    }

    const tokens = this.generateToken(user);
    this.setCookies(res, tokens);

    return {
        message: 'Login successful',
    };
}

{
    throw new Error('Failed to send verification email');
}
}

```

Лістинг фрагменту коду auth.service.ts

```

async register(name: string, email: string, password: string) {

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

```

const existingUser = await this.prisma.user.findUnique({
  where: { email },
});

if (existingUser) {
  throw new Error('User already exists');
}

const hashedPassword = await bcrypt.hash(password, 10);

// Створюємо унікальний токен для верифікації
const verificationToken = crypto.randomBytes(32).toString('hex');

const user = await this.prisma.user.create({
  data: {
    name,
    email,
    password: hashedPassword,
    isVerified: false,
    verificationToken,
    goal: null,
  },
});

// Формуємо URL для верифікації
const verificationUrl =
`${process.env.FRONTEND_URL}/verify?token=${verificationToken}`;

// HTML шаблон листа
const htmlContent = `
<div style="font-family: Arial, sans-serif; max-width: 600px; margin:
0 auto;">

  <h2>Welcome to Our App!</h2>
  <p>Hello ${name},</p>
  <p>Thank you for registering. Please verify your email by clicking
the button below:</p>
  <div style="text-align: center; margin: 30px 0;">
    <a href="${verificationUrl}"
      style="background-color: #4F46E5; color: white; padding: 12px
24px;
      text-decoration: none; border-radius: 5px;">
      Verify Email
    </a>

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

```

        </div>
        <p>If the button doesn't work, you can copy and paste this link into
your browser:</p>
        <p>${verificationUrl}</p>
    </div>
    `;

    const mailOptions = {
        from: process.env.EMAIL_USER,
        to: email,
        subject: 'Welcome to Our App - Verify Your Email',
        html: htmlContent,
    };

    try {
        await this.transporter.sendMail(mailOptions);
        return {
            message:
                'Registration successful. Please check your email to verify your
account.',
        };
    } catch (error) {
        throw new Error('Failed to send verification email');
    }
}

```

Лістинг фрагменту коду planexercise.controller.ts:

```

import {
    Body,
    Controller,
    Delete,
    Get,
    Param,
    ParseIntPipe,
    Patch,
    Post,
    Request,
    UseGuards,
} from '@nestjs/common';
import { JwtAuthGuard } from '../auth/jwt-auth.guard';
import { CreatePlanDto } from '../dto/createPlan.dto';

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

```

import { GeneratePlanDto } from '../dto/generatePlan.dto';
import { PlanExerciseService } from '../planexercise.service';
import { UpdatePlanDto } from '../plansexercise.dto';

@Controller('dashboard/plans')
@UseGuards(JwtAuthGuard)
export class PlansExerciseController {
  logger: any;

  constructor(private readonly workoutPlanService: PlanExerciseService) {}

  @Post()
  create(@Body() dto: CreatePlanDto, @Request() req) {
    const userId = req.user.userId;
    return this.workoutPlanService.create(userId, dto);
  }

  @Get()
  findAll(@Request() req) {
    const userId = req.user.userId;
    return this.workoutPlanService.findAll(userId);
  }

  @Get('/:id')
  findOne(@Param('id', ParseIntPipe) id: number, @Request() req) {
    const userId = req.user.userId;
    return this.workoutPlanService.findOne(id, userId);
  }

  @Patch('/:id')
  update(
    @Param('id', ParseIntPipe) id: number,
    @Body() dto: UpdatePlanDto,
    @Request() req,
  ) {
    const userId = req.user.userId;
    return this.workoutPlanService.updatePlan(id, dto, userId);
  }

  @Delete('/:id')
  remove(@Param('id', ParseIntPipe) id: number, @Request() req) {
    const userId = req.user.userId;
    return this.workoutPlanService.remove(id, userId);
  }
}

