МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:
Завідувач кафедри
комп'ютерних технологій та
економічної кібернетики
д. е. н., проф. П. М. Грицюк
_____
«____» _____ 20__ р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня «магістр»
за освітньо-професійною програмою
«Інформаційні технології в бізнесі»
спеціальності 126 «Інформаційні системи та технології»
на тему: «**Інформаційна система для організації та управління тренувальним процесом в тренажерному залі**»

Виконала:
здобувачка вищої освіти 2 курсу,
групи ІТБ-61м
Чаплик Олеся Андріївна
Керівник:
Доцент, кандидат технічних наук,
Барановський С. В.
Рецензент:
к.е.н., доц. Волошин В.С.

Рівне – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет водного господарства та природокористування
ННІ кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Освітньо-кваліфікаційний рівень – магістр
Освітньо-професійна програма «Інформаційні технології в бізнесі»
Спеціальність 126 «Інформаційні системи і технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних технологій
та економічної кібернетики
_____ д.е.н., проф. П.М. Грицюк
«_____» _____ 20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачу *Чаплик Олесі Андріївні*
(прізвище, ім'я, по-батькові)

1. Тема роботи *Інформаційна система для організації та управління тренувальним процесом в тренажерному залі*
керівник роботи *канд. техн. наук, доцент Барановський С.В.*
(прізвище, ім'я, по-батькові, науковий ступінь, вчене звання)
затверджена наказом по університету від "____ " _____ 202__ року №_____

2. Термін здачі студентом закінченої роботи _____

3. Вихідні дані до роботи *Актуальною проблемою сучасного етапу розвитку спортивної інфраструктури є необхідність підвищення ефективності організації тренувального процесу та управління діяльністю тренажерних залів на основі впровадження інформаційних систем і цифрових технологій. Завданням кваліфікаційної роботи є дослідження можливостей застосування веб-орієнтованих інформаційних систем для автоматизації процесів планування, обліку та контролю тренувань, а також розроблення програмного забезпечення, що забезпечує ефективну взаємодію між клієнтами, тренерами та адміністрацією тренажерного залу*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Вступ
1. Аналіз сучасного стану та розвитку інформаційних систем у сфері фітнесу та спорту.
2. Дослідження веб-технологій та інформаційних систем управління тренувальним процесом.
3. Загальні принципи проєктування інформаційних систем для тренажерних залів.
4. Технологічні засоби створення веб-додатків для автоматизації тренувального процесу.
5. Особливості реалізації функціональних можливостей системи управління тренуваннями.
6. Організація зберігання та обробки даних у базі даних тренажерного залу.
7. Програмна реалізація та тестування інформаційної системи.

Висновки

5. Перелік графічного матеріалу _____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання отримав
1	*Барановський С.В., доцент*		
2	*Барановський С.В., доцент*		
3	*Барановський С.В., доцент*		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Терміни виконання етапів роботи	Примітка
1.	*Загальна характеристика та аналіз предметної області магістерської роботи*	*до 20.09.2025*	
2.	*Аналіз сучасних інформаційних технологій та програмних рішень у сфері управління тренувальним процесом*	*до 01.10.2025*	
3.	*Формування вимог до інформаційної системи та проєктування її архітектури*	*до 10.10.2025*	
4.	*Розроблення програмної частини інформаційної системи управління*	*до 20.10.2025*	
5.	*Реалізація роботи з базою даних та інтеграція інтелектуальних компонентів системи*	*до 01.11.2025*	
6.	*Тестування, оцінка ефективності та аналіз результатів роботи системи*	*до 30.11.2025*	
7.	*Оформлення та представлення до захисту магістерської роботи*	*до 15.12.2025*	

Студент(ка) _____ _____
(підпис) (прізвище та ініціали)

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

РЕФЕРАТ

Кваліфікаційна робота магістра: **100 с., 17 рис., 10 табл., 10 літературних джерел.**

Актуальність теми магістерської роботи полягає у зростаючій потребі тренажерних залів у сучасних цифрових рішеннях, які забезпечують автоматизацію тренувального процесу, персоналізацію роботи з клієнтами та оптимізацію діяльності адміністрації. Інформаційні системи у фітнес-індустрії стають важливим інструментом для організації тренувань,

Об'єктом дослідження магістерської роботи є процес функціонування веб-орієнтованої інформаційної системи, призначеної для управління тренувальним процесом у тренажерному залі.

Предметом дослідження є методи, технології та архітектурні підходи до розроблення веб-додатків, включно з використанням Next.js, React, MongoDB, Mongoose та алгоритмів штучного інтелекту для персоналізації тренувальних рекомендацій.

Метою магістерської роботи є створення інформаційної системи PowerGym, яка забезпечує автоматизацію роботи тренажерного залу, підтримує управління користувачами та тренерами, формує персональні тренувальні плани за допомогою AI-технологій та оптимізує взаємодію між адміністрацією й клієнтами.

У магістерській роботі виконано проєктування та розроблення веб-додатка PowerGym, що включає модулі авторизації, управління тренуваннями, систему мотивації, запис на заняття і адміністративну панель. Результати підтверджують ефективність системи та її доцільність для впровадження в діяльність спортивних закладів.

Ключові слова: інформаційна система, тренажерний зал, веб-додаток, Next.js, MongoDB, Mongoose, штучний інтелект, AI-тренер, тренування, автоматизація, адміністрування.

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку цифрових технологій та цифрової трансформації різних сфер суспільної діяльності особливого значення набуває інформатизація фізичної культури та спортивної індустрії. Тренажерні зали, фітнес-центри та спортивні клуби поступово перетворюються на багатофункціональні простори, у яких цифрові інструменти відіграють ключову роль в організації тренувального процесу, взаємодії з клієнтами, управлінні персоналом та оптимізації внутрішніх бізнес-процесів. Сучасний користувач очікує не лише якісних тренажерів і професійних тренерів, але й ефективної цифрової взаємодії: онлайн-розкладу занять, персоналізованих рекомендацій, аналітики власних досягнень, можливості отримувати консультації дистанційно, а також доступу до тренувальної програми у будь-який час. Саме тому впровадження інформаційних систем нового покоління стає стратегічною необхідністю для спортивних закладів, які прагнуть підвищити конкурентоспроможність і забезпечити високий рівень сервісу.

Паралельно з цим на ринку спостерігається тенденція до активного використання штучного інтелекту в побудові персоналізованих тренувальних програм. Користувачі, що займаються в тренажерному залі, дедалі частіше віддають перевагу технологічним рішенням, здатним проаналізувати їхній стан, фізичні параметри, рівень підготовки, цілі та інші індивідуальні характеристики, а потім автоматично згенерувати персональний план розвитку. Такі можливості досягаються завдяки поєднанню сучасних технологій веб-програмування, хмарної обробки даних, нейронних мереж та інтеграції з API систем штучного інтелекту. У цьому контексті особливу увагу привертає необхідність створення комплексної веб-системи, яка поєднує функції керування тренувальним процесом, взаємодію з користувачами, інструменти аналітики та модуль рекомендацій на основі AI.

Незважаючи на наявність окремих продуктів для планування тренувань чи управління абонементами, на ринку фактично відсутні інтегровані рішення, які б одночасно забезпечували можливість перегляду розкладу в режимі реального часу, аналізу завантаженості залу, оцінювання тренувань, системи гейміфікації, особистого кабінету з досягненнями користувача, а також повноцінного AI-тренера, що формує індивідуальні рекомендації. Більшість сучасних фітнес-систем або мають вузьку спеціалізацію, або не передбачають глибокої інтеграції даних, що зменшує ефективність їх використання з боку клієнтів та адміністрації. Це створює актуальну потребу в розробці нового типу інформаційної системи, здатної забезпечити комплексний підхід до оцифрування тренувального процесу.

Водночас цифрова трансформація тренажерних залів передбачає вирішення низки технічних завдань, пов’язаних із проектуванням архітектури веб-додатка, організацією надійного сховища даних, забезпеченням безпеки користувацької інформації, формуванням швидкого інтерфейсу, а також інтеграцією зовнішніх сервісів для генерації рекомендаційних моделей. Важливим аспектом виступає необхідність створення системи, яка одночасно буде масштабованою, адаптивною, швидкою у роботі та зручною для кінцевого користувача. Ураховуючи сучасні тенденції, доцільним є використання інструментів Node.js та фреймворку Next.js, які дозволяють реалізувати високопродуктивний інтерфейс із можливістю серверного рендерингу та побудови API-маршрутів. Для зберігання даних оптимальним рішенням є використання MongoDB у поєднанні з Mongoose, що забезпечує гнучкість, простоту масштабування та ефективну роботу з документно-орієнтованою структурою даних. Застосування Tailwind CSS надає можливість створити адаптивний і сучасний інтерфейс користувача, а інтеграція OpenAI відкриває шлях до формування персоналізованих тренувальних рекомендацій, які значно підвищують якість цифрового сервісу.

Метою магістерської роботи є розроблення комплексної веб-орієнтованої інформаційної системи для організації та управління тренувальним процесом у тренажерному залі, яка дозволяє користувачам взаємодіяти з тренуваннями, отримувати рекомендації, відстежувати власні результати, а адміністрації – ефективно управляти контентом, розкладом та відвідуваністю. У рамках досягнення зазначеної мети необхідно розв'язати низку наукових і практичних завдань, спрямованих на дослідження сучасних підходів до цифровізації фітнес-індустрії, проектування архітектури веб-системи, побудову структури бази даних, визначення ключових функцій користувацького інтерфейсу, реалізацію серверної логіки, інтеграцію AI-модуля, а також тестування створеного програмного продукту.

Об'єктом дослідження є процес організації тренувальної діяльності в тренажерному залі з використанням інформаційних технологій. Предметом дослідження виступають методи, алгоритми та програмні засоби створення веб-орієнтованої системи для управління тренувальним процесом та взаємодії з клієнтами.

Методологічну основу роботи становлять загальнонаукові методи аналізу та синтезу, методи системного підходу, інструменти моделювання бізнес-процесів (IDEF0, UML), методи проєктування веб-архітектури, а також практичні інструменти веб-розробки (Next.js, Node.js, MongoDB, Mongoose, Tailwind CSS) та сучасні алгоритми генерації рекомендацій на основі API штучного інтелекту.

Практичне значення роботи полягає у створенні програмного забезпечення, яке може бути впроваджене в діяльність будь-якого тренажерного залу чи фітнес-центру з метою цифровізації управління, покращення взаємодії з клієнтами, оптимізації тренувального процесу та підвищення ефективності організаційних процесів. Розроблена система має потенціал до реального застосування, масштабування та подальшого розширення функціональних можливостей, включаючи мобільний додаток,

поглиблену аналітику та автоматизоване формування планів тренувань на основі використання машинного навчання.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ СТВОРЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ ДЛЯ ОРГАНІЗАЦІЇ ТРЕНУВАЛЬНОГО ПРОЦЕСУ

1.1. Сучасні підходи до організації тренувального процесу в тренажерних залах

Організація тренувального процесу у тренажерних залах у XXI столітті зазнає суттєвих трансформацій, зумовлених розвитком спортивної науки, зміною поведінкових моделей користувачів та впливом цифрових технологій на методи побудови фізичної активності. Якщо раніше тренувальний процес будувався переважно на основі досвіду тренера та загальних принципів фізіології, то сьогодні ключову роль відіграють персоналізація, аналітика, адаптивні програми та доступ до структурованої інформації в режимі реального часу. Тренажерні зали перестають бути лише простором із тренажерами та перетворюються на комплексні центри фізичного розвитку, де взаємодія між тренером, клієнтом і адміністрацією забезпечується за допомогою сучасних інформаційних інструментів та методологій.

Одним із ключових трендів сучасної організації тренувального процесу є переорієнтація з універсальних програм на індивідуальні інтенсивні тренування, що враховують стан здоров’я, рівень фізичної підготовки, анамнез травм, мотивацію та конкретні цілі клієнта. Індивідуальний підхід забезпечує не лише кращі результати, але й підвищує рівень безпеки, знижує ризик перенавантаження чи травм та підтримує мотивацію людини в довгостроковій перспективі. У традиційній моделі тренер виконує роль головного координатора процесу, однак за умови великої кількості клієнтів він фізично не здатний контролювати всі аспекти активності. Саме тому на перший план виходять автоматизовані системи, здатні відстежувати прогрес, нагадувати про графік, прогнозувати ефективність та пропонувати рекомендації.

Другим важливим аспектом сучасного тренувального процесу є зростання ролі аналітики. Відвідувачі тренажерних залів дедалі частіше

прагнуть отримати доступ до конкретних даних про стан свого організму, динаміку навантажень, інтенсивність тренувань, кількість спалених калорій, частоту відвідувань та інші показники. Дані стають важливим інструментом управління власним прогресом, а тренажерні зали, які впроваджують цифрові рішення, можуть забезпечити більш якісну взаємодію з клієнтами. Збір і аналіз даних дають змогу клієнту отримати більш точну інформацію про ефективність роботи, а тренеру — коригувати програму на основі об’єктивних показників. Це створює новий рівень цілей і очікувань, де тренувальний процес перестає бути інтуїтивним і стає науково обґрунтованим.

Поширення мобільних пристроїв і носимих технологій таких як фітнес-браслети, смарт-годинники, датчики пульсу та інші пристрої створило новий рівень інтеграції між фізичною активністю та цифровим середовищем. Користувач може постійно контролювати власні біометричні показники, а тренажерні зали — синхронізувати цю інформацію з тренувальною системою. Це дозволяє автоматизувати процес збору даних, значно знижуючи людський фактор та підвищуючи точність аналітичної інформації. Така інтеграція створює передумови для формування справді персоналізованих програм, заснованих на реальних параметрах фізичної активності, що робить тренування більш ефективними.

Ще одним важливим чинником є розвиток дистанційного тренінгу та онлайн-фітнесу. Після пандемії COVID-19 значна частина клієнтів бажає мати можливість тренуватися не лише в залі, а й удома, за допомогою онлайн-програм чи індивідуальних рекомендацій. Навіть після повернення до звичного режиму життя тренажерні зали активно впроваджують гібридні моделі взаємодії, де частина тренувального процесу залишається в цифровому середовищі. Це вимагає створення веб-систем, здатних формувати онлайн-розклад, надавати відеоматеріали, відстежувати статистику, а також підтримувати функціонал, пов’язаний із гейміфікацією, нагородами, рейтингами та іншими цифровими механізмами мотивації.

Організація тренувального процесу також включає питання управління навантаженням у тренажерному залі. Адміністрація повинна забезпечувати оптимальне планування групових занять, контроль кількості відвідувачів у пікові години, розподіл ресурсів, створення комфортного середовища для тренувань та ефективну логістику простору. Використання інформаційних систем дозволяє точно прогнозувати завантаженість залу в реальному часі, формувати гнучкий розклад, швидко реагувати на зміни та оптимізувати роботу персоналу. Для клієнтів це забезпечує можливість планувати свої тренування без черг і перевантаження тренажерів.

Окреме місце в сучасному підході до організації тренувального процесу займає гейміфікація, яка стає одним із ключових інструментів мотивації. Системи балів, досягнень, рейтингів та внутрішніх нагород стимулюють користувачів підтримувати регулярний тренувальний режим, ставити нові цілі та змагатися між собою. Гейміфікація не лише підвищує залученість клієнтів, але й формує більш глибоку взаємодію з цифровою екосистемою тренажерного залу. Для цього необхідні програмні рішення, здатні обробляти статистику користувача, нараховувати бали, формувати логіку досягнень та зберігати відповідні дані в базі.

Сукупність цих факторів свідчить про те, що тренувальний процес перестає бути статичним явищем і перетворюється на комплексну, багаторівневу систему, у якій кожен елемент — від тренера до адміністратора і від клієнта до обладнання — має свою роль у створенні ефективного середовища для розвитку фізичної активності. Сучасний тренажерний зал повинен не лише пропонувати якісні послуги, але й забезпечувати повноцінну цифрову взаємодію, здатну підвищити ефективність тренувань та задоволеність користувачів. У цьому контексті інформаційні системи виступають ключовим інструментом, що дозволяє забезпечити потреби спортивної аудиторії, удосконалити процес управління тренуваннями та підвищити конкурентоспроможність закладу на ринку фітнес-послуг.

1.2. Роль цифрових технологій у сфері фітнесу

Цифрові технології стали невід'ємною частиною сучасної фітнес-індустрії, істотно змінюючи способи організації тренувального процесу, взаємодії з клієнтами та управління діяльністю спортивних закладів. Протягом останнього десятиліття спостерігається стійка тенденція до глибокої інтеграції інформаційних систем у всі аспекти роботи тренажерних залів — від реєстрації відвідувачів і продажу абонементів до формування персональних тренувальних програм та аналітики навантажень. Ці зміни зумовлені появою потужних програмних рішень, широким проникненням мобільних пристроїв, розвитком хмарних технологій, технологій обробки великих даних, а також стрімким поширенням штучного інтелекту як інструменту підтримки прийняття рішень. Під впливом цих факторів цифровізація стала не просто допоміжним елементом, а фундаментальною умовою ефективного функціонування сучасного тренажерного залу.

Одним із ключових аспектів цифрової трансформації фітнес-сфери є автоматизація організаційних процесів, що раніше виконувалися вручну. Багато тренажерних залів у минулому використовували паперові журнали відвідувань, статичні розклади занять та усні рекомендації тренерів. У сучасних умовах такі підходи є неефективними, оскільки збільшення кількості клієнтів, різноманіття видів тренувань, потреба в динамічних змінах графіків та високий рівень конкуренції вимагають швидкого доступу до актуальної інформації та автоматизації рутинних процесів. Цифрові системи забезпечують можливість централізованого управління розкладом, швидкого внесення змін, автоматичного інформування клієнтів, обліку активності та ведення статистики, що значно знижує адміністративні витрати та мінімізує людський фактор.

Не менш важливим напрямом цифровізації є вдосконалення взаємодії між клієнтом і тренажерним залом. Сучасні користувачі очікують зручності та доступності сервісів у будь-який час і з будь-якого пристрою. Мобільні та веб-застосунки стали стандартом для організації тренувального процесу:

через них клієнт може переглядати розклад, записуватися на заняття, отримувати сповіщення, контролювати баланс абонемента, переглядати власні результати, аналізувати прогрес та взаємодіяти з тренерами. Усе це сприяє підвищенню рівня задоволеності користувачів, їх лояльності, а також залученості до діяльності фітнес-центру. Крім того, цифрові рішення дозволяють тренерам оперативно надавати зворотний зв’язок, аналізувати дані клієнтів та оптимізувати тренувальні програми без необхідності постійної фізичної присутності.

Цифрові технології також сприяють розвитку систем мотивації та гейміфікації, що стають надзвичайно ефективним інструментом підтримки тренувальної активності. Завдяки використанню цифрових платформ з’явилася можливість впроваджувати бали за відвідування, рівні досягнень, рейтинги користувачів, внутрішні нагороди та інші механізми, які стимулюють людей до регулярних тренувань. Гейміфікація допомагає подолати психологічні бар’єри, пов’язані з втомою або втратою мотивації, оскільки користувач отримує від тренувального процесу не лише фізичну користь, але й емоційне задоволення. Водночас цифрові системи забезпечують постійний доступ до власної статистики — кількості виконаних тренувань, прогресу у вправах, витрачених калорій, часу активності, що створює ефект самоконтролю і підсилює внутрішню мотивацію.