```

		Годлевський О.О.			ІІЗ.КР.Б-121-25-ІІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

```

@Post('/:id/exercises')
addExerciseToPlan(
  @Param('id', ParseIntPipe) planId: number,
  @Body()
  dto: {
    exerciseId: number;
    order: number;
    reps?: number;
    duration?: number;
    restDuration?: number;
  },
  @Request() req,
) {
  const userId = req.user.userId;
  return this.workoutPlanService.addExerciseToPlan(planId, userId, dto);
}

@Delete('/:plandId/exercises/:exerciseId')
async removeExerciseFromPlan(
  @Param('plandId', ParseIntPipe) planId: number,
  @Param('exerciseId', ParseIntPipe) exerciseId: number,
  @Request() req,
) {
  const userId = req.user.userId;
  return this.workoutPlanService.removeExerciseFromPlan(
    planId,
    exerciseId,
    userId,
  );
}

@Patch('/:planId/exercises/:exerciseId')
async updateExerciseInPlan(
  @Param('planId', ParseIntPipe) planId: number,
  @Param('exerciseId', ParseIntPipe) exerciseId: number,
  @Body()
  dto: {
    order: number;
    reps?: number;
    duration?: number;
    restDuration?: number;
  },
  @Request() req,

```

		Годлевський О.О.			ІІЗ.КР.Б-121-25-ІІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

```

    ) {
        const userId = req.user.userId;
        return this.workoutPlanService.updateExerciseInPlan(
            planId,
            exerciseId,
            userId,
            dto,
        );
    }

    @Post('/:id/complete')
    async completePlan(
        @Param('id', ParseIntPipe) planId: number,
        @Body()
        body: {
            trackedExercises: {
                exerciseId: number;
                type: 'reps' | 'time';
                value: number;
                targetValue: number;
                completed: boolean;
                pauseCount: number;
                resetCount: number;
            }[];
            status: 'COMPLETED' | 'CANCELLED';
        },
        @Request() req,
    ) {
        const userId = req.user.userId;
        return this.workoutPlanService.completePlan(planId, userId, body);
    }

    @Get('/:id/progress')
    @UseGuards(JwtAuthGuard)
    async getPlanProgress(
        @Param('id', ParseIntPipe) planId: number,
        @Request() req,
    ) {
        const userId = req.user.userId;
        return this.workoutPlanService.getPlanProgress(userId, planId);
    }

    //збереження проміжного прогресу

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

```

@Post('/:id/progress')
async savePartialProgress(
  @Param('id', ParseIntPipe) planId: number,
  @Body()
  body: {
    trackedExercises: {
      exerciseId: number;
      type: 'reps' | 'time';
      value: number;
      targetValue: number;
      completed: boolean;
      pauseCount: number;
      resetCount: number;
    }[];
  },
  @Request() req,
) {
  const userId = req.user.userId;
  return this.workoutPlanService.savePartialProgress(planId, userId, body);
}

@Post('generate')
async generatePlan(@Body() dto: GeneratePlanDto, @Request() req) {
  const userId = req.user.userId;
  return this.workoutPlanService.generatePlan(userId, dto);
}

@Get()
async getPlans(@Request() req) {
  const userId = req.user.userId;
  return this.workoutPlanService.findAll(userId);
}

@Get('/:id/exercises')
async getPlanExercises(
  @Param('id', ParseIntPipe) planId: number,
  @Request() req,
) {
  const userId = req.user.userId;
  return this.workoutPlanService.getPlanExercises(planId, userId);
}

@Get('calendar/:month/:year')
@UseGuards(JwtAuthGuard)

```

		Годлевський О.О.			ІІЗ.КР.Б-121-25-ІІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

```

async getCalendar(
  @Param('month') month: string,
  @Param('year') year: string,
  @Request() req,
) {
  const userId = req.user.userId;
  const monthNum = parseInt(month, 10);
  const yearNum = parseInt(year, 10);

  if (
    isNaN(monthNum) ||
    monthNum < 1 ||
    monthNum > 12 ||
    isNaN(yearNum) ||
    yearNum < 2000 ||
    yearNum > 2100
  ) {
    return { error: 'Invalid month or year' };
  }

  const calendar = await this.workoutPlanService.getCalendar(
    userId,
    monthNum,
    yearNum,
  );
  return calendar;
}

@Patch('day/:planId/:dayId/complete')
@UseGuards(JwtAuthGuard)
async markDayAsCompleted(
  @Param('planId') planId: string,
  @Param('dayId') dayId: string,
  @Request() req,
) {
  const userId = req.user.userId;
  const planIdNum = parseInt(planId, 10);
  const dayIdNum = parseInt(dayId, 10);

  if (isNaN(planIdNum) || isNaN(dayIdNum)) {
    return { error: 'Invalid planId or dayId' };
  }
}

```

		Годлевський О.О.			ІІЗ.КР.Б-121-25-ІІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

```

        await this.workoutPlanService.markDayAsCompleted(
            userId,
            planIdNum,
            dayIdNum,
        );
        return { success: true };
    }
}

```

Лістинг фрагменту коду planexercise.service.ts:

```

import {
    ForbiddenException,
    Injectable,
    NotFoundException,
} from '@nestjs/common';
import { PrismaService } from '../../prisma/prisma.service';
import { CreatePlanDto } from '../dto/createPlan.dto';
import { UpdatePlanDto } from '../dto/updatePlan.dto';

@Injectable()
export class PlanExerciseService {
    logger: any;

    constructor(private prisma: PrismaService) {}

    async create(userId: number, createPlanDto: CreatePlanDto) {
        return this.prisma.workoutPlan.create({
            data: {
                ...createPlanDto,
                userId,
                status: 'IN_PROGRESS',
            },
        });
    }

    async findAll(userId: number) {
        return this.prisma.workoutPlan.findMany({
            where: { userId },
            include: {
                WorkoutDay: {
                    include: {
                        exercises: {
                            include: {
                                exercise: true,

```

```

        },
        },
        },
        },
        },
        orderBy: { createdAt: 'desc' },
    });
}

```

```

async findOne(id: number, userId: number) {
    const plan = await this.prisma.workoutPlan.findFirst({
        where: {
            id,
            userId,
        },
        include: {
            WorkoutDay: {
                include: {
                    exercises: {
                        include: {
                            exercise: true,
                        },
                    },
                },
            },
        },
    });

    if (!plan) {
        throw new NotFoundException(`План з ID ${id} не знайдено`);
    }

    return plan;
}

async updatePlan(id: number, updatePlanDto: UpdatePlanDto, userId: number)
{
    console.log(
        `Service: Updating plan ${id} for user ${userId} with data:`,
        updatePlanDto,
    );

    // Перевіряємо, чи існує план

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

const plan = await this.prisma.workoutPlan.findFirst({
  where: {
    id,
  },
});

if (!plan) {
  console.log(`Plan ${id} not found`);
  throw new NotFoundException(`План з ID ${id} не знайдено`);
}

// Перевіряємо, чи належить план користувачу
if (plan.userId !== userId) {
  console.log(
    `Plan ${id} does not belong to user ${userId}, it belongs to user ${plan.userId}`,
  );
  throw new ForbiddenException(`План не належить користувачу`);
}

try {
  const updatedPlan = await this.prisma.workoutPlan.update({
    where: { id },
    data: updatePlanDto,
  });
  console.log(`Plan ${id} updated successfully`);
  return updatedPlan;
} catch (error) {
  console.error(`Error updating plan ${id}:`, error);
  throw error;
}
}

async remove(id: number, userId: number) {
  await this.findOne(id, userId);
  return this.prisma.workoutPlan.delete({
    where: { id },
  });
}

async addExerciseToPlan(
  planId: number,
  userId: number,

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

```

    dto: {
      exerciseId: number;
      order: number;
      reps?: number;
      duration?: number;
      restDuration?: number;
    },
  ) {
    const plan = await this.findOne(planId, userId);
    if (!plan) throw new NotFoundException('Workout plan not found');

    // Знаходимо перший день тренування або створюємо новий, якщо немає днів
    let workoutDayId: number;

    if (plan.WorkoutDay && plan.WorkoutDay.length > 0) {
      workoutDayId = plan.WorkoutDay[0].id;
    } else {
      // Створюємо новий день тренування
      const newDay = await this.prisma.workoutDay.create({
        data: {
          workoutPlanId: planId,
          dayNumber: 1,
          name: 'День 1',
        },
      });
      workoutDayId = newDay.id;
    }

    return this.prisma.workoutPlanExercise.create({
      data: {
        exerciseId: dto.exerciseId,
        order: dto.order,
        reps: dto.reps,
        duration: dto.duration,
        restDuration: dto.restDuration,
        workoutPlanId: planId,
        workoutDayId: workoutDayId,
        isCompleted: false,
      },
    });
  }

  async getPlanExercises(planId: number, userId: number) {
    const plan = await this.findOne(planId, userId);

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