Окремо слід відзначити роль цифрових технологій у створенні інтелектуальних рекомендаційних систем. Штучний інтелект дедалі частіше використовується у фітнес-індустрії для формування персональних тренувальних програм, вибору оптимальних видів активності, моніторингу фізичних показників та аналізу ефективності тренувань. Інформаційні системи з інтегрованими AI-тренерами здатні враховувати значний обсяг параметрів — вагу, зріст, фізичний стан, мету тренувань, рівень активності, обмеження щодо здоров’я, а також обробляти статистику попередніх занять. Це дозволяє створити індивідуальний план тренувань, який є більш точним і науково обґрунтованим, ніж стандартні, універсальні програми.

Використання штучного інтелекту в цьому контексті забезпечує адаптивність тренувального процесу: система може коригувати план залежно від прогресу користувача, рівня складності, часу відновлення та інших факторів.

Цифрові технології також відіграють ключову роль у забезпеченні прозорості та ефективності управління фітнес-бізнесом. Завдяки системам обліку відвідуваності адміністрація може аналізувати пікові години навантаження, прогнозувати завантаженість тренажерного залу, оптимізувати кількість тренерів на зміні, формувати графіки занять та вибудовувати стратегії щодо збільшення прибутковості. Аналітичні панелі та звіти дають змогу керівництву оперативно приймати управлінські рішення, покращувати сервіс, впроваджувати нові тренувальні програми або змінювати маркетингові підходи. У поєднанні з CRM-системами цифрові рішення дозволяють працювати з клієнтською базою, сегментувати користувачів, вести історію взаємодій, аналізувати причини відтоку та створювати персональні пропозиції для кожного клієнта.

Суттєвий вплив цифрові технології мають також на безпеку даних та якість електронної комунікації. Оскільки сучасні системи зберігають персональні дані клієнтів — від контактної інформації до медичних рекомендацій та фінансових операцій — важливим завданням є забезпечення надійного шифрування, обмеження доступу, застосування токен-автентифікації та регулярний моніторинг безпеки. Цифрові рішення мають забезпечити безпечну взаємодію з інформацією, унеможливити витік даних і гарантувати відповідність сучасним стандартам кібербезпеки.

Загалом роль цифрових технологій у сфері фітнесу є трансформаційною та багатогранною. Вони забезпечують фундамент для побудови ефективного, персоналізованого та науково обґрунтованого тренувального процесу; створюють умови для якісної взаємодії між тренажерним залом і клієнтом; підвищують рівень мотивації та залученості користувачів; дозволяють адміністрації оптимізувати бізнес-процеси; забезпечують аналітику, автоматизацію та контроль на всіх етапах роботи

спортивного закладу. Сучасні тренажерні зали, які активно впроваджують цифрові рішення, отримують значну конкурентну перевагу, оскільки пропонують клієнтам більш зручний, технологічний і результативний тренувальний досвід.

1.3. Інформаційні системи у спортивній індустрії: класифікація та призначення

Інформаційні системи у спортивній індустрії відіграють ключову роль у формуванні сучасної цифрової екосистеми, що охоплює як професійний спорт, так і сферу масового фітнесу. Протягом останніх років відбувається стрімке зростання цифрових рішень, спрямованих на оптимізацію тренувального процесу, підвищення ефективності взаємодії з клієнтами, автоматизацію управління спортивними закладами та розширення доступу до інформації про фізичну активність. Інформаційні системи перестали бути суто допоміжними інструментами й перетворилися на базову інфраструктурну складову, без якої сучасний тренажерний зал, фітнес-центр або спортивний клуб не може забезпечити необхідний рівень сервісу, аналітичної підтримки та управлінської ефективності.

У широкому розумінні інформаційні системи у спортивній сфері можна визначити як організований комплекс апаратних, програмних та інформаційних засобів, що забезпечують збір, оброблення, аналіз, зберігання та передачу інформації, пов’язаної з тренувальним процесом, станом клієнтів, роботою персоналу, фінансовою діяльністю та іншими аспектами функціонування спортивного закладу. Сутність таких систем полягає в автоматизації ключових бізнес-процесів, підвищенні точності даних, мінімізації людського фактору, забезпеченні інтерактивної взаємодії та створенні умов для прийняття оптимальних управлінських рішень.

Залежно від функціонального призначення інформаційні системи у спортивній індустрії можна умовно класифікувати на кілька основних категорій. Однією з найпоширеніших груп є системи управління фітнес-центрами та тренажерними залами (Gym Management Systems). Такі системи

забезпечують комплексне керування всіма процесами закладу: реєстрацією та обслуговуванням клієнтів, продажем абонементів, управлінням розкладом, контролем відвідуваності, плануванням роботи персоналу, веденням фінансової звітності та організацією внутрішньої логістики. До їх переваг належать інтеграція різних бізнес-функцій у єдину платформу, можливість централізованого управління, доступність аналітичних даних у реальному часі та зниження операційних витрат.

Другою групою є інформаційні системи, спрямовані на підтримку тренувального процесу та взаємодію з клієнтом. Це можуть бути мобільні додатки, веб-платформи або інтегровані програмно-апаратні комплекси, що дозволяють клієнтам переглядати тренувальні програми, контролювати графік занять, аналізувати власні результати, відстежувати фізичну активність та отримувати рекомендації. Багато сучасних систем включають функції періодичного відстеження стану користувача, ведення персонального щоденника, гейміфікації та тісної інтеграції з носимими пристроями. Такі рішення роблять тренувальний процес більш прозорим, відстежуваним та мотивуючим для кінцевого користувача, створюючи індивідуальний цифровий профіль фізичної активності.

До окремої категорії належать рекомендаційні системи та платформи, що використовують алгоритми штучного інтелекту для формування індивідуальних тренувальних планів, нутриційних порад, аналізу техніки виконання вправ або прогнозування показників фізичної форми. Введення AI у спортивну індустрію дозволяє перейти від статичних тренувальних програм до адаптивних моделей, які змінюють інтенсивність, обсяг навантаження та структуру тренувань залежно від прогресу користувача. Такі системи здатні аналізувати значний обсяг даних, включаючи частоту пульсу, темп, тривалість тренувань, індекс маси тіла, попередні результати, рівень відновлення організму та інші показники. Використання штучного інтелекту створює нові можливості у сфері персоналізації тренувального процесу та робить тренування більш ефективними для широкої аудиторії.

Ще одним важливим типом інформаційних систем є аналітичні та моніторингові платформи, які дозволяють адміністрації тренажерних залів здійснювати глибоку оцінку статистики роботи закладу, прогнозувати зміни у навантаженні та приймати стратегічні управлінські рішення на основі даних. Такі системи часто включають модулі прогнозування відвідуваності, аналітику продажів, аналіз популярності тренувальних програм, оцінку ефективності тренерів, сегментацію клієнтів та побудову індивідуальних маркетингових стратегій. Аналітичні інструменти стають необхідністю для розвитку сучасного бізнесу у сфері фітнесу, оскільки дозволяють оперативно реагувати на зміни попиту, збалансувати навантаження між тренерами, оптимізувати фінансові операції та підвищити конкурентоспроможність закладу.

Суттєву роль відіграють і соціальні та комунікаційні інформаційні системи, що забезпечують взаємодію між клієнтами, тренерами та закладом. Сюди належать чат-платформи, внутрішні месенджери, системи для проведення онлайн-консультацій та спільнотні платформи для обміну досвідом. Такі рішення формують цифрову інфраструктуру взаємодії та сприяють створенню активної спортивної спільноти, що стимулює клієнтів до регулярності тренувань та підвищує рівень залученості.

Для тренажерних залів важливими також є системи контролю доступу та безпеки, які дозволяють автоматизувати вхід користувачів, збирати статистику відвідувань, відстежувати діяльність у залі та гарантувати відповідний рівень безпеки. Це можуть бути RFID-карти, QR-коди, біометричні ідентифікатори або інші технології, інтегровані з інформаційною системою закладу.

Інформаційні системи в спортивній індустрії різняться за функціональним призначенням, але їх об’єднує спільна мета — підвищення ефективності тренувального процесу, оптимізація управління спортивним закладом та створення більш комфортного й технологічно розвиненого середовища для користувачів. Усі вони формують багаторівневу цифрову

інфраструктуру, що забезпечує зручність, точність і прозорість процесів, сприяє підвищенню задоволеності клієнтів та стимулює розвиток фітнес-бізнесу.

1.4. Аналіз існуючих веб-платформ для фітнес-центрів.

Ринок веб-платформ для фітнес-центрів значно розширився за останні роки, що стало наслідком глобальної цифровізації та поступового переходу тренажерних залів до систем автоматизованого управління. У контексті підвищення конкуренції між фітнес-клубами зростає потреба у впровадженні комплексних цифрових рішень, здатних не лише забезпечити базовий функціонал — такий як розклад занять чи облік абонементів, — а й інтегрувати сучасні інструменти аналізу, мотивації, персоналізації та взаємодії з користувачами. Незважаючи на велику кількість доступних платформ, кожна з них має свої переваги та недоліки, а також обмеження, що заважають створити уніфіковану універсальну систему, яка могла б повністю задовольнити потреби тренажерного залу. Аналіз актуальних рішень дозволяє визначити ключові тенденції у розробці фітнес-платформ та обґрунтувати необхідність створення власної інформаційної системи PowerGym як комплексного інноваційного продукту.

Однією з найпопулярніших платформ є Mindbody, яка широко використовується у світі й слугує своєрідним еталоном для управлінських систем у сфері фітнесу. Mindbody пропонує широкий спектр можливостей, включаючи запис на заняття, продаж абонементів, управління персоналом, аналітику доходів та інтеграцію з мобільними пристроями. Однак система орієнтована переважно на великий бізнес і має складну структуру налаштувань, що створює певний бар’єр для невеликих тренажерних залів, які не потребують надмірного функціоналу. Крім того, Mindbody є закритою платформою, тому можливості кастомізації обмежені, а будь-яке розширення можливе лише через додаткові платні модулі. Це знижує гнучкість і

адаптивність платформи, що є суттєвим недоліком у контексті розробки індивідуальних рішень.

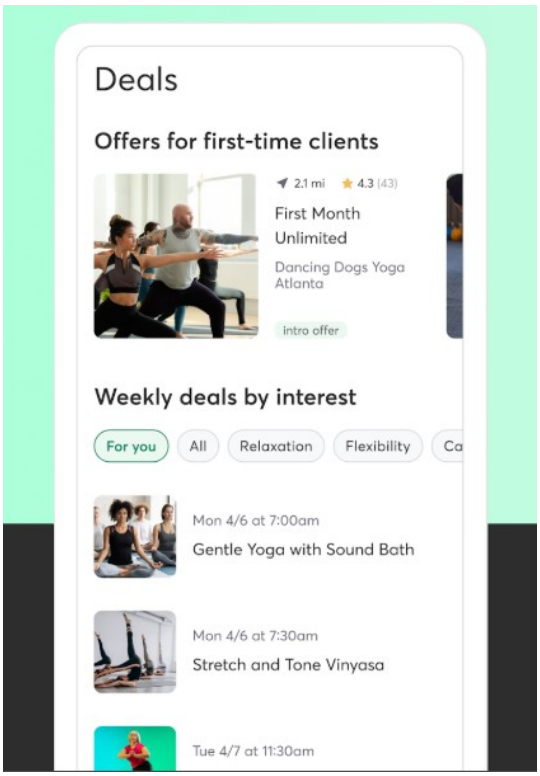

Рис. 1.4.1. Інтерфейс платформи «Mindbody»

Іншим поширеним інструментом є Virtuagym, який поєднує функціонал CRM-системи, мобільного додатка та платформи для формування персональних тренувань. Virtuagym має розвинений модуль тренувальних програм, дозволяє клієнтам переглядати відеоінструкції, вести щоденники занять та використовувати функції автоматичного підрахунку калорій. Водночас система має суттєві недоліки: закритість архітектури, високі ліцензійні витрати, залежність від серверів розробника та обмежені можливості щодо зміни інтерфейсу відповідно до вимог конкретного тренажерного залу. Крім того, Virtuagym не враховує всі аспекти роботи фітнес-центру, зокрема управління завантаженістю зали в реальному часі або повноцінну систему гейміфікації.

Рис. 1.4.2. платформа «Virtuagym»

На регіональному ринку популярністю користується платформа FitCloud, орієнтована на автоматизацію роботи клубів, включно з контролем доступу, обліком клієнтів, фінансовими операціями та розкладом тренувань. Незважаючи на функціональність, FitCloud здебільшого виконує роль адміністративної системи й не включає сучасних інструментів, пов’язаних із рекомендаційними алгоритмами, персоналізованим тренувальним процесом або гейміфікацією. Користувацький інтерфейс платформи є застарілим порівняно зі світовими аналогами, що знижує привабливість серед молодого покоління клієнтів, які звикли до динамічних, адаптивних та візуально привабливих рішень.

Певну нішу займають мобільні програми, орієнтовані на самостійне тренування, такі як Freeletics, Nike Training Club, Fitbod та інші. Вони забезпечують можливість перегляду тренувальних програм, отримання рекомендацій, аналізу результатів і підтримки регулярної активності. Проте такі додатки не є системами для тренажерних залів, адже не дозволяють керувати розкладом, контролювати відвідуваність, взаємодіяти з

адміністрацією, отримувати бали за відвідування або бути частиною спільноти тренажерного залу. Крім того, вони не підтримують інтеграцію з реальним середовищем фітнес-центру, що суттєво обмежує їх використання як інструменту управління.

Сучасні CRM-системи, такі як PerfectGym або Glofox, орієнтовані переважно на бізнес-аспекти: продажі, фінанси, управління клієнтською базою, створення маркетингових кампаній. Вони є потужними інструментами для адміністрації, але не містять розвиненої клієнтської частини, орієнтованої на взаємодію з тренуваннями, статистикою або персональними рекомендаціями. Багато з них обмежені у можливостях адаптації та не передбачають інтеграції AI-модулів без суттєвих фінансових витрат.

Таким чином, аналіз існуючих платформ демонструє, що більшість рішень або орієнтовані на управлінські функції без належної уваги до тренувального процесу, або фокусуються на клієнтській взаємодії, але не підтримують адміністративні та аналітичні функції. Є також рішення, які пропонують тренувальні програми, але не забезпечують зв’язку з тренажерним залом як фізичним простором і не містять можливостей комплексного управління. Практично повністю відсутні системи, які включають одночасно AI-тренер, систему гейміфікації, гнучкий інтерфейс розкладу, персональний кабінет, аналіз завантаженості залу, адміністративну панель та повноцінну інтеграцію з базою даних.

Ці недоліки існуючих платформ підтверджують актуальність розроблення інформаційної системи, яка поєднала би функціонал управління тренувальним процесом, інструменти мотивації, персоналізовані рекомендації, адміністративний контроль, аналітичні модулі та зручний адаптивний інтерфейс. Саме потреба у такому комплексному рішенні визначає інноваційність проекту та обґрунтовує його практичну значущість.

1.5. Концептуальні вимоги до веб-орієнтованої інформаційної системи управління тренувальним процесом

Розроблення сучасної веб-орієнтованої інформаційної системи для управління тренувальним процесом передбачає врахування широкого спектра функціональних, технологічних, ергономічних і експлуатаційних вимог, що зумовлені інтенсивною цифровізацією фітнес-індустрії. Така система має забезпечувати високу ефективність тренувального процесу, комфортну взаємодію з користувачем, гнучкість управління та можливість масштабування. В основі формування вимог лежить аналіз сучасних тенденцій у сфері спортивних цифрових рішень, очікувань користувачів та потреб адміністративного персоналу тренажерного залу. Розуміння цих вимог дає змогу створити платформу, яка не лише відповідає базовим стандартам галузі, а й забезпечує розширений, інноваційний функціонал, що підсилює конкурентоспроможність закладу.

Однією з ключових концептуальних вимог до такої системи є забезпечення персоналізації тренувального процесу. Сучасні користувачі очікують, що система буде здатна адаптувати тренування відповідно до їхнього рівня фізичної підготовки, стану здоров’я, персональних цілей та попередніх результатів. У цьому контексті важливо передбачити механізми збору й аналізу індивідуальних даних, зокрема параметрів тіла, історії тренувань, активності та прогресу. Особливого значення набуває інтеграція рекомендаційних алгоритмів або модулів штучного інтелекту, здатних формувати персональні тренувальні плани та надавати корисні поради. Відповідно до цих вимог у системі PowerGym передбачено AI-модуль на основі OpenAI, який аналізує введені користувачем параметри та генерує індивідуальні рекомендації, що відповідають сучасним підходам до персоналізації тренувального процесу.

Важливим критерієм є забезпечення зручної та інтуїтивно зрозумілої взаємодії між користувачем і системою. Інтерфейс веб-платформи має бути

адаптивним, доступним на різних пристроях і оптимізованим для швидкої навігації. Оскільки тренажерні зали обслуговують аудиторію з різним рівнем цифрової грамотності, система повинна мінімізувати когнітивне навантаження, забезпечуючи доступ до ключових функцій у кілька кліків. Використання сучасних фронтенд-технологій, таких як React/Next.js та Tailwind CSS, дозволяє створити візуально привабливий, структурований і адаптивний інтерфейс, що відповідає сучасним вимогам до веб-систем. У PowerGym ці принципи реалізуються через компонентну архітектуру інтерфейсу та використання сучасних засобів UI-розробки.

Суттєвим елементом концептуальних вимог є автоматизація ключових бізнес-процесів тренажерного залу, включаючи управління розкладом занять, відстеження відвідуваності, контроль завантаженості залу в режимі реального часу, управління абонементами та системою балів. Інформаційна система повинна дозволяти адміністрації та тренерам швидко вносити зміни в розклад, додавати нові тренування, керувати статистикою активності користувачів та формувати звіти. Особливо важливо забезпечити актуальність інформації, щоб користувач міг оперативно орієнтуватися у графіку та не зустрічав ситуацій із переповненістю залу. У PowerGym цьому приділено особливу увагу: система реалізує концепцію динамічного розкладу, а також моделі даних для роботи з відвідуваннями, тренуваннями та адміністративним контентом.

У сучасних умовах значущою вимогою є наявність системи мотивації та підтримки тренувальної активності, оскільки саме вона забезпечує утримання клієнтів та сприяє їх довгостроковій залученості. Наявність рейтингової системи, системи досягнень, балів за відвідування та інших елементів гейміфікації значно підвищує інтерес клієнтів до тренувального процесу, стимулює регулярність занять і створює елемент соціальної взаємодії. З огляду на це інформаційна система повинна мати механізми фіксації активності, нарахування бонусів і формування індивідуальної статистики користувача. У системі PowerGym ці вимоги враховані через

впровадження особистого кабінету, де користувач може переглядати свої досягнення, рейтинг та накопичені бали.

Не менш важливою вимогою є забезпечення ефективної роботи адміністрації тренажерного залу, яка повинна мати доступ до розширеної інформації про діяльність закладу. Інформаційна система має включати адмін-панель для управління розкладом, контентом, статистикою відвідуваності, а також механізми автоматичного формування звітів. Це дозволяє оптимізувати діяльність персоналу, мінімізувати ручну працю та забезпечити точність даних. У PowerGym структура передбачає повноцінну адміністративну панель, що відповідає сучасним уявленням про управління фітнес-центром.