```

    if (!plan) throw new NotFoundException('Workout plan not found');

    const exercises = await this.prisma.workoutPlanExercise.findMany({
      where: { workoutPlanId: planId },
      include: {
        exercise: {
          select: { id: true, name: true, videoUrl: true }, // Явно вказуємо
поля
        },
      },
      orderBy: { order: 'asc' },
    });

    console.log('Повернуті вправи:', exercises);
    return exercises;
  }

  async removeExerciseFromPlan(
    planId: number,
    exerciseId: number,
    userId: number,
  ) {
    const plan = await this.findOne(planId, userId);
    if (!plan) throw new NotFoundException('Workout plan not found');

    return this.prisma.workoutPlanExercise.deleteMany({
      where: {
        workoutPlanId: planId,
        id: exerciseId,
      },
    });
  }

  async updateExerciseInPlan(
    planId: number,
    exerciseId: number,
    userId: number,
    dto: {
      order: number;
      reps?: number;
      duration?: number;
      restDuration?: number;
    },
  ) {

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

```

const plan = await this.findOne(planId, userId);
if (!plan) throw new NotFoundException('Workout plan not found');

return this.prisma.workoutPlanExercise.update({
  where: { id: exerciseId, workoutPlanId: planId },
  data: dto,
});
}

async completePlan(
  planId: number,
  userId: number,
  body: {
    trackedExercises: {
      exerciseId: number;
      type: 'reps' | 'time';
      value: number;
      targetValue: number;
      completed: boolean;
      pauseCount: number;
      resetCount: number;
    }[];
    status: 'COMPLETED' | 'CANCELLED';
  },
) {
  //зберігаємо всі trackedExercises
  for (const ex of body.trackedExercises) {
    await this.prisma.trackedExercise.create({
      data: {
        userId,
        exerciseId: ex.exerciseId,
        type: ex.type,
        value: ex.value,
        targetValue: ex.targetValue,
        completed: ex.completed,
        pauseCount: ex.pauseCount,
        resetCount: ex.resetCount,
        workoutPlanId: planId,
        completedAt: new Date(),
      },
    });
  }

  await this.prisma.workoutPlan.update({

```

		Годлевський О.О.			ІІЗ.КР.Б-121-25-ІІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

```

        where: { id: planId },
        data: { status: body.status },
    });
    return { success: true };
}

async getPlanProgress(userId: number, planId: number) {
    // Отримуємо всі відстежені вправи для цього плану
    const trackedExercises = await this.prisma.trackedExercise.findMany({
        where: {
            userId,
            workoutPlanId: planId,
        },
        include: {
            exercise: true,
        },
        orderBy: {
            completedAt: 'desc',
        },
    });

    // Отримуємо загальну кількість вправ у плані
    const plan = await this.prisma.workoutPlan.findUnique({
        where: { id: planId },
        include: {
            WorkoutDay: {
                include: {
                    exercises: true,
                },
            },
        },
    });

    // Рахуємо загальну кількість вправ у плані
    const totalExercises = plan.WorkoutDay.reduce(
        (sum, day) => sum + day.exercises.length,
        0,
    );

    // Рахуємо кількість виконаних вправ
    const completedExercises = trackedExercises.filter(
        (ex) => ex.completed,
    ).length;

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

```

// Пахуємо прогрес у відсотках
const progressPercentage =
  totalExercises > 0
    ? Math.round((completedExercises / totalExercises) * 100)
    : 0;

return {
  trackedExercises,
  progress: {
    total: totalExercises,
    completed: completedExercises,
    percentage: progressPercentage,
  },
};
}

async savePartialProgress(
  planId: number,
  userId: number,
  body: {
    trackedExercises: {
      exerciseId: number;
      type: 'reps' | 'time';
      value: number;
      targetValue: number;
      completed: boolean;
      pauseCount: number;
      resetCount: number;
    }[];
  },
) {
  for (const ex of body.trackedExercises) {
    await this.prisma.trackedExercise.create({
      data: {
        userId,
        exerciseId: ex.exerciseId,
        type: ex.type,
        value: ex.value,
        targetValue: ex.targetValue,
        completed: ex.completed,
        pauseCount: ex.pauseCount,
        resetCount: ex.resetCount,
      },
    });
  }
}

```

		Годлевський О.О.			ІІЗ.КР.Б-121-25-ІІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

```

        workoutPlanId: planId,
        completedAt: new Date(),
    },
    });
}
return { success: true };
}

async getCalendar(userId: number, month: number, year: number) {
    console.log(
        'Fetching calendar for user:',
        userId,
        'month:',
        month,
        'year:',
        year,
    );

    const plans = await this.prisma.workoutPlan.findMany({
        where: { userId, status: 'IN_PROGRESS' },
        include: {
            WorkoutDay: {
                where: {
                    scheduledDate: {
                        gte: new Date(year, month - 1, 1),
                        lt: new Date(year, month, 1),
                    },
                },
                include: { exercises: { include: { exercise: true } } },
            },
        },
    });

    if (!plans.length) {
        const daysInMonth = new Date(year, month, 0).getDate();
        return {
            month,
            year,
            days: Array.from({ length: daysInMonth }, (_, i) => ({
                day: i + 1,
                hasWorkouts: false,
                workouts: [],
            })),
        };
    }
}

```

		Годлевський О.О.			ІІЗ.КР.Б-121-25-ІІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

```

    };
}

const calendarDays = [];
const daysInMonth = new Date(year, month, 0).getDate();
for (let day = 1; day <= daysInMonth; day++) {
    const date = new Date(year, month - 1, day);
    const dayWorkouts = plans.flatMap((plan) =>
        plan.WorkoutDay.filter((wd) => {
            const wdDate = new Date(wd.scheduledDate || date);
            return (
                wdDate.getDate() === day &&
                wdDate.getMonth() === month - 1 &&
                wdDate.getFullYear() === year
            );
        })).map((wd) => ({
            planId: plan.id,
            dayId: wd.id,
            dayName: wd.name || `Day ${wd.dayNumber}`,
            isCompleted: wd.isCompleted ?? false,
            exercises: wd.exercises.map((e) => ({
                id: e.id,
                name: e.exercise.name,
                isCompleted: e.isCompleted ?? false,
            })),
        })),
    );

    calendarDays.push({
        day,
        hasWorkouts: dayWorkouts.length > 0,
        workouts: dayWorkouts,
    });
}

return { month, year, days: calendarDays };
}

async markDayAsCompleted(userId: number, planId: number, dayId: number) {
    console.log(
        'Marking day as completed - user:',
        userId,
        'planId:',

```

		Годлевський О.О.			ІІЗ.КР.Б-121-25-ІІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

```

        planId,
        'dayId:',
        dayId,
    );
    await this.prisma.workoutDay.update({
        where: { id: dayId, workoutPlanId: planId, workoutPlan: { userId } },
        data: { isCompleted: true, completedAt: new Date() },
    });
}

async getWorkoutCalendar(userId: number, month: number, year: number) {
    console.log(
        'Service: Processing calendar request - userId:',
        userId,
        'month:',
        month,
        'year:',
        year,
    );

    const plans = await this.prisma.workoutPlan.findMany({
        where: { userId, status: 'IN_PROGRESS' },
        include: {
            WorkoutDay: {
                where: {
                    scheduledDate: {
                        gte: new Date(year, month - 1, 1),
                        lt: new Date(year, month, 1),
                    },
                },
                include: { exercises: { include: { exercise: true } } },
            },
        },
    });

    if (!plans.length) {
        const daysInMonth = new Date(year, month, 0).getDate();
        return {
            month,
            year,
            days: Array.from({ length: daysInMonth }, (_, i) => ({
                day: i + 1,
                hasWorkouts: false,
            }));
    }
}