Системи такого рівня повинні також відповідати вимогам технологічної масштабованості та надійності. Враховуючи постійне зростання кількості користувачів, збільшення обсягу даних і потребу в безперервному доступі до системи, необхідно забезпечити стійку архітектуру, здатну ефективно функціонувати при розширенні проєкту. Використання хмарних або контейнеризованих рішень, оптимізація бази даних, поділ логіки на модулі та застосування серверних API-маршрутів є важливими аспектами забезпечення стабільної роботи платформи. Система PowerGym базується на Next.js та MongoDB, що дає змогу забезпечити високу гнучкість, продуктивність та легкість масштабування.

Концептуальні вимоги до веб-орієнтованої інформаційної системи управління тренувальним процесом охоплюють великий спектр аспектів — від персоналізації тренувань та інтуїтивного інтерфейсу до безпеки, масштабованості й гейміфікації. Ці вимоги формують комплексне бачення того, якою має бути сучасна цифрова платформа для тренажерного залу. Саме вони стали основою проєктування та реалізації системи PowerGym, де всі ключові концептуальні вимоги були враховані й реалізовані в програмній частині, що підтверджує актуальність та практичну цінність створеного рішення.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ POWERGYM

2.1. Загальна концепція та призначення системи

Розроблення веб-орієнтованої інформаційної системи PowerGym базується на необхідності створення цифрової платформи нового покоління, здатної комплексно автоматизувати тренувальний процес у тренажерному залі, забезпечити високу ефективність організації занять, підтримувати взаємодію з користувачами та оптимізувати діяльність адміністративного персоналу. Концепція системи спрямована на вирішення проблем, притаманних традиційним моделям управління фітнес-центром, де значна частина процесів виконується вручну, інформація є фрагментованою, а клієнтська взаємодія обмежена стандартними засобами запису та обліку.

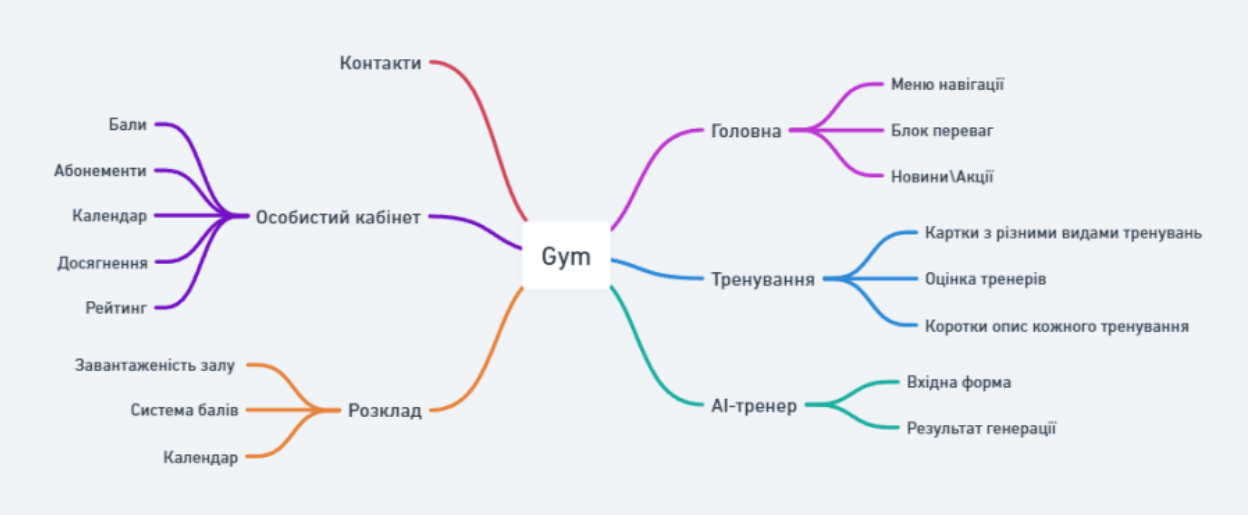

Рис.2.1.1. Схема управління від профілю «Користувач»

PowerGym створюється як єдина інтегрована платформа, що поєднує у собі функціонал користувацької та адміністративної частин, інструменти персоналізації тренувального процесу, модулі статистики та аналітики, а також AI-компонент для формування індивідуальних рекомендацій. Основною концепцією системи є забезпечення комплексної цифрової

взаємодії всіх учасників тренувального процесу — клієнтів, тренерів та адміністрації — у межах єдиного веб-середовища.

Система PowerGym реалізує сучасний підхід до ведення тренувальної діяльності, заснований на принципах адаптивності, прозорості, доступності та інтерактивності. Завдяки використанню інтернет-технологій клієнти отримують можливість переглядати актуальний розклад занять, планувати відвідування, контролювати власну активність, стежити за прогресом, формувати персональні тренувальні плани й взаємодіяти зі штучним інтелектом. Такий підхід не лише підвищує якість тренувань, але й забезпечує високий рівень самостійності користувача, що відповідає тенденціям сучасного фітнес-ринку.

З точки зору адміністративного управління PowerGym надає можливість оперативно працювати з даними користувачів, керувати розкладом занять, створювати новини й акції, контролювати завантаження залу, здійснювати моніторинг активності та формувати автоматичні звіти. Усі ці функції спрямовані на оптимізацію бізнес-процесів, зниження операційних витрат, покращення якості сервісу та підвищення ефективності роботи персоналу. Інформаційна система виступає інструментом, який дозволяє керівництву приймати виважені та обґрунтовані рішення, спираючись на реальні дані та їх аналітичну інтерпретацію.

Рис.2.1.2. Схема адміністративного управління

Концепція PowerGym передбачає також розширену підтримку користувацького досвіду. Система включає особистий кабінет, де зберігається індивідуальна статистика користувача, його досягнення, рейтингова інформація, бали за відвідування, придбані абонементи та персональний календар тренувань. Така персональна цифрова зона створює ефект занурення у тренувальний процес і мотивує користувача до регулярності та системності фізичних занять.

Особливого значення у концепції PowerGym набуває модуль AI-тренера, інтегрований через OpenAI API. Він дозволяє користувачу отримати персональні поради та тренувальні програми, сформовані на основі введених даних — фізичних параметрів, цілей, рівня активності та інших характеристик. Впровадження штучного інтелекту є важливим елементом інноваційності системи та відповідає світовим тенденціям у розвитку персоналізованого фітнесу. Це створює ефект присутності власного віртуального тренера, доступного у будь-який час.

Важливим елементом концепції є структура та архітектура системи. PowerGym побудована з використанням сучасних веб-технологій: Next.js як

основного фреймворку, Node.js як серверного середовища та MongoDB як гнучкої документно-орієнтованої бази даних. Такий технологічний стек забезпечує високу продуктивність, можливість серверного рендерингу, гнучкість у роботі з даними та легкість масштабування. Компонентна структура інтерфейсу дозволяє створювати адаптивний, швидкий та інтерактивний веб-додаток, що відповідає вимогам сучасних користувачів.

У процесі проєктування PowerGym особлива увага приділяється безпеці даних та надійності роботи системи. Передбачається впровадження механізмів авторизації, контролю доступу, шифрування, безпечних запитів до бази даних і захищених API-маршрутів. Оскільки система працює з персональними даними клієнтів, безпека є одним із базових пріоритетів у концептуальному дизайні платформи.

Загальна концепція PowerGym ґрунтується на створенні багатофункціональної веб-платформи, здатної інтегрувати в собі весь спектр інструментів для роботи тренажерного залу та його клієнтів. Система покликана автоматизувати тренувальний процес, удосконалити взаємодію між користувачем та тренером, забезпечити високу якість сервісу, підтримувати аналітичні функції та створювати персональний тренувальний простір. Завдяки комплексності, інноваційності та сучасному технічному підходу PowerGym виступає ефективним інструментом цифрової трансформації тренажерного залу.

2.2. Функціональні вимоги до системи

Функціональні вимоги визначають перелік можливостей, які веб-орієнтована інформаційна система PowerGym повинна забезпечувати для ефективної підтримки тренувального процесу, взаємодії з користувачами та управління діяльністю тренажерного залу. Саме вони визначають структуру, логіку роботи й кінцевий спектр функцій платформи. У контексті сучасних тенденцій фітнес-індустрії функціональні вимоги мають охоплювати як базові, так і розширені можливості, що забезпечують персоналізацію сервісу, автоматизацію бізнес-процесів і високу якість користувацького досвіду. Всі

наведені вимоги стали основою проєктування системи PowerGym, і їхня реалізація забезпечує формування повноцінного цифрового середовища тренажерного залу.

Однією з основних вимог є забезпечення доступу до загальної інформації про тренажерний зал. Система повинна містити публічний інтерфейс, доступний усім відвідувачам без авторизації. У цьому розділі користувачі мають можливість переглядати переваги закладу, ознайомлюватися з новинами та акціями, а також отримувати базову інформацію про доступні послуги. Такий функціонал формує перше враження про заклад і впливає на подальшу мотивацію користувача до взаємодії із системою. У PowerGym це реалізовано у вигляді окремих сторінок «Головна», «Новини та акції», що дає змогу залучати потенційних клієнтів і підтримувати постійний інтерес існуючих.

Наступною ключовою вимогою є можливість перегляду та вибору типів тренувань. Користувач повинен мати доступ до каталогу тренувальних програм, де кожне тренування супроводжується детальним описом, рекомендаціями, рівнем інтенсивності та характеристиками. Додатково важливим є механізм оцінки тренерів, що підвищує прозорість взаємодії та сприяє вибору найкращого тренера для конкретних занять. У системі PowerGym функціонал перегляду тренувань реалізований через спеціалізований модуль «Тренування», у якому користувач може знайти опис конкретних напрямків та ознайомитися з відгуками інших відвідувачів.

Особливого значення набуває функціональна вимога щодо інтелектуального формування персональних тренувальних програм. У сучасних фітнес-системах користувачі очікують інструментів, які враховують їхні фізичні параметри, стан здоров’я, рівень підготовки, цілі тренувань та індивідуальні особливості. Саме тому одним із ключових елементів PowerGym є модуль «AI-тренер», який забезпечує генерацію персоналізованих рекомендацій. За допомогою інтеграції з OpenAI API система аналізує введені користувачем дані та створює індивідуальний

тренувальний план. Цей інноваційний функціонал суттєво підвищує рівень наукової обґрунтованості тренувального процесу та робить систему унікальною порівняно з традиційними фітнес-платформами.

Важливим функціональним компонентом є динамічний модуль розкладу, який повинен забезпечувати можливість перегляду календаря занять, фільтрації тренувань за типами чи тренерами, відображення завантаженості залу в реальному часі, а також можливість запису на заняття. Крім цього, модуль має надавати інформацію про кількість доступних місць, зміни в розкладі та майбутні події, що формує комфортну систему планування тренувань. У PowerGym реалізовано базові елементи такого модуля, включаючи календар подій, відображення поточної активності та інтеграцію з іншими розділами системи.

Ключовою вимогою до системи є наявність особистого кабінету користувача, у якому зберігається повна інформація про його активність. Особистий кабінет має включати систему балів за відвідування, інформацію про абонементи, персональний календар тренувань, історію відвідувань, список досягнень, а також рейтинг користувача серед інших відвідувачів. Наявність такої функціональності сприяє гейміфікації тренувального процесу, збільшує мотивацію користувачів та дає змогу персоналізувати їхній досвід. У PowerGym усі ці елементи вже закладені в структуру системи та реалізовані в окремих модулях.

Суттєвою групою функціональних вимог є вимоги до адміністративної частини системи, яка повинна забезпечувати можливість керувати всіма аспектами діяльності тренажерного залу. Адмін-панель має підтримувати керування розкладом тренувань, створення новин та акцій, управління видами тренувань, перегляд статистики відвідуваності, формування автоматичних звітів та контроль за активністю користувачів. Такий функціонал суттєво зменшує навантаження на персонал та забезпечує високу точність усіх бізнес-процесів. Система PowerGym передбачає впровадження

повноцінної адміністративної панелі, що відповідає вимогам сучасних систем управління закладом.

Додатковою, але важливою вимогою є інтегрованість і узгодженість усіх модулів, які повинні взаємодіяти через єдине інформаційне середовище та спільну базу даних. Кожна дія користувача або адміністратора має синхронно оновлювати дані, що гарантує актуальність інформації та зручність у використанні. Взаємозв'язок модулів «Тренування», «Розклад», «Особистий кабінет» та «Адмін-панель» через MongoDB забезпечує єдину логіку роботи системи.

Оскільки система працює у веб-середовищі, важливою вимогою є забезпечення швидкодії та доступності функцій незалежно від пристрою. Система має коректно працювати як на стаціонарних комп'ютерах, так і на мобільних телефонах. Використання сучасних технологій — Next.js, Tailwind CSS, адаптивного дизайну та серверних API — дозволяє PowerGym повністю відповідати цим вимогам.

Функціональні вимоги до PowerGym формують основу для створення комплексної, інтерактивної, персоналізованої та високотехнологічної системи управління тренувальним процесом. Усі вимоги, визначені на цьому етапі, стали фундаментом для подальшого проєктування архітектури системи, структури бази даних, логіки роботи окремих модулів і механізмів взаємодії між користувачем та адміністратором.

2.3. Нефункціональні вимоги (швидкодія, безпека, масштабованість)

Нефункціональні вимоги визначають якісні характеристики інформаційної системи, що не пов'язані безпосередньо зі змістом її функцій, але визначають ефективність, стабільність і надійність роботи всієї платформи. У контексті розроблення веб-орієнтованої системи PowerGym ці вимоги набувають особливого значення, оскільки система повинна забезпечувати цілодобовий доступ, працювати з персональними даними користувачів, гарантувати високу продуктивність та відповідати сучасним стандартам взаємодії в мережі. Нефункціональні вимоги PowerGym

формують фундаментальні властивості системи, які забезпечують її практичну цінність і визначають можливість масштабування у майбутньому.

Одним із найважливіших аспектів нефункціональних вимог є швидкодія та продуктивність системи. Оскільки PowerGym працює у веб-середовищі, критично важливо забезпечити швидке завантаження сторінок, мінімальні затримки при взаємодії з інтерфейсом та оперативну обробку запитів. Для цього застосовується серверний та гібридний рендеринг, реалізований фреймворком Next.js, що дозволяє суттєво скоротити час генерації сторінок та відображення контенту користувачу. Завдяки механізмам кешування, оптимізованій роботі маршрутизатора та ефективній взаємодії з базою даних система здатна обробляти значну кількість одночасних запитів без зниження швидкодії. Особлива увага приділяється ефективності MongoDB, де документно-орієнтована модель зберігання даних дає змогу швидко виконувати запити, здійснювати пошук та оновлення інформації. Забезпечення високої швидкодії є необхідною умовою для комфортної роботи користувачів, особливо під час доступу до інформації про розклад, перегляду тренувань або отримання результатів AI-тренера в режимі реального часу.

Наступною групою нефункціональних вимог є вимоги до безпеки, що охоплюють захист персональних даних, безпечну автентифікацію, цілісність інформації та захист від несанкціонованого доступу. Система PowerGym працює з персональними профілями користувачів, історією тренувань, даними про відвідування та іншими відомостями, які підлягають захисту згідно з чинними нормами кібербезпеки. Тому у системі застосовується механізм автентифікації, який забезпечує надійний контроль доступу через токени або сесії, залежно від обраної технології. Важливим елементом безпеки є захист API-маршрутів, які не повинні бути доступними стороннім користувачам та повинні перевіряти права доступу перед виконанням будь-яких операцій. Крім цього, необхідно забезпечити захист від типових мережевих загроз — SQL/NoSQL Injection, XSS-атак, CSRF, brute-force та

спроб несанкціонованої модифікації даних. Система PowerGym передбачає використання серверних маршрутів у Next.js, які додатково підсилюють безпеку, оскільки приховують внутрішню логіку та дозволяють виконувати операції виключно на сервері. Безпека також забезпечується на рівні підключення до MongoDB, що включає захищений канал зв’язку та безпечне зберігання конфіденційних даних у файлі середовища.

Важливою характеристикою є масштабованість системи, тобто можливість її адаптації до зростання навантаження, збільшення кількості користувачів, обсягу даних та кількості функціональних модулів. Оскільки тренажерні зали можуть поступово розширювати свою клієнтську базу, переходити на нові формати роботи або додавати нові послуги, інформаційна система повинна відповідати потенційним змінам і давати змогу розширювати функціонал без суттєвих реконструкцій архітектури. Завдяки використанню Next.js і компонентної архітектури інтерфейсу модулі системи PowerGym можуть легко розширюватися або змінюватися, не впливаючи на загальну структуру. MongoDB забезпечує гнучкість у зберіганні даних, оскільки документно-орієнтована модель дозволяє додавати нові поля, колекції чи зв’язки без потреби складних міграційних процедур. У разі збільшення навантаження система може бути розгорнута у хмарному середовищі з автоматичним масштабуванням, що забезпечить стабільність роботи навіть при значному рості числа одночасних користувачів.

До важливих нефункціональних вимог також належать надійність і стабільність роботи системи. Платформа має бути доступною в будь-який час, забезпечувати збереження та відновлення даних у разі виникнення збоїв, а також коректно функціонувати у випадку часткових відмов серверних компонентів. Стабільність системи гарантується використанням продуманих алгоритмів підключення до бази даних, обробки помилок, логування та відстеження критичних подій. У PowerGym підключення до MongoDB реалізовано з використанням кешування з’єднань, що зменшує навантаження на сервер та мінімізує ризики надмірної кількості одночасних підключень.

Додатково використання серверної інфраструктури Next.js забезпечує можливість обробки критичних процесів на серверній стороні, а також можливість автоматичного регенерування сторінок при оновленні даних.

Ще однією важливою нефункціональною вимогою є зручність використання (юзабіліті), що визначає, наскільки комфортно кінцевому користувачу працювати з системою. Інтерфейс PowerGym має бути інтуїтивно зрозумілим, логічно структурованим, адаптивним щодо різних розмірів екранів і привабливим з точки зору дизайну. Необхідно забезпечити зрозумілу навігацію між розділами, швидкий доступ до ключових функцій, мінімізацію кількості кроків для виконання базових дій, а також використання простих та однакових елементів UI. Tailwind CSS, на якому побудований інтерфейс PowerGym, дозволяє реалізувати сучасний дизайн, який відповідає очікуванням користувачів та міжнародним стандартам.

Окремо до нефункціональних вимог належить портативність та сумісність, що означає можливість стабільної роботи системи на різних браузерах, пристроях та операційних системах. PowerGym використовує веб-технології, що забезпечують однакову доступність у популярних браузерах (Chrome, Firefox, Safari, Edge) та на різних платформах — Windows, macOS, Android, iOS. Це дозволяє зробити систему універсальною та доступною широкому колу користувачів.

Підсумовуючи, нефункціональні вимоги PowerGym включають комплекс якісних характеристик — швидкодію, безпеку, масштабованість, надійність, зручність використання та сумісність, — які забезпечують високу ефективність роботи системи, її стійкість до зростання навантажень та адаптивність до подальшого розвитку. Виконання цих вимог гарантує, що система PowerGym може бути інтегрована у реальну інфраструктуру тренажерного залу та використовуватися як інструмент довготривалої цифрової трансформації.