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

        workouts: [],
    })),
};
}

const calendarDays = [];
const daysInMonth = new Date(year, month, 0).getDate();
for (let day = 1; day <= daysInMonth; day++) {
    const date = new Date(year, month - 1, day);
    const dayWorkouts = plans.flatMap((plan) =>
        plan.WorkoutDay.filter((wd) => {
            const wdDate = new Date(wd.scheduledDate || date);
            return (
                wdDate.getDate() === day &&
                wdDate.getMonth() === month - 1 &&
                wdDate.getFullYear() === year
            );
        })).map((wd) => ({
            planId: plan.id,
            planName: plan.name,
            dayId: wd.id,
            dayName: wd.name || `Day ${wd.dayNumber}`,
            isCompleted: wd.isCompleted ?? false,
            completedAt: wd.completedAt,
            exercisesCount: wd.exercises.length,
            completedExercisesCount: wd.exercises.filter(
                (e) => e.isCompleted ?? false,
            ).length,
        })),
    );

    calendarDays.push({
        day,
        hasWorkouts: dayWorkouts.length > 0,
        workouts: dayWorkouts,
    });
}

return { month, year, days: calendarDays };
}

async markExerciseAsCompleted(
    planId: number,

```

		Годлевський О.О.			ІІЗ.КР.Б-121-25-ІІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

```

        exerciseId: number,
        userId: number,
    ) {
        const plan = await this.prisma.workoutPlan.findUnique({
            where: { id: planId, userId },
            include: { WorkoutDay: { include: { exercises: true } } },
        });
        if (!plan) throw new NotFoundException('Plan not found');
        const exercise = plan.WorkoutDay.flatMap((d) => d.exercises).find(
            (e) => e.id === exerciseId,
        );
        if (!exercise) throw new NotFoundException('Exercise not found');
        return this.prisma.workoutPlanExercise.update({
            where: { id: exerciseId },
            data: { isCompleted: true, completedAt: new Date() },
        });
    }

    private calculateDayDate(startDate: Date, dayNumber: number): Date {
        const date = new Date(startDate);

        // Додаємо (dayNumber - 1) днів до початкової дати
        // Віднімаємо 1, бо перший день (dayNumber = 1) має бути в день startDate
        date.setDate(date.getDate() + (dayNumber - 1));

        return date;
    }

    async generatePlan(
        userId: number,
        dto: { daysPerWeek: number; fitnessGoal: string; startDate?: Date },
    ) {
        const user = await this.prisma.user.findUnique({ where: { id: userId }
    });
        if (!user) throw new NotFoundException('User not found');

        const plan = await this.prisma.workoutPlan.create({
            data: {
                userId,
                name: `Plan for ${user.goal || dto.fitnessGoal}`,
                fitnessGoal: user.goal || dto.fitnessGoal,
                startDate: dto.startDate || new Date(),
                status: 'IN_PROGRESS',
            },
        },

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

```

});

// Отримуємо вправи з включеним exerciseCategory
const exercises = await this.prisma.exercise.findMany({
  include: { exerciseCategory: true },
});

console.log('Total exercises fetched:', exercises.length);

// Фільтрація вправ залежно від цілі
let targetExercises = exercises.filter((e) => {
  const category = e.exerciseCategory?.name || '';
  const muscleGroup = e.muscleGroup?.toLowerCase() || '';

  return (
    (dto.fitnessGoal === 'LOSE_WEIGHT' &&
      (category === 'Cardio' ||
        ['ноги', 'грудь', 'спина'].some((mg) =>
          muscleGroup.includes(mg.toLowerCase()),
        ))) ||
    (dto.fitnessGoal === 'GAIN_MUSCLE' &&
      ['грудь', 'спина', 'ноги', 'плечі', 'руки'].some((mg) =>
        muscleGroup.includes(mg.toLowerCase()),
      )) ||
    (dto.fitnessGoal === 'STAY_ACTIVE' &&
      (category === 'Flexibility' ||
        ['ноги', 'корисний суглоб', 'руки'].some((mg) =>
          muscleGroup.includes(mg.toLowerCase()),
        )))
  );
});

console.log('Filtered exercises for goal:', targetExercises.length);

if (targetExercises.length === 0) {
  throw new Error('No exercises found for the specified fitness goal.');
```

```

}

const intensity = 1.0; // Базова інтенсивність
const exercisesPerDay = 3; // Кількість вправ на день
```

```

// Генерація днів тренувань з випадковим вибором вправ
const workoutDays = Array.from({ length: dto.daysPerWeek }, (_, i) => {
  const dayNumber = i + 1;
  const dayName = `Day ${dayNumber}`;

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

const scheduledDate = this.calculateDayDate(
  dto.startDate || new Date(),
  dayNumber,
);

// Випадковий вибір вправ для дня
const dayExercises = this.getRandomExercises(
  targetExercises,
  exercisesPerDay,
);

return {
  dayNumber,
  name: dayName,
  scheduledDate,
  exercises: {
    createMany: {
      data: dayExercises.map((e, idx) =>
        this.configureExerciseParams(e, idx, dto.fitnessGoal,
intensity),
      ),
    },
  },
};
});

for (const day of workoutDays) {
  const createdDay = await this.prisma.workoutDay.create({
    data: {
      workoutPlanId: plan.id,
      dayNumber: day.dayNumber,
      name: day.name,
      scheduledDate: day.scheduledDate,
    },
  });
  await this.prisma.workoutPlanExercise.createMany({
    data: day.exercises.createMany.data.map((ex) => ({
      ...ex,
      workoutPlanId: plan.id,
      workoutDayId: createdDay.id,
    })),
  });
}

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