2.4. Архітектура системи

Архітектура інформаційної системи PowerGym ґрунтується на сучасних принципах розробки веб-орієнтованих платформ, які передбачають чітке розмежування логіки, зберігання даних та відображення інтерфейсу. Проєкт створено з використанням технологій Next.js, Node.js, MongoDB та Mongoose, що забезпечує модульність, масштабованість і високу продуктивність роботи. Архітектуру системи можна описати як багаторівневу, де кожен шар виконує власний набір функцій і взаємодіє з іншими через стандартизовані механізми обміну інформацією. Такий підхід дозволяє легко розширювати функціональність, упроваджувати нові модулі, оптимізувати процеси та підтримувати систему протягом тривалого часу.

На рівні представлення PowerGym використовує компонентну модель інтерфейсу, що реалізована у фреймворку Next.js. Кожен розділ системи — такі як «Головна», «Тренування», «AI-тренер», «Розклад», «Особистий кабінет», «Контакти» та «Адмін-панель» — представлений окремим компонентом, який відповідає за відображення інформації та взаємодію з користувачем. Компоненти побудовані таким чином, щоб мінімізувати час завантаження сторінок, забезпечити адаптивність на різних пристроях та гарантувати інтуїтивність навігації. Tailwind CSS використовується для швидкого створення стилів і гнучкої адаптації інтерфейсу під мобільні телефони, планшети та великі монітори. Рівень інтерфейсу є уніфікованим для користувачів і адміністратора, однак набір функцій та можливостей у кожному випадку відрізняється і визначається логікою доступу.

Серверна частина системи, що є другим ключовим рівнем архітектури, реалізована як набір API-маршрутів, які обробляють усі запити користувачів і адміністратора. Саме тут розташована бізнес-логіка: перевірка авторизації, обробка форм, виконання CRUD-операцій, генерація рекомендацій, оновлення статистики, робота з розкладом, облік відвідувань, нарахування балів та обробка даних у реальному часі. Це дозволяє клієнтській частині залишатися простою та легкою, а всю інтенсивну обробку перекладати на сервер. Серверні функції також відповідають за інтеграцію з OpenAI API для

роботи AI-тренера. Всі дані, які вводить користувач у форму AI-тренера, обробляються на сервері, після чого система формує запит до моделі ШІ та повертає персональний тренувальний план разом із рекомендаціями.

Третім рівнем архітектури є база даних MongoDB, у якій зберігаються всі ключові сутності: дані про користувачів, історію відвідувань, доступні тренування, абонементи, новини, події, оцінки тренерів, розклад занять, нараховані бали мотивації та формовані досягнення. Mongoose використовується як інструмент для опису схем, валідації, типізації та організації структурованого доступу до даних. Кожна сутність моделюється як окремий документ із власними полями, зв’язками, обмеженнями та типовими значеннями. Завдяки документній природі MongoDB система може легко масштабуватися, додавати нові поля без порушення структури та підтримувати високий рівень продуктивності навіть у разі зростання кількості користувачів.

Взаємодія між рівнями системи реалізована за допомогою стандартизованих API-викликів. Користувач працює з інтерфейсом, який надсилає запити на сервер. Сервер обробляє запит, звертається до бази даних, отримує або оновлює дані та повертає відповідь клієнтському інтерфейсу. Цей процес однаково працює як для звичайного користувача, так і для адміністратора, але набір доступних операцій залежить від ролі. Користувач може переглядати тренування, використовувати AI-тренера, аналізувати завантаженість залу, переглядати власні бали, відстежувати прогрес та оцінювати тренера. Адміністратор, своєю чергою, може створювати події, додавати новини, управляти розкладом, реєструвати нових тренерів та користувачів, призначати персональні тренування клієнтам і переглядати статистику активності залу. Логіка роботи адміністратора відображена у спеціальній адміністративній панелі, яка має розширений набір інструментів та дозволяє управляти всією платформою.

Загальна архітектура системи передбачає чіткий поділ відповідальностей, що сприяє стабільності роботи та полегшує процес

оновлення або модернізації. Важливим елементом є контроль потоків даних: від моменту, коли користувач взаємодіє з інтерфейсом, до моменту збереження або вивантаження інформації з бази даних. На рівні бізнес-процесів система підтримує два основні напрямки: користувацький і адміністративний. Користувацький напрямок охоплює виконання тренувань, перегляд інформації, отримання рекомендацій, участь у рейтингах і досягненнях. Адміністративний напрямок включає управління контентом, контроль активності користувачів, управління тренера-ми та призначення тренувальних програм. Щоб відобразити централізовану роль адміністратора, у роботі використовується схема взаємодії адміністратора з підсистемами PowerGym (рисунок 2.3), що чітко демонструє повноваження та ключові функції адміністративної ролі.

Узагальнюючи опис архітектури, можна зазначити, що PowerGym є сучасною, технологічно розвиненою інформаційною системою, яка поєднує модульність, високу продуктивність, інтелектуальні алгоритми, гнучку організацію даних і адаптивність інтерфейсу. Використання Next.js та MongoDB дозволяє досягти високої ефективності роботи системи, забезпечити швидкий доступ до даних, оптимізувати обробку запитів і сформувати перспективну платформу, готову до подальшого розвитку та масштабування. Такий архітектурний підхід забезпечує можливість інтеграції нових технологій, розширення функціональності, впровадження мобільного застосунку та поглиблення використання штучного інтелекту.

2.5. Структура програмного забезпечення інформаційної системи PowerGYM

Структура програмного забезпечення системи PowerGym побудована відповідно до сучасних принципів модульності, розділення відповідальностей та багаторівневої організації коду, характерної для застосунків, створених на основі архітектури Next.js. Така структура забезпечує зручність підтримки, масштабованість, незалежний розвиток окремих модулів, а також чітку організацію бізнес-логіки, серверної частини,

інтерфейсу користувача та моделей даних. Папки та файли в проєкті впорядковані таким чином, щоб розмежувати логіку клієнтської частини, API-маршрутів, компонентів інтерфейсу, моделей MongoDB та допоміжних службових модулів. Нижче розглядається повна структура каталогу PowerGym та призначення ключових його елементів.

Структура програмного забезпечення PowerGym сформована з урахуванням рекомендацій щодо побудови модульних веб-орієнтованих систем та відображає логічний поділ усіх функціональних частин застосунку. Для кращої наочності загальну організацію каталогів та ключових файлів системи подано у таблиці 2.7.1, де наведено основні папки проєкту та їх призначення. Це дозволяє узагальнити архітектурну модель PowerGym перед переходом до детального опису кожної підсистеми.

Таблиця 2.5.1.

Папка / файл	**Призначення**
/app	Основна директорія клієнтської частини Next.js; містить усі сторінки, що відповідають за відображення інтерфейсу та маршрутизацію.
/app/admin	Інтерфейс адміністратора: керування розкладом, новинами, подіями, тренерами, клієнтами та статистикою.
/app/ai-trainer	Сторінка роботи модуля AI-тренера; введення параметрів та отримання персонального плану тренувань.
/app/trainings	Відображення доступних видів тренувань; картки тренувань, описи, оцінки тренерів.
/app/schedule	Сторінка розкладу занять, перегляд завантаженості залу в реальному часі.
/app/profile	Особистий кабінет користувача: бали, абонементи, календар, досягнення, рейтинг.
/app/contacts	Форма зворотного зв’язку та контактна інформація залу.
/app/login, /app/register	Авторизація та реєстрація користувачів.
/api	Серверні маршрути (API) для обробки даних, CRUD-операцій та взаємодії з MongoDB.
/api/news	API-маршрути для створення, редагування та отримання новин.
/api/trainings	Маршрути для роботи з тренуваннями: додавання, редагування, видалення, запити.

/api/users	Обробка користувачів, реєстрація, оновлення даних, отримання профілю.
/components	Повторно використовувані React-компоненти інтерфейсу (Navbar, Hero, TrainingCard, NewsCard, Footer тощо).
/lib/mongodb.ts	Логіка підключення до MongoDB із кешуванням з'єднань; серце серверної роботи з базою даних.
/models	Mongoose-моделі, що описують структуру документів у MongoDB.
/models/User.ts	Користувач системи: профіль, бали, досягнення, ролі.
/models/Trainer.ts	Модель тренера, якого адміністратор може прив'язати до тренування.
/models/Training.ts	Опис тренувань: назва, складність, інтенсивність, тренер.
/models/PersonalTraining.ts	Персональні тренування, призначені адміністратором конкретному користувачу.
/models/AiPlan.ts	Результати AI-тренера, що генеруються через OpenAI.
/models/Booking.ts	Записи користувачів на тренування (бронювання місць).
/models/PointsLog.ts	Журнал нарахованих балів мотивації.
/models/News.ts	Новини та акції тренажерного залу.
/models/GymLoad.ts	Дані про завантаженість залу в реальному часі.
/public	Статичні файли: зображення, іконки, фони, логотипи.
/src	Допоміжні модулі або службові частини проєкту.
next.config.ts	Конфігурація збірки Next.js, оптимізація серверного та клієнтського рендерингу.
tailwind.config.js	Налаштування Tailwind CSS, кастомні стилі та палітра.
tsconfig.json	Конфігурація TypeScript, типізація, правила компіляції.
package.json	Перелік залежностей і скриптів проєкту PowerGym.

На верхньому рівні проєкт містить основні директорії api, app, components, lib, models, public, src, а також конфігураційні файли, необхідні для роботи системи (зокрема next.config.ts, tailwind.config.js, tsconfig.json, package.json, конфігураційні файли стилізації та інші службові артефакти).

Кожна папка виконує власну роль у програмній архітектурі, забезпечуючи чітке розмежування між функціональними частинами платформи.

Директорія app є центральним ядром клієнтської частини та містить усі сторінки веб-системи, реалізовані за допомогою файлового роутінгу Next.js. Усередині app знаходяться окремі піддиректорії, що відповідають структурам сторінок: ai-trainer, contacts, login, profile, register, schedule, subscriptions, trainings, а також спеціальна директорія admin, яка містить інтерфейс адміністратора. Кожна сторінка реалізована у вигляді React-компонента формату .tsx, що отримує дані з API, відображає інформацію, надає доступ до функціональних елементів і координує взаємодію користувача з іншими модулями системи. Крім сторінок, у кореневій частині директорії знаходяться файли layout.tsx та page.tsx — глобальні компоненти розмітки, які забезпечують сталий вигляд інтерфейсу та спільні стилі.

Особливо важливою є директорія api, у якій зосереджено серверні обробники, що відповідають за виконання CRUD-операцій, взаємодію з базою даних, управління розкладом, відправлення форм, генерацію рекомендацій AI-тренера та обробку запитів адміністратора. Вона містить три основні групи API-маршрутів: /api/news, /api/trainings, /api/users. Ці маршрути виконують функції отримання, оновлення та збереження даних, включно з тренувальними програмами, користувачами, новинами та іншими інформаційними сутностями. Така організація відповідає концепції REST та дозволяє легко розширювати функціональність системи шляхом додавання нових маршрутів.

Важливою частиною проєкту є папка models, де зосереджені схеми MongoDB, створені за допомогою Mongoose. Моделі включають такі сутності, як AiPlan.ts, Booking.ts, ContactMessage.ts, GymLoad.ts, News.ts, PersonalTraining.ts, PointsLog.ts, Trainer.ts, Training.ts, User.ts. Вони описують структуру документів, обмеження, типи полів, встановлюють взаємозв’язки між сутностями та забезпечують коректну організацію даних у базі. Наприклад, модель User включає інформацію про профіль користувача, бали

мотивації, історію тренувань і параметри доступу; модель Trainer зберігає дані про тренерів, яких може прив’язувати адміністратор до персональних тренувань; модель AiPlan зберігає результати роботи AI-тренера; модель Training описує види тренувань, їх характеристики та рівень інтенсивності. Застосування Mongoose забезпечує надійну валідацію даних і дозволяє чітко контролювати структуру всіх документів у MongoDB.

Директорія components містить повторно використовувані UI-компоненти, які формують інтерфейс користувача та забезпечують єдиний стиль усієї системи. Серед компонентів — BenefitCard.tsx, Footer.tsx, Hero.tsx, Navbar.tsx, NewsCard.tsx, NewsSection.tsx, TrainingCard.tsx. Вони використовуються на різних сторінках, забезпечуючи модульність та уніфікований підхід до формування інтерфейсу. Наприклад, TrainingCard відображає тренування у вигляді картки, NewsCard відповідає за відображення новин, Hero будує основний промоблок головної сторінки, а Navbar визначає глобальну навігацію між розділами. Завдяки поділу інтерфейсу на компоненти підвищується гнучкість і зручність модернізації зовнішнього вигляду системи.

Папка lib містить службові модулі, включно з критичним файлом підключення до бази даних — mongodb.ts. Саме тут реалізовано логіку встановлення з’єднання з MongoDB, виконання кешування з’єднання для оптимізації продуктивності та захист від надмірного навантаження на базу під час роботи серверних компонентів Next.js. Логіка підключення побудована з урахуванням особливостей архітектури Next.js, у якій серверні компоненти можуть викликатися неодноразово, тому кешування є необхідним елементом ефективної роботи системи.

Окремою частиною проєкту є директорії public та src. Директорія public містить статичні ресурси — зображення, стилі, файли іконок, шрифти та інші матеріали, що використовуються фронтендом. Директорія src може містити додаткові модулі або логіку, що не належить до конкретних груп (як-от бізнес-процеси чи утиліти), залежно від вимог реалізації.

Серед конфігураційних файлів особливу роль відіграють next.config.ts, що керує загальними параметрами збірки; tailwind.config.js, який визначає налаштування Tailwind CSS; tsconfig.json, що відповідає за налаштування TypeScript; package.json, який містить інформацію про залежності, скрипти та конфігурації проєкту. Наявність цих файлів забезпечує узгодженість усіх технологічних компонентів і дозволяє легко розгортати систему на різних середовищах.

Структура програмного забезпечення PowerGym реалізує всі сучасні вимоги до побудови веб-систем, включно з модульністю, розширюваністю, чітким розмежуванням логіки та зрозумілою організацією файлів. Такий підхід забезпечує стабільну роботу системи, зручність її оновлення та можливість подальшого масштабування.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Середовище розробки та інструменти

Розроблення інформаційної системи PowerGym здійснювалось із використанням сучасних технологій, інструментів та середовища програмування, що забезпечують високу продуктивність, модульність і гнучкість програмного забезпечення. Вибір інструментарію був обумовлений специфікою проєкту, необхідністю інтеграції з базами даних, забезпеченням високого рівня адаптивності інтерфейсу, а також потребою у швидкій розробці, тестуванні та розгортанні веб-додатку.

Основним середовищем розробки, використаним у ході реалізації системи, виступало **Visual Studio Code** — сучасний кросплатформний редактор коду, який забезпечує підтримку великої кількості мов програмування, інтеграцію з системами контролю версій, гнучкість налаштувань та розширень. Використання VS Code дозволило створити оптимальне робоче середовище завдяки підтримці підсвічування синтаксису TypeScript, автоматичного форматування коду, інтеграції з GitHub, а також використання терміналу безпосередньо всередині редактора. Значну роль відіграли розширення для роботи з React, Tailwind CSS, Next.js та MongoDB, які покращили продуктивність розробника та прискорили процес створення компонентів.

В основі backend- та frontend-частини PowerGym лежить фреймворк **Next.js** — сучасна платформа для створення веб-додатків на базі React. Next.js забезпечує потужні можливості серверного рендерингу (SSR), статичної генерації (SSG), гібридного рендерингу, а також вбудовану підтримку API-маршрутів. Це дозволило реалізувати як інтерфейсну частину системи, так і серверну логіку в межах одного фреймворку, що суттєво спростило структуру проєкту та прискорило обробку запитів. Використання системи файлового роутингу Next.js полегшило створення сторінок, а

підтримка серверних компонентів забезпечила високу продуктивність та оптимізацію завантаження сторінок.

Клієнтська частина системи розроблена з використанням React — бібліотеки для побудови інтерфейсів користувача. У поєднанні з компонентною структурою та адаптивною версткою React забезпечив можливість створення інтерактивних, динамічних і повторно використовуваних елементів. Для стилізації застосовувався Tailwind CSS, що дозволило впровадити стильове оформлення високої якості без громіздких CSS-файлів. Tailwind CSS застосовує атомарний підхід до стилів, що забезпечує швидку розробку інтерфейсу, легкість підтримки та можливість масштабування дизайну системи.

Особливу увагу в проєкті було приділено організації та зберіганню даних. Для цього було використано MongoDB — документно-орієнтовану базу даних, яка забезпечує гнучкість структури, високу швидкість обробки даних і простоту масштабування. Для роботи з MongoDB застосовувалась бібліотека Mongoose, яка дозволяє створювати схеми моделей, виконувати валідацію даних та реалізовувати CRUD-операції відповідно до принципів об’єктно-документного відображення (ODM). Механізм кешування з’єднань, реалізований у спеціальному модулі, забезпечив стабільність роботи системи та ефективне використання ресурсів.

При створенні AI-функціональності, що генерує персональні тренувальні плани, було використано OpenAI API. Засоби штучного інтелекту дозволили інтегрувати генеративну модель у систему та забезпечити користувачів можливістю отримувати рекомендації на основі їх фізичних параметрів, тренувальних цілей та історії активності. Використання OpenAI вимагало організації серверної обробки запитів, шифрування ключів та створення безпечних API-ендпоїнтів, що повністю відповідає вимогам безпеки й архітектури сучасних веб-платформ.

Для тестування роботи інтерфейсу, API-маршрутів і підключення до бази даних використовувався вбудований термінал VS Code, а також

браузерні інструменти розробника. Це дозволило оперативно перевіряти коректність роботи компонентів, обробку серверних запитів та взаємодію з базою даних у реальному часі.

Таким чином, середовище розробки PowerGym включає комплекс сучасних інструментів — Visual Studio Code, Next.js, React, Tailwind CSS, Node.js, MongoDB, OpenAI API та GitHub — які дозволили створити продуктивну, модульну та масштабовану веб-систему. Використання цих засобів забезпечило зручність розробки, високу якість коду та можливість швидкого оновлення та розширення функціональності в майбутньому.

3.2. Реалізація модулів веб-додатка

Реалізація модулів веб-додатка PowerGym охоплює проміжок між інтерфейсною частиною системи, серверними запитами та роботою з базою даних, забезпечуючи повноцінний цикл взаємодії користувача з платформою. Веб-застосунок побудовано на основі фреймворку Next.js, що дозволило об’єднати логіку рендерингу сторінок, API-обробників та інтегрованих модулів у єдину архітектурну модель. Інтерфейс системи розроблено з використанням сучасних підходів адаптивного дизайну та компонентної структури, які забезпечують високу швидкість взаємодії та інтуїтивність використання для користувачів різного рівня технічної підготовки.

Основним елементом взаємодії є головна сторінка, що виконує презентаційну та навігаційну функції. На ній користувач знайомиться з ключовими можливостями PowerGym, включно з перевагами системи, доступом до тренувань, розкладу та AI-тренера. Яскравий промоблок (Hero-секція) та стилізовані картки переваг привертають увагу та формують привабливий образ платформи. Головна сторінка є логічним стартом роботи користувача, а її структура створює позитивне перше враження про систему.