```

return this.prisma.workoutPlan.findUnique({
  where: { id: plan.id },
  include: {
    WorkoutDay: { include: { exercises: { include: { exercise: true } } } }
  },
},
);
});
}

private configureExerciseParams(
  exercise: any,
  index: number,
  goal: string,
  intensity: number,
) {
  if (!exercise || !exercise.id) {
    console.error('Invalid exercise object:', exercise);
    throw new Error('Invalid exercise data');
  }

  // Базові параметри
  const result = {
    exerciseId: exercise.id,
    order: index + 1,
    reps: null,
    duration: null,
    restDuration: null,
  };

  // Налаштовуємо параметри залежно від типу вправи
  if (exercise.exerciseCategory?.name === 'Cardio') {
    // Для кардіо встановлюємо тривалість
    result.duration =
      goal === 'LOSE_WEIGHT'
        ? Math.round(intensity * 60) // Більше для схуднення
        : Math.round(intensity * 30); // Менше для інших цілей
  } else {
    // Для силових вправ встановлюємо повторення
    result.reps = this.calculateReps(intensity, exercise.muscleGroup);
  }

  // Встановлюємо час відпочинку

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		88

```

        result.restDuration = this.calculateRestDuration(intensity, goal);

        return result;
    }

    private calculateReps(intensity: number, muscleGroup: string): number {
        // Базова кількість повторень
        let baseReps = 10;

        // Модифікуємо залежно від групи м'язів
        if (['chest', 'back', 'legs'].includes(muscleGroup)) {
            baseReps = 8; // Менше повторень для великих груп м'язів
        } else if (['arms', 'shoulders'].includes(muscleGroup)) {
            baseReps = 12; // Більше повторень для малих груп м'язів
        } else if (muscleGroup === 'core') {
            baseReps = 15; // Найбільше для кору
        }

        // Модифікуємо залежно від інтенсивності
        return Math.round(baseReps * intensity);
    }

    private calculateRestDuration(intensity: number, goal: string): number {
        // Базовий час відпочинку в секундах
        let baseRest = 60;

        // Модифікуємо залежно від цілі
        if (goal === 'LOSE_WEIGHT') {
            baseRest = 45; // Менше відпочинку для схуднення
        } else if (goal === 'GAIN_MUSCLE') {
            baseRest = 90; // Більше відпочинку для набору маси
        }

        // Модифікуємо залежно від інтенсивності
        return Math.round(baseRest * intensity);
    }

    private getRandomExercises(exercises: any[], count: number): any[] {
        if (!exercises || exercises.length === 0) {
            console.warn('Немає доступних вправ для вибору');
            return [];
        }
    }

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

```

    // Якщо вправ менше, ніж потрібно, повертаємо всі доступні
    if (exercises.length <= count) {
        console.log(`Доступно лише ${exercises.length} вправ, повертаємо всі`);
        return [...exercises];
    }