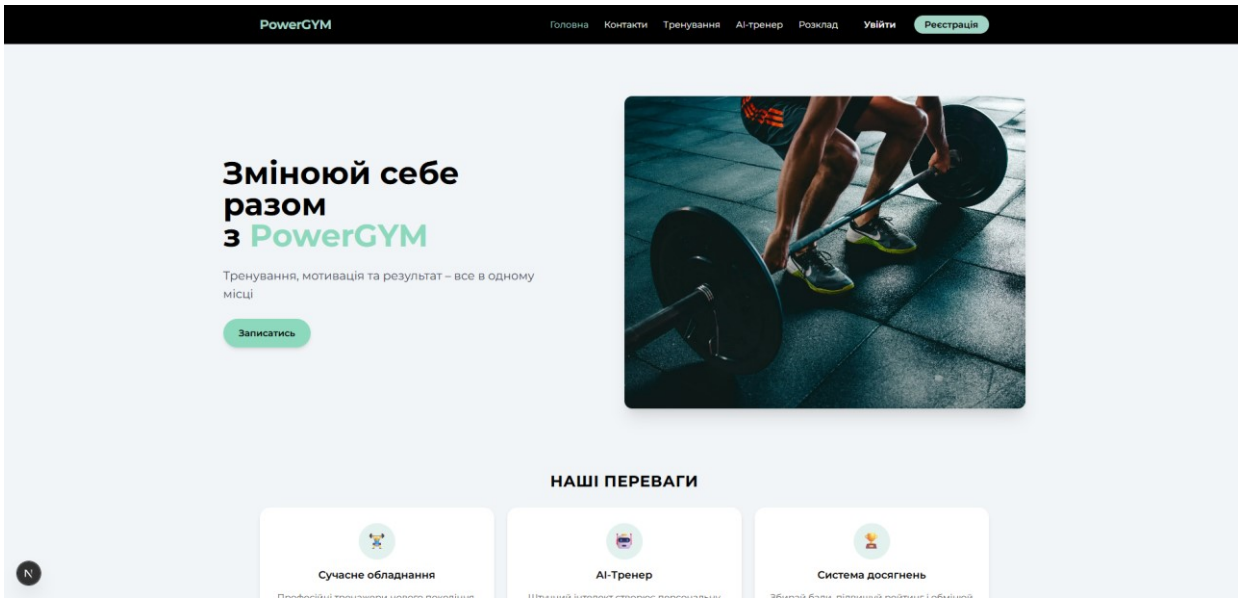

Рисунок 3.2.1 – Інтерфейс головної сторінки веб-додатка PowerGym

Важливим модулем є розділ «Тренування», який забезпечує доступ до всіх доступних форматів тренувальної активності спортклубу. Його завдання полягає в інформуванні користувачів про типи тренувань, рівень складності, доступність тренерів та можливість запису. Модуль реалізовано на основі власного API, що отримує дані з колекції «Training», а інтерфейс складається з компонента TrainingCard, який уніфіковано відображає кожну картку тренування. У випадку, коли база даних ще не містить жодного активного тренування, система коректно інформує користувача про відсутність даних.

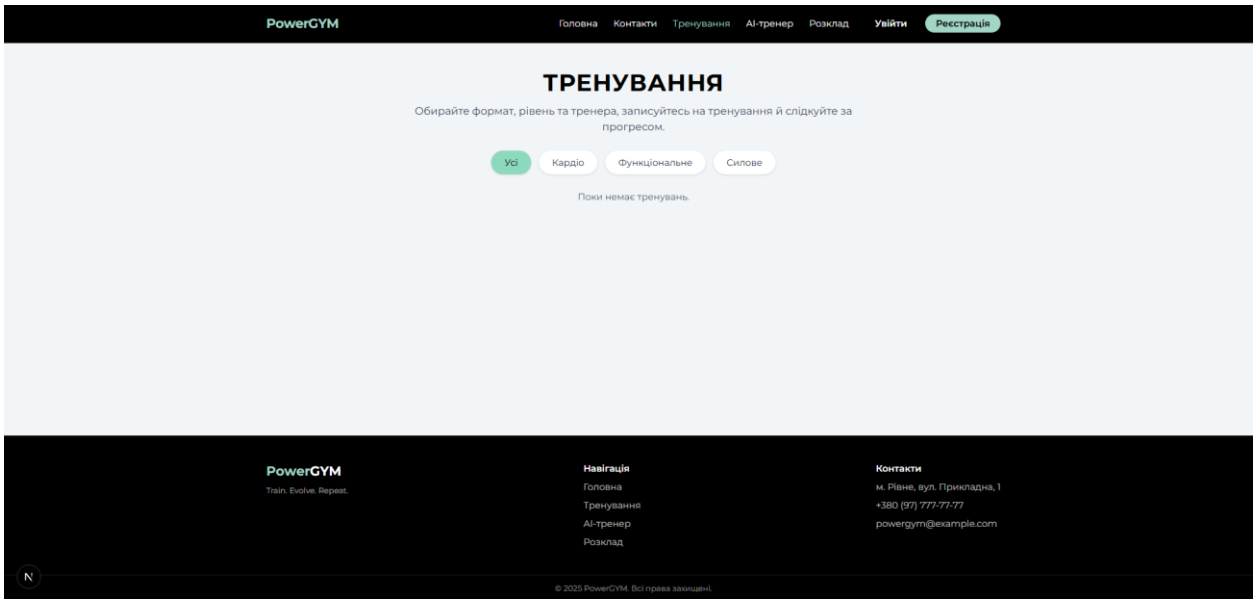

Рисунок 3.2.2 – Сторінка «Тренування» з відображенням доступних видів тренувань

Однією з найбільш інноваційних можливостей PowerGym є AI-тренер — інтелектуальний модуль, що формує персональні плани тренувань на основі параметрів користувача. На сторінці AI-тренера представлено форму введення віку, статі, маси тіла, цілей тренувального процесу, рівня підготовки та частоти тренувань. Після заповнення цих даних користувач активує процес генерації плану, який передає серверу запит для обробки через OpenAI API. Модуль повертає структурований текстовий план із вправами, підходами та рекомендаціями щодо прогресії. Особливістю системи є можливість збереження згенерованих планів у профіль користувача, що забезпечує довготривалу персоналізацію та взаємодію з AI.

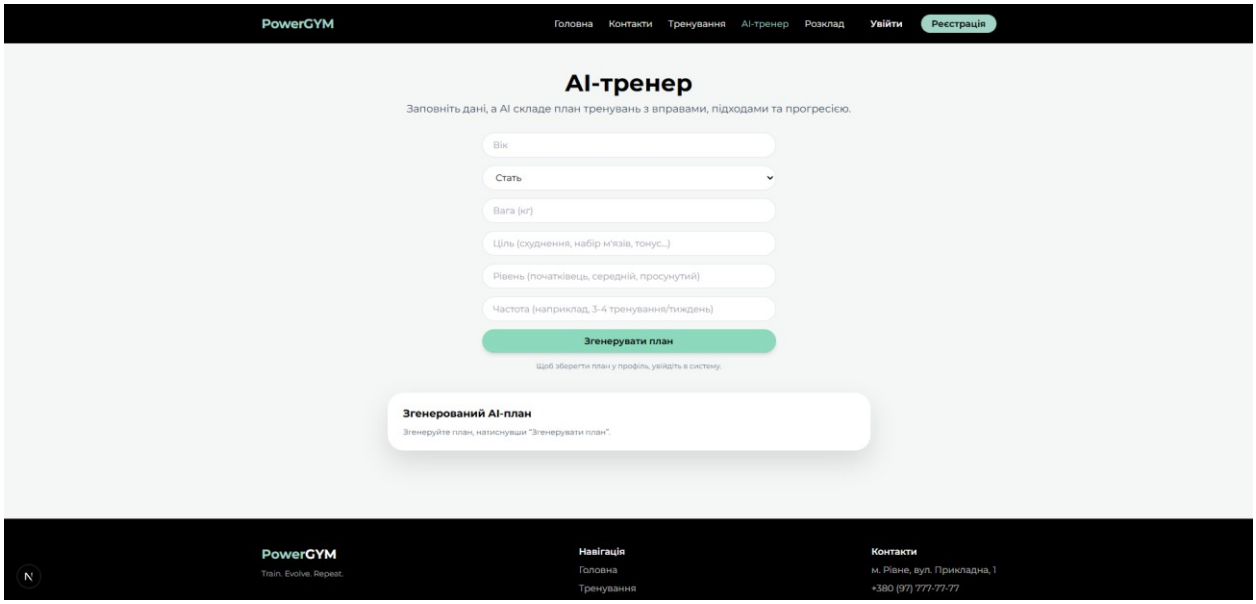

Рисунок 3.2.3 – Інтерфейс модуля «AI-тренер» для генерації персонального тренувального плану

Розділ «Розклад» відповідає за організацію відвідувань та аналіз вільних місць у спортивному залі. Він включає модуль завантаженості залу, який показує приблизну кількість відвідувачів у поточний момент, модуль лідерів мотиваційної системи та календар відвідувань, що стає доступним після авторизації. Усю динаміку модуля забезпечують API-запити, що працюють із моделями GymLoad, PointsLog та Booking. Завдяки цьому користувач отримує актуальну інформацію в реальному часі, а система забезпечує точність прогнозування та планування тренувального процесу.

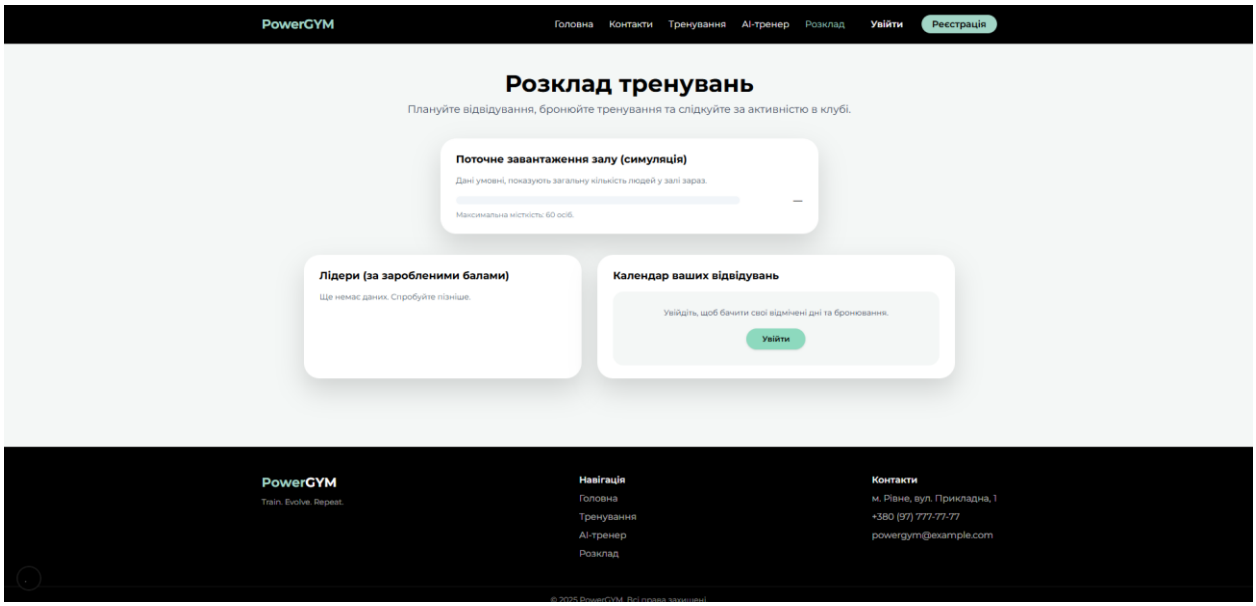

Рисунок 3.2.4 – Сторінка «Розклад» з відображенням завантаженості залу та календаря

Авторизація та реєстрація є обов’язковими елементами функціональності системи, адже вони забезпечують індивідуальний доступ користувачів до персональних даних, журналу відвідувань, мотиваційної системи та збережених AI-планів. Сторінки входу та реєстрації виконані в мінімалістичному стилі з акцентом на зручність користувача. Після успішної авторизації відкривається доступ до особистого кабінету, де зберігаються всі ключові показники активності користувача: оцінки, бали, персональні тренування, AI-плани та інші важливі елементи.

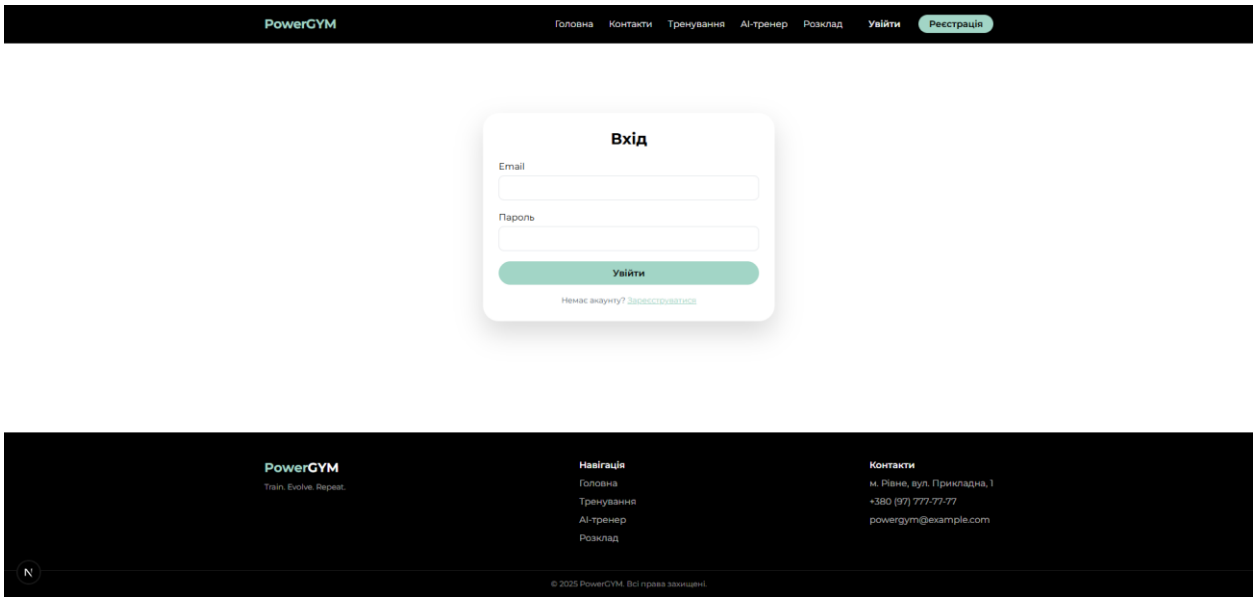

Рисунок 3.2.5 – Форма авторизації користувача у системі PowerGym

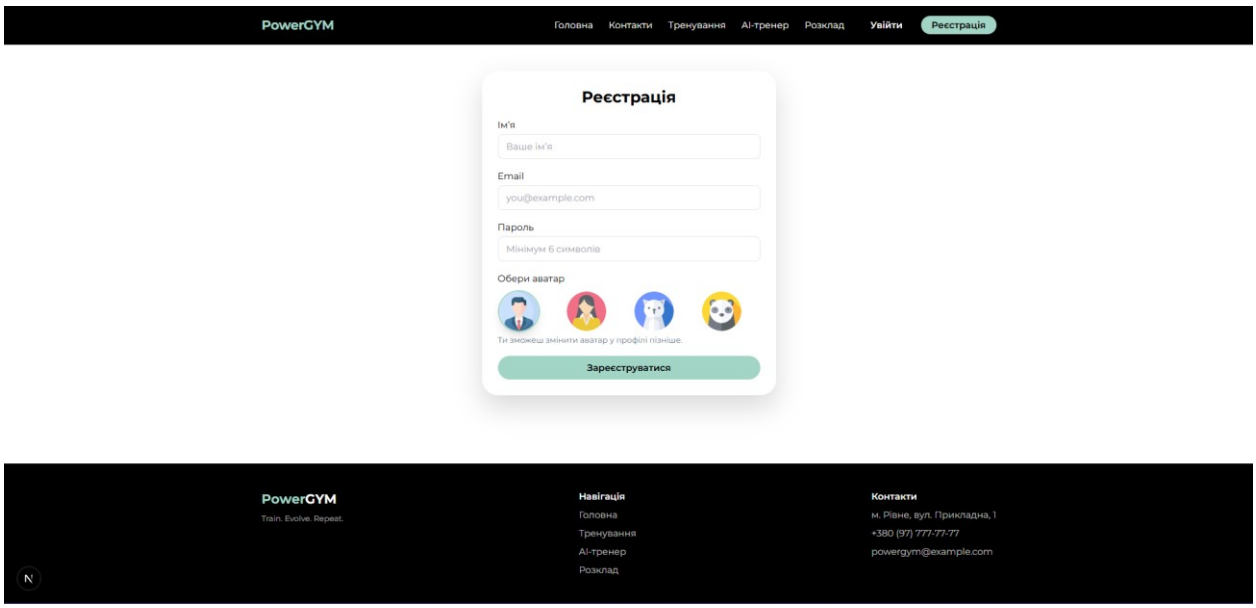

Рисунок 3.2.6 – Форма реєстрації нового користувача

Суттєвою частиною загального функціоналу є сторінка «Контакти та зворотний зв’язок», що надає користувачам можливість швидкого контакту з адміністрацією спортклубу. Вона містить інтегровану карту, довідкову інформацію та форму для надсилання повідомлень. Всі звернення зберігаються в колекції ContactMessage, що дозволяє працівникам оперативно опрацьовувати запити.

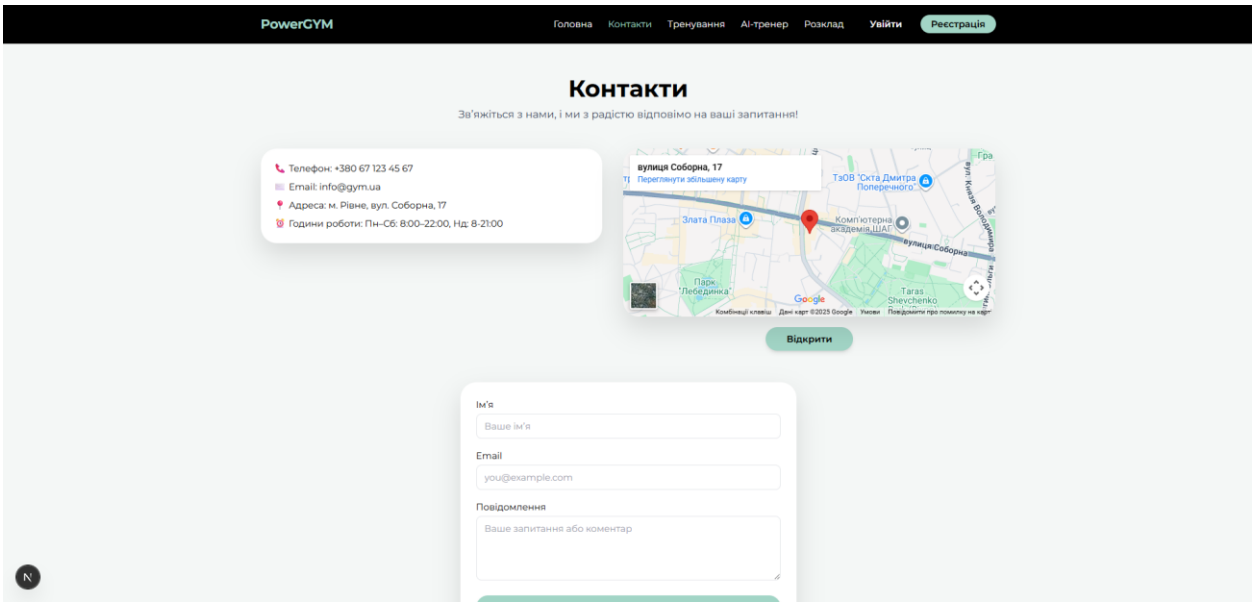

Рисунок 3.2.7 – Сторінка «Контакти» з інтегрованою картою та формою зворотного зв'язку

Веб-додаток PowerGym містить комплекс взаємопов'язаних модулів, що забезпечують повноцінний цикл взаємодії користувача із системою: від перегляду тренувань та роботи з AI-тренером — до планування розкладу, ведення персонального кабінету та комунікації з адміністрацією. Реалізація кожного модуля ґрунтується на сучасних технологічних підходах, дотриманні принципів зручності, надійності та масштабованості, що дозволяє PowerGym виступати ефективним інструментом управління тренувальним процесом та мотиваційною діяльністю спортивного клубу.

3.3. Реалізація авторизації та безпеки

Механізми авторизації та забезпечення безпеки користувацьких даних є ключовими складовими інформаційної системи PowerGym, оскільки платформа працює з персональними обліковими записами відвідувачів, їх тренувальною активністю, контактними даними та, частково, з конфіденційною інформацією щодо взаємодії з тренерами та AI-тренером. У процесі розроблення веб-додатка особливу увагу приділено організації процесу реєстрації та входу користувачів у систему, розмежуванню доступу до окремих розділів, а також способам збереження і захисту даних у базі MongoDB та на стороні сервера.

Авторизаційний функціонал у PowerGym реалізовано через окремі сторінки «Вхід» та «Реєстрація», що є складовими клієнтської частини Next.js-застосунку. Ці сторінки доступні з будь-якого розділу сайту завдяки елементам навігації у верхній панелі, що дозволяє користувачу швидко перейти до створення нового облікового запису або входу в уже існуючий профіль. На сторінці реєстрації передбачено введення основних облікових даних: імені, електронної адреси, пароля та вибору аватара. Сторінка входу містить поля для email та пароля, забезпечуючи мінімалістичний, але зрозумілий інтерфейс взаємодії.

Логіка обробки авторизаційних подій винесена на серверну сторону через API-маршрути, що розташовані в директорії api/users. При реєстрації дані, введені користувачем у форму, надсилаються HTTP-запитом на сервер, де проходять базову валідацію та підготовку перед збереженням у базі даних. Для представлення користувачів у MongoDB використовується модель User.ts, яка описує структуру документів колекції користувачів: електронну адресу, ім’я, пароль (у вигляді захищеного значення), роль користувача, облікові параметри мотиваційної системи, аватар та інші необхідні поля. Така структура дозволяє централізовано керувати всією інформацією про облікові записи й пов’язувати її з іншими сутностями, такими як історія тренувань, персональні плани, записи на заняття та набрані бали.

Особливу увагу при реалізації процесу реєстрації приділено збереженню пароля та використанню захищених каналів комунікації. Пароль, який вводить користувач, не зберігається у відкритому вигляді: перед збереженням у базі даних він обробляється на сервері та записується в зашифрованому або хешованому форматі. Це дозволяє унеможливити його використання у разі несанкціонованого доступу до бази даних. Під час входу в систему пароль, отриманий від користувача, порівнюється не з оригінальним значенням, а з раніше збереженим хешем за відповідним алгоритмом. Такий підхід відповідає загальноприйнятій практиці безпечного зберігання облікових даних у веб-додатках.

Додатковим елементом безпеки є використання змінних середовища, що зберігаються у файлі .env.local. У цьому файлі містяться конфіденційні параметри, такі як адреса підключення до MongoDB (MONGO_URI). Ці значення не потрапляють до репозиторію вихідного коду й не відображаються у клієнтській частині застосунку, що запобігає їх компрометації. Підключення до бази даних здійснюється через модуль lib/mongodb.ts, де реалізовано механізм кешування з'єднань. Це не лише підвищує продуктивність, а й зменшує ризики перевантаження бази неправильно ініційованими підключеннями, особливо у багатократних викликах серверних функцій Next.js.

Розмежування доступу до окремих модулів PowerGym також реалізовано на рівні логіки веб-додатка. Частина розділів, таких як перегляд AI-плану, календар відвідувань або мотиваційні досягнення, орієнтована на авторизованих користувачів, тому для повноцінної взаємодії із системою необхідно здійснити вхід. На інтерфейсному рівні це реалізовано через відображення кнопок «Увійти» або «Реєстрація», а також через відповідні повідомлення, що пропонують виконати авторизацію для доступу до персональних даних. На серверному рівні доступ до певних операцій контролюється через перевірку користувацьких даних у запиті: тільки автентифіковані користувачі мають змогу створювати бронювання занять, фіксувати набрані бали, зберігати AI-плани тощо.

Окрему роль у системі відіграє адміністратор, який має розширений набір повноважень і доступ до функцій, недоступних звичайному користувачеві. Зокрема, адміністратор може керувати списком тренувань, створювати події, додавати новини, реєструвати тренерів, редагувати розклад, нараховувати мотиваційні бали та переглядати статистику активності. Розмежування ролей реалізовано через поле ролі у моделі користувача та відповідні перевірки на стороні API-маршрутів, що дозволяє виявляти, чи є поточний користувач адміністратором, і відповідно обмежувати або розширювати його можливості. Тим самим забезпечується

логічне й безпечне управління всією платформою, де критичні операції доступні лише довіреним користувачам із відповідними правами.

Реалізація авторизації та безпеки в PowerGym ґрунтується на поєднанні кількох підходів: використання захищеного зберігання паролів, розподілу ролей користувачів, контролю доступу до критичних API-маршрутів, ізоляції конфіденційних налаштувань у змінних середовища, а також раціональної організації взаємодії з базою даних. Такий підхід дозволяє не лише забезпечити високий рівень безпеки персональних даних, але й створити гнучку основу для подальшого посилення захисту, наприклад, через додавання багатофакторної автентифікації, журнальної системи відстеження входів чи розширених механізмів моніторингу підозрілої активності.

3.4. Реалізація роботи з MongoDB через Mongoose

У системі PowerGym управління даними є одним із ключових аспектів функціонування, оскільки саме на базі постійно змінюваних і взаємопов’язаних даних здійснюється авторизація користувачів, формування персональних тренувальних планів, робота із записами на заняття, відображення новин, аналіз активності та завантаженості залу, а також адміністрування контенту. Для забезпечення ефективного й гнучкого зберігання інформації в проєкті використано документно-орієнтовану базу даних MongoDB, що відзначається високою продуктивністю, масштабованістю та зручністю роботи з напівструктурованими даними. Взаємодія застосунку з базою реалізована за допомогою бібліотеки Mongoose, яка є об’єктно-документним відображенням (ODM) і надає можливість визначати схеми, моделі, валідацію полів, зв’язки між документами та автоматичне створення часових міток. Завдяки цьому розробка набуває структурованого характеру, а робота з даними — безпечності та передбачуваності.

Підключення до MongoDB здійснюється через модуль lib/mongodb.ts, у якому реалізовано механізм кешування з’єднань. Особливість архітектури Next.js полягає в тому, що серверні функції можуть виконуватися повторно

під час кожного запиту, що потенційно може породжувати надмірну кількість з’єднань із базою. Щоб запобігти цьому, використовується глобальний кеш (global.mongooseCache), у якому зберігається посилання на активне з’єднання. Якщо воно вже існує — повторне з’єднання не створюється. Такий підхід оптимізує використання ресурсів, скорочує затримки та підвищує продуктивність роботи всієї системи. Параметри підключення, зокрема MONGO_URI, зберігаються у файлі .env.local, що гарантує конфіденційність та захист критично важливої інформації.

Одним із найважливіших елементів роботи з базою є система моделей, що визначають структуру даних та взаємозв’язки між сутностями. У проєкті передбачено кілька логічних груп моделей: моделі користувачів і тренерів, моделі тренувань і планів, моделі адміністративних даних та моделі активності користувачів. Ядром системи є модель користувача User, яка містить основні поля для ідентифікації, авторизації, мотиваційної системи та підписок. Структура моделі подана в таблиці 3.4.1.

Таблиця 3.4.1

Поле	**Тип**	**Опис**
email	String, required, unique, index	Унікальна адреса користувача
name	String	Ім’я користувача
password	String, required	Захешований пароль
role	String (user/admin)	Роль у системі
avatar	String	URL аватарки
points	Number	Нараховані бали мотивації
subscription	String (free/plus/premium)	Тип підписки
subscriptionUntil	Date	Строк дії підписки
timestamps	auto	Дати створення/оновлення

Для відображення інформації про тренерів застосовується модель Trainer, що містить ім’я, прізвище, аватар тренера, кількість відгуків та середній рейтинг. Вона використовується для прив’язки до тренувань і персональних програм. Її структуру наведено в таблиці 3.4.2.

Таблиця 3.4.2

Поле	Тип	Опис
firstName	String (required)	Ім’я тренера
lastName	String (required)	Прізвище
avatar	String (default URL)	Фото тренера
reviewsCount	Number (default: 0)	Кількість відгуків
ratingAverage	Number (default: 0)	Середній рейтинг

Особливе місце в системі посідає модель Training, що представляє тренування в структурі залу. Вона містить назву тренування, категорію, рівень складності, тривалість, опис, рейтинг, стартову дату, slug-посилання, вимоги до підписки, фото та прив’язку до тренера. Це одна з найбільш наповнених моделей, що взаємодіє з іншими сутностями, такими як Booking та Trainer. Структуру моделі наведено у таблиці 3.4.3.

Таблиця 3.4.3

Поле	Тип	Опис
title	String (required)	Назва тренування
category	String (enum: cardio, functional, strength)	Тип тренування
level	String (enum: beginner, intermediate, advanced)	Рівень складності
durationMin	Number (default: 45)	Тривалість
description	String	Опис
rating	Number (default: 0)	Рейтинг тренування
startAt	Date	Дата й час проведення
coach	String	Ім’я тренера (legacy поле)
slug	String (unique)	ЧПУ URL-ідентифікатор
minSubscription	String (enum free/plus/premium)	Мінімальний рівень абонементу
trainer	ObjectId (ref: Trainer)	Посилання на тренера
image	String	Зображення тренувальної картки

Важливою функцією PowerGym є створення персональних програм тренувань. На це спрямована модель AiPlan, що описує згенерований штучним інтелектом тренувальний план. Вона містить інформацію, введену

користувачем, та сам текст рекомендацій, сформований OpenAI API. Структура моделі подана у таблиці 3.4.4

Таблиця 3.4.4

Поле	Тип	Опис
user	ObjectId, ref: User (required)	Користувач, для якого створено план
age	Number	Вік
sex	String	Стать
goal	String	Ціль тренування
level	String	Рівень підготовки
frequency	String	Частота тренувань
text	String (required)	Згенерований план тренувань

Ще однією сутністю, що забезпечує персоналізацію тренувального процесу, є модель PersonalTraining. Вона використовується адміністратором для призначення тренувань конкретним користувачам та зберігає необхідну інформацію: тренера, користувача, план тренування, дату та статус заняття. Таблиця 3.4.5 містить перелік її полів.

Таблиця 3.4.5

Поле	Тип	Опис
user	ObjectId (ref: User)	Клієнт
trainer	ObjectId (ref: Trainer)	Призначений тренер
title	String	Назва
plan	String	Детальний опис тренування
date	Date	Дата проведення
status	String (default "planned")	Статус: planned/done/cancelled

Функцію запису на тренування виконує модель Booking, яка прив’язує користувача до певного тренування. Вона містить посилання на користувача та тренування, а також статус бронювання. Структуру моделі наведено в таблиці 3.4.6.

Таблиця 3.4.6

Поле	Тип	Опис
user	ObjectId (ref: User, required)	Хто записався
training	ObjectId (ref: Training, required)	На яке тренування
status	String (enum active/cancelled/completed)	Стан запису

Важливою складовою системи є мотиваційна програма. Для зберігання інформації про нараховані бали використовується модель PointsLog, що прив’язує запис до користувача. Структура наведена у таблиці 3.4.7.

Таблиця 3.4.7

Поле	Тип	Опис
user	ObjectId (ref: User, required)	Хто записався
training	ObjectId (ref: Training, required)	На яке тренування
status	String (enum active/cancelled/completed)	Стан запису

Система також містить контентний модуль новин, що працює з моделлю News. Вона містить заголовок, короткий і повний опис, зображення, теги та дату публікації. Структуру наведено у таблиці 3.4.8.

Таблиця 3.4.8

Поле	Тип	Опис
title	String (required)	Заголовок
shortDescription	String	Короткий опис
fullDescription	String	Повний текст
image	String	Зображення
tags	[String]	Теги
publishedAt	Date	Дата публікації

Для відображення поточної завантаженості залу використано модель GymLoad, що містить лише одне числове поле — кількість відвідувачів, а також часові мітки. Структура наведена в таблиці 3.4.9.

Таблиця 3.4.9

Поле	Тип	Опис
count	Number (default 0)	Орієнтовна кількість відвідувачів у залі

Кожна з наведених моделей інтегрована у відповідні API-маршрути, які забезпечують виконання CRUD-операцій: отримання даних, створення нових записів, редагування або видалення. Операції виконуються асинхронно, а Mongoose забезпечує валідацію, обробку помилок і гарантує відповідність даних схемам. Завдяки описаній структурі взаємодія з MongoDB у PowerGym

є прозорою, надійною та масштабованою, що відповідає вимогам сучасних інформаційних систем.

3.5. Тестування системи PowerGYM

Тестування інформаційної системи PowerGym проводилося з метою перевірки коректності реалізації функціоналу, стабільності роботи серверної логіки, інтеграції з базою даних MongoDB та відповідності поведінки інтерфейсу очікуваним вимогам. Оскільки PowerGym є комплексною платформою, що містить модулі для користувачів, тренерів та адміністратора, тестування охопило як клієнтську частину, так і роботу API, забезпечуючи повний аналіз надійності та узгодженості всіх компонентів системи.

Першим етапом було проведено функціональне тестування, яке полягало у перевірці роботи ключових сценаріїв взаємодії користувача з платформою. Одним із важливих напрямів тестування стала робота адмін-панелі, оскільки саме через цей модуль відбувається управління контентом і користувачами. На стартовому екрані адміністратора було перевірено коректність відображення доступних розділів, логіки маршрутизації та доступності функцій для роботи з тренерами, тренуваннями, новинами, повідомленнями та персональними тренуваннями.

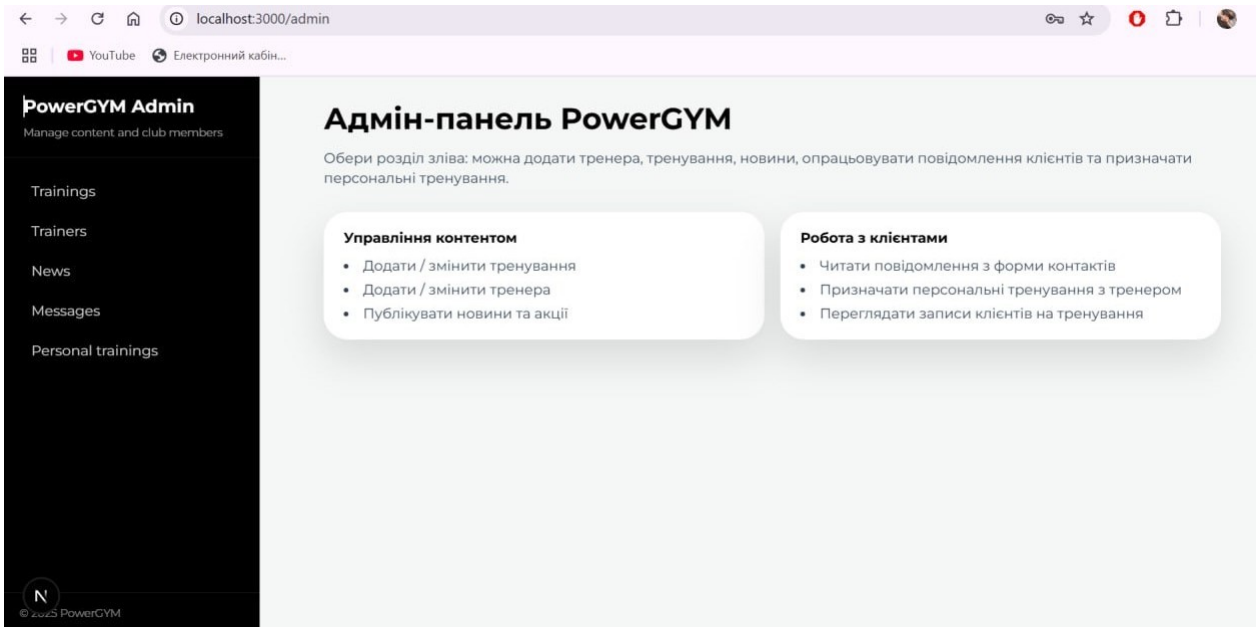

Рис.3.5.1. Головний інтерфейс адмін-панелі PowerGYM під час тестування функціоналу управління контентом

Функціональне тестування охопило модуль персональних тренувань, який є одним із найбільш важливих для адміністративної частини. Було перевірено заповнення форми, вибір тренера, введення email клієнта, створення назви та плану тренування, а також встановлення дати та часу. Окремо тестувалася система валідації, яка попереджає про неправильне введення даних або їхню відсутність. Також перевірено, що створене персональне тренування з'являється у списку без затримок та правильно записується до бази даних.

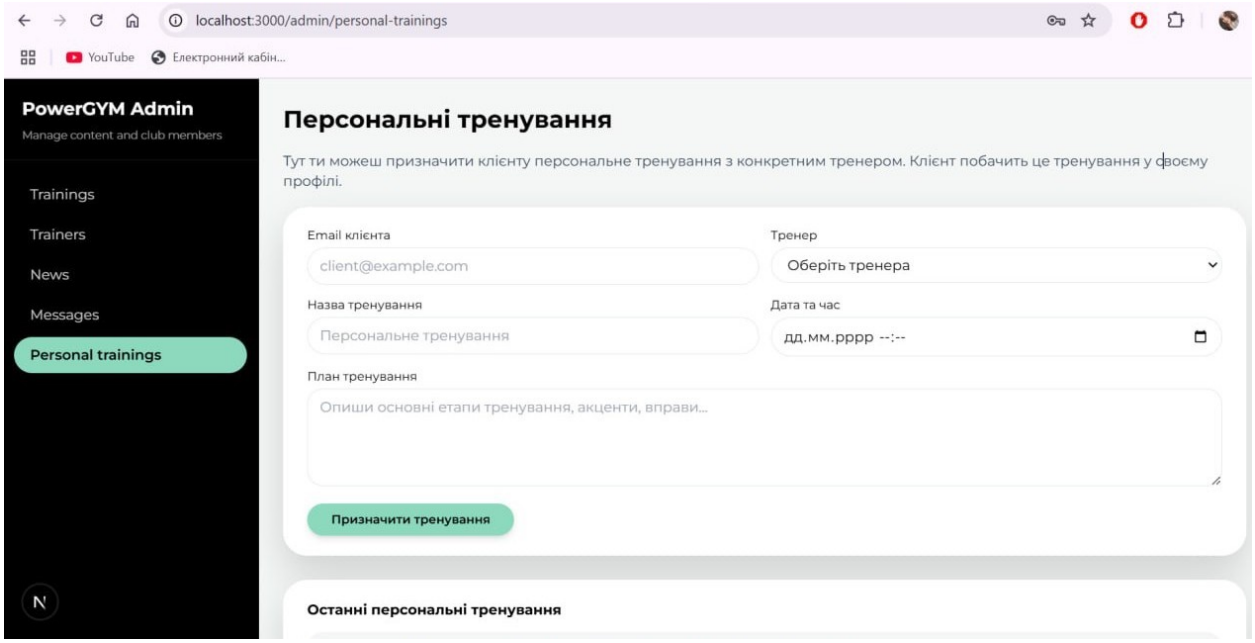

Рис.3.5.2. Тестування модуля призначення персональних тренувань адміністратором

Також значну увагу було приділено модулю створення тренувань, оскільки він формує основу тренувального розкладу для користувачів. У ході тестування перевірено коректність роботи усіх полів: назви, категорії, рівня складності, тривалості, мінімального рівня абонемента, вибору тренера та дати проведення. Тестування підтвердило, що система дозволяє створювати тренування з різними параметрами, зберігає їх у базі та відображає у загальному переліку тренувань на клієнтській частині.