    // Перемішуємо масив вправ
    const shuffled = [...exercises].sort(() => 0.5 - Math.random());

    // Повертаємо потрібну кількість вправ
    return shuffled.slice(0, count);
}
}}

```

Додаток Б

Лістинг generatePlanDto.ts

```

import {
    IsEnum,
    IsNotEmpty,
    IsNumber,
    IsOptional,
    IsString,
    Min,
} from 'class-validator';

export enum GoalType {
    LOSE_WEIGHT = 'LOSE_WEIGHT',
    GAIN_MUSCLE = 'GAIN_MUSCLE',
    STAY_ACTIVE = 'STAY_ACTIVE',
    // FLEXIBILITY = 'FLEXIBILITY',
}

export enum ExerciseCategory {
    STRENGTH = 'STRENGTH',
    CARDIO = 'CARDIO',
    FLEXIBILITY = 'FLEXIBILITY',
    BALANCE = 'BALANCE',
    ARMS = 'ARMS',
}

export enum DifficultyLevel {
    BEGINNER = 'BEGINNER',

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ПЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		90

```

    INTERMEDIATE = 'INTERMEDIATE',
    ADVANCED = 'ADVANCED',
}

export enum PlanStatus {
    IN_PROGRESS = 'IN_PROGRESS',
    COMPLETED = 'COMPLETED',
    CANCELLED = 'CANCELLED',
}

export class CreatePlanDto {
    @IsString()
    @IsNotEmpty()
    name: string;

    @IsString()
    @IsNotEmpty()
    description: string;

    @IsOptional()
    @IsEnum(GoalType)
    fitnessGoal?: GoalType = GoalType.LOSE_WEIGHT;
}

export class UpdatePlanDto {
    @IsString()
    @IsOptional()
    name?: string;

    @IsString()
    @IsOptional()
    description?: string;

    @IsEnum(GoalType)
    @IsOptional()
    fitnessGoal?: GoalType;

    @IsOptional()
    @IsEnum(PlanStatus)
    status?: PlanStatus;
}

export class AddExerciseToPlanDto {

```

		Годлевський О.О.			ІІЗ.КР.Б-121-25-ІІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		91

```

        @IsNumber()
        @Min(1)
        exerciseId: number;

        @IsNumber()
        @Min(0)
        order: number;

        @IsNumber()
        @IsOptional()
        @Min(0)
        reps?: number;

        @IsNumber()
        @IsOptional()
        @Min(0)
        duration?: number;

        @IsNumber()
        @IsOptional()
        @Min(0)
        restDuration?: number;
    }

    export class UpdateExerciseInPlanDto {
        @IsNumber()
        @IsOptional()
        @Min(0)
        order?: number;

        @IsNumber()
        @IsOptional()
        @Min(0)
        reps?: number;

        @IsNumber()
        @IsOptional()
        @Min(0)
        duration?: number;

        @IsNumber()
        @IsOptional()
        @Min(0)

```

		Годлевський О.О.			ІІЗ.КР.Б-121-25-ІІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		92

```

        restDuration?: number;
    }

    export class CreateExerciseDto {
        @IsString()
        @IsNotEmpty()
        name: string;

        @IsString()
        @IsNotEmpty()
        muscleGroup: string;

        @IsNumber()
        @IsOptional()
        exerciseCategoryId?: number;

        @IsString()
        @IsOptional()
        difficulty?: string;

        @IsString()
        @IsOptional()
        videoUrl?: string;

        @IsString()
        @IsOptional()
        description?: string;

        @IsString()
        @IsOptional()
        equipment?: string;

        @IsNumber()
        @IsOptional()
        duration?: number;

        @IsNumber()
        @IsOptional()
        reps?: number;

        @IsNumber()
        @IsOptional()
        restDuration?: number;
    }

```

		Годлевський О.О.			ІІЗ.КР.Б-121-25-ІІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		93

```

@IsNumber()
@IsOptional()
calories?: number;

@IsString()
@IsOptional()
intensity?: string;

@IsString()
@IsOptional()
imageUrl?: string;
}

```

		Годлевський О.О.			ІПЗ.КР.Б-121-25-ІЗ	
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		94