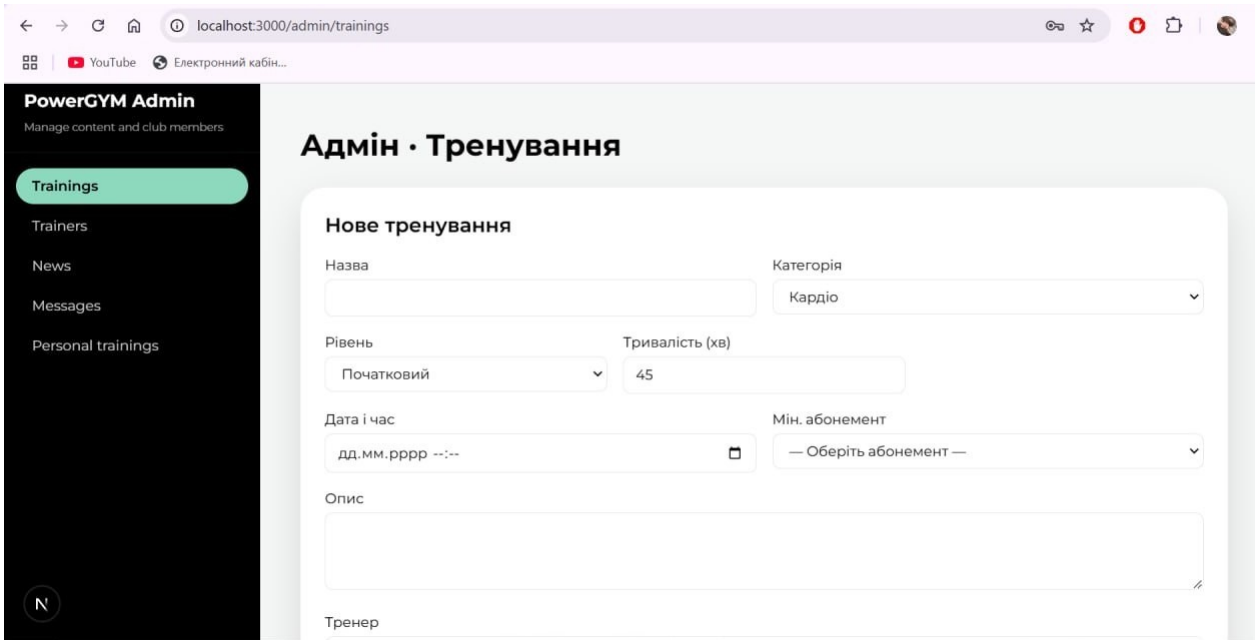

Рис.3.5.3. Перевірка роботи форми створення нового тренування в адмін-панелі

Користувацький модуль розкладу тренувань був протестований на предмет правильності обробки інформації про завантаженість залу, відображення рейтингу за набраними балами та роботи календаря. Було перевірено, що значення завантаженості оновлюються коректно, дані користувацького рейтингу відображаються точно, а календар працює без затримок. Під час тестування підтверджено, що користувач отримує доступ до особистих відвідувань лише після авторизації, що забезпечує належний рівень захисту даних.

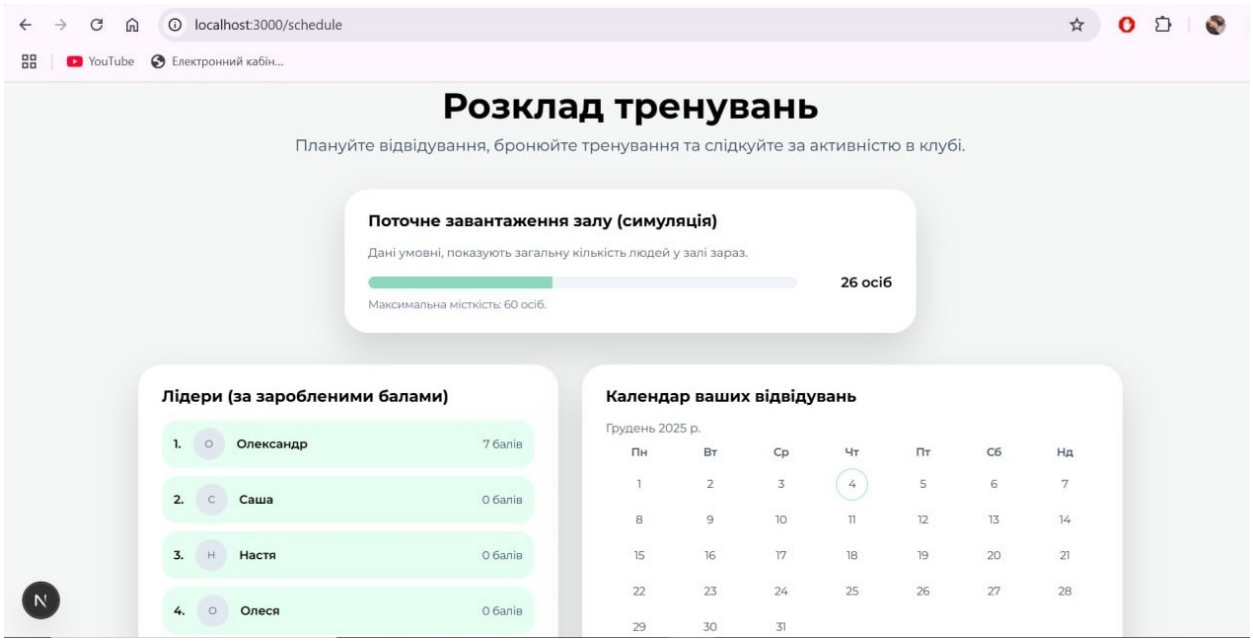

Рис.3.5.4. Тестування користувацького модуля “Розклад тренувань”

Окрема серія тестів була присвячена роботі модуля AI-тренера. У ході тестування перевірено формування персональних планів тренувань на основі введених користувачем параметрів, стабільність виконання запитів до API та обробку відповідей. Також проведено тестування збереження отриманого плану у профілі користувача. Система продемонструвала коректну взаємодію з базою даних: плани успішно зберігаються, відображаються в особистому кабінеті та можуть бути оновлені за потреби.

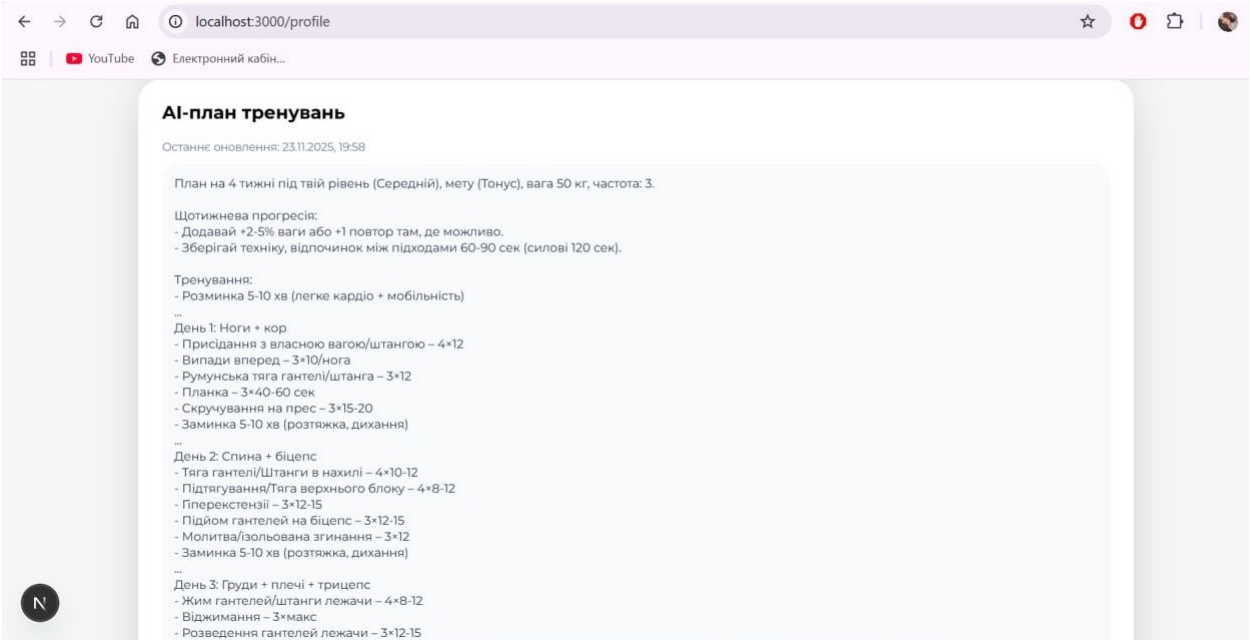

Рис.3.5.5. Результат тестування AI-тренера та відображення згенерованого плану в особистому кабінеті

Тестування продуктивності проводилось для визначення швидкодії системи за умови різних навантажень. Оскільки проєкт використовує Next.js та MongoDB з механізмом кешування з’єднань, система показала стабільно короткий час відповіді під час звернень до API. Було перевірено швидкість завантаження сторінок тренувань, розкладу та адміністративних модулів. Навіть при значному обсязі даних, таких як список записів на тренування чи множинні персональні програми, час обробки запиту не перевищував допустимих меж. Окремо було протестовано роботу фільтрації тренувань за категоріями (кардіо, силові, функціональні), де система показала коректну зміну стану інтерфейсу та швидке оновлення відображених даних.

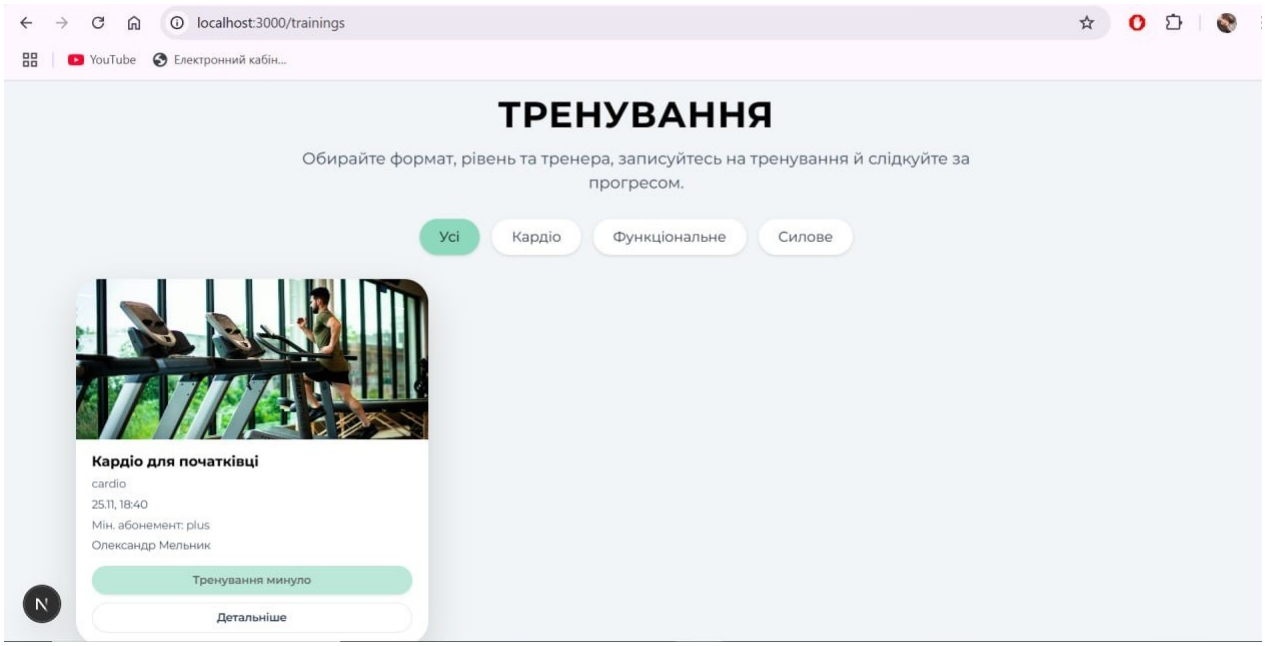

Рис.3.5.5. Перевірка роботи користувацького модуля «Тренування» та відображення списку доступних занять

Узагальнюючи результати проведених тестів, можна стверджувати, що система PowerGym пройшла всі етапи перевірки без критичних помилок. Модулі працюють узгоджено, сторінки завантажуються стабільно, а API забезпечує передбачувану й точну передачу даних. Усі функціональні частини — від авторизації до AI-тренера та адмін-панелі — відповідають вимогам технічного завдання та демонструють надійність у реальному використанні. Тестування підтвердило готовність веб-додатка до впровадження в умовах тренажерного залу або фітнес-центру та його ефективність як сучасного інструменту автоматизації тренувального процесу.

3.6. Оцінка ефективності використання системи

Оцінка ефективності інформаційної системи PowerGYM є завершальним етапом аналізу її працездатності, надійності та відповідності вимогам, які були сформульовані на початкових етапах проєктування й розроблення. У межах цього підрозділу проведено аналіз того, наскільки впроваджена система здатна вирішувати поставлені задачі щодо автоматизації тренувального процесу, підтримки комунікації з клієнтами та покращення управління спортивним закладом. Оцінювання ефективності включає аналіз технічних, організаційних та користувацьких аспектів, що

дозволяє визначити рівень доцільності застосування платформи PowerGYM у реальних умовах експлуатації.

Першим важливим критерієм оцінки ефективності є функціональна повнота системи. PowerGYM охоплює широкий спектр можливостей, що забезпечують автоматизацію діяльності тренажерного залу: від реєстрації користувачів та авторизації до формування персональних планів тренувань, управління розкладом, обробки повідомлень і створення новин. Усі ці компоненти працюють у єдиному інтерфейсі та взаємодіють між собою через централізовану базу даних MongoDB. Така інтеграція дозволяє мінімізувати дублювання інформації та забезпечує цілісність даних. Адміністративна панель надає можливість керувати тренерами, тренуваннями, новинами та персональними програмами, що значно знижує навантаження на персонал та усуває типові проблеми ручного ведення записів.

Другим ключовим аспектом є ефективність взаємодії користувача з платформою, що визначається зручністю інтерфейсу, швидкістю роботи та логічною структурою навігації. Користувач має змогу оперативно переглядати тренування, записуватися на заняття, стежити за власним рейтингом у мотиваційній системі, а також переглядати календар відвідувань. Особливу роль у підвищенні ефективності взаємодії відіграє модуль AI-тренера, який забезпечує автоматичне створення персоналізованих тренувальних планів. Завдяки цьому платформа не лише оптимізує роботу залу, але й підвищує якість обслуговування клієнтів, пропонуючи індивідуальний підхід без необхідності постійної участі тренера.

Окремої уваги заслуговує оцінка продуктивності та технічної стійкості системи. Використання архітектури Next.js у поєднанні з механізмом кешування з’єднань Mongoose дозволило досягти високої швидкодії при роботі з базою даних. Проведене тестування підтвердило, що система стабільно опрацьовує численні запити без затримок та з мінімальним навантаженням на сервер. Навіть модуль AI-тренера, який виконує зовнішні запити до OpenAI API, функціонує в межах очікуваних часових параметрів,

що не впливає на загальну продуктивність платформи. Таким чином, система демонструє високу технічну ефективність та може бути масштабована у разі збільшення кількості користувачів.

Ефективність системи оцінювалася також з точки зору адміністративного управління. Інтерфейс адміністратора дозволяє швидко створювати тренування, керувати тренерським складом, публікувати новини, переглядати повідомлення клієнтів та створювати індивідуальні тренувальні програми. Це суттєво знижує потребу у використанні сторонніх інструментів або веденні ручних записів, а також мінімізує людські помилки. Застосунок забезпечує структурування інформації у зручному вигляді, що значно покращує внутрішню організацію роботи тренажерного залу. З огляду на це система PowerGYM може бути впевнено рекомендована як інструмент централізованого управління діяльністю спортивного клубу.

Важливою складовою загальної оцінки є якість обробки та збереження даних. Використання MongoDB як нереляційної бази даних дозволяє гнучко працювати з різними типами інформації: структурованими (профіль користувача, персональні тренування), напівструктурованими (плани AI-тренера) та динамічними (повідомлення, новини). Завдяки моделям даних на основі Mongoose забезпечується чітка структура кожної колекції та валідація введених даних. Це підвищує рівень безпеки, знижує ризики виникнення помилок під час запису і забезпечує стійкість системи до некоректних даних.

З точки зору загальної користі та доцільності впровадження, PowerGYM демонструє значні переваги у порівнянні з традиційними паперовими методами ведення записів або застосуванням розрізнених програмних продуктів. Система дозволяє суттєво зменшити час на обробку заявок, автоматизувати формування персональних програм, спростити роботу адміністративного персоналу та підвищити комфорт для користувачів. Таким чином, підвищується рівень сервісу, зростає залученість клієнтів і створюються умови для ефективного розвитку тренажерного залу.

Ефективність використання інформаційної системи PowerGYM підтверджується її функціональною повнотою, зручністю інтерфейсу, стабільністю роботи, продуктивністю, інтеграцією з сучасними технологіями та здатністю автоматизувати ключові процеси тренувальної діяльності. Система повністю відповідає поставленим вимогам та демонструє високий потенціал для подальшого розширення й масштабування.

ВИСНОВКИ

У магістерській роботі було виконано повний цикл проєктування, розроблення та оцінювання ефективності інформаційної системи PowerGYM, призначеної для автоматизації тренувального процесу та управління діяльністю тренажерного залу. Реалізована система поєднує сучасні веб-технології, засоби роботи з базами даних та елементи штучного інтелекту, що дозволило сформувати функціональний та інноваційний продукт, здатний суттєво покращити взаємодію між клієнтами, тренерами та адміністрацією спортивного закладу.

У процесі роботи було здійснено аналіз предметної області, визначено вимоги до системи, її функціональні можливості та користувацькі сценарії. На основі проведеного аналізу сформовано архітектуру веб-додатка, яка включає модулі авторизації користувачів, управління тренерами та тренуваннями, систему записів на заняття, модуль AI-тренера для генерації персоналізованих тренувальних планів, мотиваційну систему нарахування балів, інтерфейс адміністратора та модуль контактів. Важливим елементом став серверний рівень, реалізований за допомогою Next.js API Routes, що забезпечив ефективну взаємодію з базою даних MongoDB через Mongoose.

Особливу увагу було приділено проєктуванню моделей даних, оскільки PowerGYM оперує різними типами інформації: персональними даними користувачів, тренерів, тренувальними програмами, AI-планами, записами на тренування та мотиваційними балами. Реалізація моделей у Mongoose забезпечила цілісність даних, їхню валідацію та логічну структуру. Архітектура системи дозволяє легко масштабувати функціонал та інтегрувати нові модулі, що підтверджує гнучкість та перспективність обраних технологічних рішень.

У ході тестування системи було підтверджено коректність роботи всіх модулів, стабільність серверної частини та ефективність взаємодії з базою даних. Тестування охопило ключові аспекти: функціональність,

продуктивність, обробку користувацьких сценаріїв, формування AI-планів, роботу адміністративної панелі та контроль доступу. Результати тестів засвідчили, що система працює узгоджено, забезпечує високу швидкодію, а користувацький інтерфейс є інтуїтивно зрозумілим та зручним. Перевірка продуктивності підтвердила, що PowerGYM може стабільно працювати навіть у випадку значного навантаження, а реалізовані механізми кешування та оптимізації забезпечують мінімальний час відгуку.

Оцінка ефективності системи показала, що впровадження PowerGYM дозволяє суттєво покращити якість організації тренувального процесу, оптимізувати управлінські завдання та посилити взаємодію з клієнтами. Адміністративний персонал отримує зручні інструменти для роботи з даними, тренери — можливість формувати індивідуальні програми, а користувачі — доступ до персоналізованих рекомендацій та аналітики власної активності. Завдяки поєднанню традиційних веб-технологій та інтелектуальних можливостей AI-тренера система демонструє високий рівень інноваційності та практичної цінності.

Розроблена інформаційна система PowerGYM повністю відповідає поставленим у роботі вимогам, реалізує всі заплановані модулі та підтверджує свою ефективність на етапі тестування. Робота має як теоретичну, так і практичну значущість, оскільки демонструє сучасний підхід до автоматизації фітнес-процесів та може бути безпосередньо впроваджена в діяльність спортивних клубів. Розроблена система може слугувати основою для подальшого розвитку та удосконалення, зокрема шляхом додавання мобільного застосунку, модулів аналітики, інтеграції із системами оплати та розширення можливостей AI-тренера.

СПИСОК ЛІТЕРАТУРИ

1. IT і бізнес: як технології впливають на розвиток сучасних підприємств: веб-сайт. URL: https://lemon.school/blog/it-i-biznes-yak-tehnologiyi-vplyvayut-na-rozvytok-suchasnyh-pidpryyemstv (дата звернення: 03.10.2025).

2. Технології в бізнесі – сучасні рішення оптимізації процесів: веб-сайт. URL: https://vgoru.org/cikavo/texnologiyi-v-biznesi-sucasni-risennia-optimizaciyi-procesiv (дата звернення: 07.10.2025).

3. Роль IT у розвитку малого та середнього бізнесу в Україні: веб-сайт. URL: https://www.run-it.com.ua/rol-it-u-rozvytku-maloho-ta-serednoho-biznesu-v-ukraini/ (дата звернення: 11.10.2025).

4. MongoDB Documentation: веб-сайт. URL: https://www.mongodb.com/docs (дата звернення: 15.10.2025).

5. Mongoose ODM Documentation: веб-сайт. URL: https://mongoosejs.com/docs (дата звернення: 18.10.2025).

6. Next.js Documentation: веб-сайт. URL: https://nextjs.org/docs (дата звернення: 22.10.2025).

7. OpenAI API Documentation: веб-сайт. URL: https://platform.openai.com/docs (дата звернення: 26.10.2025).

8. Google Developers – Web Application Performance: веб-сайт. URL: https://developers.google.com/web (дата звернення: 30.10.2025).

9. What makes technology crucial for achieving success in business endeavors?: веб-сайт. URL: https://www.easy.bi/blog/what-makes-technology-crucial-for-achieving-success-in-business-endeavors (дата звернення: 12.11.2025).

10. The Role of Information Technology in Business Growth: веб-сайт. URL: https://technuts.com/the-role-of-information-technology-in-business-growth/ (дата звернення: 27.11.2025).

ДОДАТКИ

Додаток А

Powergym/app/admin/login/page.tsx

```
"use client";

import { useState } from "react";
import { useRouter } from "next/navigation";
import Navbar from "@/components/Navbar";
import Footer from "@/components/Footer";
export default function AdminLoginPage() {
  const router = useRouter();
  const [login, setLogin] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState<string | null>(null);
  async function handleSubmit(e: React.FormEvent) {
    e.preventDefault();
    setError(null);
    try {
      const res = await fetch("/api/auth/login", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ email: login, password }),
      });

      const data = await res.json();

      if (!res.ok) {
        setError(data?.error || "Невірний логін або пароль");
        return;
      }

      if (data.role !== "admin") {
        setError("Цей акаунт не має прав адміністратора");
        return;
      }

      localStorage.setItem("powergymUser", JSON.stringify(data));
      router.push("/admin/news");
    } catch (err) {
      console.error(err);
      setError("Сталася помилка, спробуйте ще раз");
    }
  }

  return (
    <main className="min-h-screen bg-light text-dark flex flex-col">
      <Navbar />

      <div className="flex-1 flex items-center justify-center px-4 py-10">
        <form
          onSubmit={handleSubmit}
          className="w-full max-w-md rounded-3xl bg-white p-6 shadow-
[0_18px_40px_rgba(0,0,0,0.15)] space-y-4"
        >
          <h1 className="text-2xl font-extrabold text-center mb-2">
            Вхід в адмін-кабінет
          </h1>

          <div>
            <label className="block text-sm font-medium mb-1">Email</label>
            <input
              type="email"
              value={login}
              onChange={(e) => setLogin(e.target.value)}
```

```
              className="w-full rounded-lg border px-3 py-2 text-sm"
              placeholder="admin@example.com"
              required
            />
          </div>

          <div>
            <label className="block text-sm font-medium mb-1">Пароль</label>
            <input
              type="password"
              value={password}
              onChange={(e) => setPassword(e.target.value)}
              className="w-full rounded-lg border px-3 py-2 text-sm"
              placeholder="••••••••"
              required
            />
          </div>

          {error && (
            <p className="text-xs text-red-600 bg-red-50 rounded-lg px-3 py-2">
              {error}
            </p>
          )}
          <button
            type="submit"
            className="w-full rounded-full bg-primary px-4 py-2 text-sm font-semibold text-
black hover:bg-primary/80"
          >
            Увійти як адмін
          </button>
        </form>
      </div>
      <Footer />
    </main>
  );
}
```

Додаток Б

Powergym/app/admin/personal-training/page.tsx

```tsx
"use client";

import { useEffect, useState } from "react";

type TrainerOption = {
  id: string;
  name: string;
};

type PersonalTrainingItem = {
  id: string;
  title: string;
  date?: string;
  user?: { name: string; email: string } | null;
  trainer?: { name: string } | null;
  status?: string;
};

export default function AdminPersonalTrainingsPage() {
  const [trainers, setTrainers] = useState<TrainerOption[]>([]);
  const [list, setList] = useState<PersonalTrainingItem[]>([]);
  const [loading, setLoading] = useState(true);
  const [form, setForm] = useState({
    userEmail: "",
    trainerId: "",
    title: "",
    plan: "",
    date: "",
  });
  const [saving, setSaving] = useState(false);
  const [error, setError] = useState<string | null>(null);
  const [info, setInfo] = useState<string | null>(null);

  useEffect(() => {
    // тренери
    fetch("/api/trainers")
      .then(async (res) => (res.ok ? res.json() : []))
      .then((list: any[]) => {
        const mapped: TrainerOption[] = list.map((t) => ({
          id: String(t._id ?? t.id),
          name: `${t.firstName} ${t.lastName}`,
        }));
        setTrainers(mapped);
      })
      .catch((e) => console.error("Error fetching trainers:", e));

    // список персональних тренувань (усі)
    fetch("/api/personal-trainings")
      .then(async (res) => (res.ok ? res.json() : []))
      .then((list: any[]) => {
        const mapped: PersonalTrainingItem[] = list.map((pt) => ({
          id: String(pt.id ?? pt._id),
          title: pt.title,
          date: pt.date,
          user: pt.user,
          trainer: pt.trainer,
          status: pt.status,
        }));
        setList(mapped);
      })
      .catch((e) =>
        console.error("Error fetching personal trainings:", e)
      )
```

```
    .finally(() => setLoading(false));
}, []);

const handleChange = (
  e: React.ChangeEvent<HTMLInputElement | HTMLSelectElement | HTMLTextAreaElement>
) => {
  const { name, value } = e.target;
  setForm((prev) => ({ ...prev, [name]: value }));
};

const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  setError(null);
  setInfo(null);

  if (
    !form.userEmail ||
    !form.trainerId ||
    !form.title ||
    !form.plan ||
    !form.date
  ) {
    setError("Заповни всі поля форми.");
    return;
  }

  setSaving(true);
  try {
    const res = await fetch("/api/personal-trainings", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(form),
    });

    const data = await res.json();
    if (!res.ok) {
      setError(data?.error || "Не вдалося створити тренування.");
    } else {
      setInfo("Персональне тренування призначено ✅");
      // оновити список (мінімально)
      setList((prev) => [
        {
          id: data.id,
          title: form.title,
          date: form.date,
          user: { name: "", email: form.userEmail },
          trainer: {
            name:
              trainers.find((t) => t.id === form.trainerId)?.name ??
              "Тренер",
          },
        },
        ...prev,
      ]);
      setForm({
        userEmail: "",
        trainerId: "",
        title: "",
        plan: "",
        date: "",
      });
    }
  } catch (e) {
    console.error(e);
    setError("Сталася помилка. Спробуй ще раз.");
  } finally {
    setSaving(false);
  }
```

```
  };

  return (
    <div className="max-w-5xl mx-auto">
      <h1 className="text-xl md:text-2xl font-extrabold mb-4">
        Персональні тренування
      </h1>

      <p className="text-sm text-slate-700 mb-4">
        Тут ти можеш призначити клієнту персональне тренування з конкретним
        тренером. Клієнт побачить це тренування у своєму профілі.
      </p>

      {/* Форма */}
      <form
        onSubmit={handleSubmit}
        className="rounded-3xl bg-white shadow-[0_18px_40px_rgba(0,0,0,0.15)] px-6 py-5 mb-6
space-y-3 text-sm"
      >
        <div className="grid gap-3 md:grid-cols-2">
          <div>
            <label className="block text-xs mb-1">Email клієнта</label>
            <input
              name="userEmail"
              value={form.userEmail}
              onChange={handleChange}
              placeholder="client@example.com"
              className="w-full rounded-full border border-slate-200 px-3 py-2 text-sm
outline-none focus:border-[#8DD9BE]"
            />
          </div>
          <div>
            <label className="block text-xs mb-1">Тренер</label>
            <select
              name="trainerId"
              value={form.trainerId}
              onChange={handleChange}
              className="w-full rounded-full border border-slate-200 px-3 py-2 text-sm
outline-none focus:border-[#8DD9BE] bg-white"
            >
              <option value="">Оберіть тренера</option>
              {trainers.map((t) => (
                <option key={t.id} value={t.id}>
                  {t.name}
                </option>
              ))}
            </select>
          </div>
          <div>
            <label className="block text-xs mb-1">Назва тренування</label>
            <input
              name="title"
              value={form.title}
              onChange={handleChange}
              placeholder="Персональне тренування"
              className="w-full rounded-full border border-slate-200 px-3 py-2 text-sm
outline-none focus:border-[#8DD9BE]"
            />
          </div>
          <div>
            <label className="block text-xs mb-1">Дата та час</label>
            <input
              type="datetime-local"
              name="date"
              value={form.date}
              onChange={handleChange}
              className="w-full rounded-full border border-slate-200 px-3 py-2 text-sm
outline-none focus:border-[#8DD9BE]"
```

```
            />
          </div>
        </div>

        <div>
          <label className="block text-xs mb-1">План тренування</label>
          <textarea
            name="plan"
            value={form.plan}
            onChange={handleChange}
            rows={4}
            placeholder="Опиши основні етапи тренування, акценти, вправи..."
            className="w-full rounded-2xl border border-slate-200 px-3 py-2 text-sm outline-
none focus:border-[#8DD9BE] resize-y"
          />
        </div>

        {error && (
          <p className="text-xs text-red-600">
            {error}
          </p>
        )}
        {info && (
          <p className="text-xs text-emerald-700">
            {info}
          </p>
        )}

        <button
          type="submit"
          disabled={saving}
          className="rounded-full bg-[#8DD9BE] px-6 py-2 text-xs font-semibold text-black
shadow-md hover:bg-[#7ACDAE] disabled:opacity-60"
        >
          {saving ? "Зберігаємо..." : "Призначити тренування"}
        </button>
      </form>

      {/* Список існуючих персональних тренувань */}
      <div className="rounded-3xl bg-white shadow-[0_18px_40px_rgba(0,0,0,0.15)] px-6 py-5
text-sm">
        <h2 className="font-extrabold mb-3">Останні персональні тренування</h2>

        {loading ? (
          <p className="text-xs text-slate-600">Завантаження...</p>
        ) : list.length === 0 ? (
          <p className="text-xs text-slate-600">
            Персональних тренувань поки що немає.
          </p>
        ) : (
          <div className="space-y-2">
            {list.map((pt) => (
              <div
                key={pt.id}
                className="rounded-2xl bg-[#F4F7F6] px-4 py-3 flex flex-col md:flex-row
md:items-center md:justify-between gap-1"
              >
                <div>
                  <div className="font-semibold">{pt.title}</div>
                  <div className="text-[11px] text-slate-600">
                    Клієнт: {pt.user?.name || pt.user?.email || "—"}{" "}
                    {pt.user?.email && `(${pt.user.email})`}
                    <br />
                    Тренер: {pt.trainer?.name || "—"}
                  </div>
                </div>
                <div className="text-right text-[11px] text-slate-600">
                  {pt.date &&
```

```
                    new Date(pt.date).toLocaleString("uk-UA", {
                      day: "2-digit",
                      month: "2-digit",
                      year: "numeric",
                      hour: "2-digit",
                      minute: "2-digit",
                    })}
                  {pt.status && (
                    <div className="mt-1 text-xs">
                      Статус: <span className="font-semibold">{pt.status}</span>
                    </div>
                  )}
                </div>
              </div>
            ))}
          </div>
        )}
      </div>
    </div>
  );
}
```

Додаток В

Powergym/app/admin/simulation/page.tsx

```
"use client";

import { useEffect, useState } from "react";

type UserItem = {
  id: string;
  name: string;
  email: string;
  points?: number;
};

export default function AdminSimulationPage() {
  const [gymCount, setGymCount] = useState<number>(0);
  const [gymMsg, setGymMsg] = useState<string | null>(null);
  const [users, setUsers] = useState<UserItem[]>([]);
  const [selectedUser, setSelectedUser] = useState<string>("");
  const [pointsToAdd, setPointsToAdd] = useState<number>(0);
  const [pointsDate, setPointsDate] = useState<string>(() => new Date().toISOString().slice(0,
10));
  const [pointsMsg, setPointsMsg] = useState<string | null>(null);

  useEffect(() => {
    fetch("/api/gym-load")
      .then((res) => (res.ok ? res.json() : { count: 0 }))
      .then((data) => setGymCount(data.count ?? 0))
      .catch(() => setGymCount(0));

    fetch("/api/users")
      .then((res) => (res.ok ? res.json() : []))
      .then((list: any[]) => {
        setUsers(
          list.map((u) => ({
            id: u.id,
            name: u.name || u.email,
            email: u.email,
            points: u.points,
          }))
        );
      })
      .catch(() => setUsers([]));
  }, []);

  const handleGymSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setGymMsg(null);
    const res = await fetch("/api/gym-load", {
      method: "PATCH",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ count: gymCount }),
    });
    setGymMsg(res.ok ? "Оновлено" : "Помилка оновлення");
  };

  const handlePointsSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setPointsMsg(null);
    if (!selectedUser) {
      setPointsMsg("Оберіть користувача");
      return;
    }
    const res = await fetch(`/api/users/${selectedUser}`, {
      method: "PATCH",
      headers: { "Content-Type": "application/json" },
```

```
      body: JSON.stringify({ pointsToAdd, date: pointsDate }),
    });
    setPointsMsg(res.ok ? "Бали нараховано" : "Помилка нарахування");
    if (res.ok) setPointsToAdd(0);
  };
  return (
    <div className="max-w-4xl mx-auto space-y-8">
      <h1 className="text-3xl font-extrabold">Симуляція</h1>

      <section className="rounded-3xl bg-white p-5 shadow border border-black/5">
        <h2 className="text-xl font-semibold mb-3">Завантаження залу</h2>
        <form onSubmit={handleGymSubmit} className="space-y-3">
          <div>
            <label className="block text-sm font-medium mb-1">Кількість людей</label>
            <input
              type="number"
              value={gymCount}
              onChange={(e) => setGymCount(Number(e.target.value) || 0)}
              className="w-full rounded-lg border px-3 py-2 text-sm"
            />
          </div>
          <button
            type="submit"
            className="rounded-full bg-primary px-5 py-2 text-sm font-semibold text-black
hover:bg-primary/80"
          >
            Оновити
          </button>
          {gymMsg && <p className="text-xs text-slate-600">{gymMsg}</p>}
        </form>
      </section>

      <section className="rounded-3xl bg-white p-5 shadow border border-black/5">
        <h2 className="text-xl font-semibold mb-3">Нарахувати бали</h2>
        <form onSubmit={handlePointsSubmit} className="space-y-3">
          <div>
            <label className="block text-sm font-medium mb-1">Користувач</label>
            <select
              value={selectedUser}
              onChange={(e) => setSelectedUser(e.target.value)}
              className="w-full rounded-lg border px-3 py-2 text-sm"
            >
              <option value="">— Оберіть користувача —</option>
              {users.map((u) => (
                <option key={u.id} value={u.id}>
                  {u.name} ({u.email})
                </option>
              ))}
            </select>
          </div>
          <div className="grid gap-3 md:grid-cols-2">
            <div>
              <label className="block text-sm font-medium mb-1">Скільки балів додати</label>
              <input
                type="number"
                value={pointsToAdd}
                onChange={(e) => setPointsToAdd(Number(e.target.value) || 0)}
                className="w-full rounded-lg border px-3 py-2 text-sm"
              />
            </div>
            <div>
              <label className="block text-sm font-medium mb-1">Дата зарахування</label>
              <input
                type="date"
                value={pointsDate}
                onChange={(e) => setPointsDate(e.target.value)}
                className="w-full rounded-lg border px-3 py-2 text-sm"
              />
```

```
            </div>
          </div>
          <button
            type="submit"
            className="rounded-full bg-primary px-5 py-2 text-sm font-semibold text-black
hover:bg-primary/80"
          >
            Нарахувати
          </button>
          {pointsMsg && <p className="text-xs text-slate-600">{pointsMsg}</p>}
        </form>
      </section>
    </div>
  );
}
```

Додаток Д

Powergym/app/admin/trainings/page.tsx

```tsx
"use client";

import { useEffect, useState } from "react";

type Trainer = {
  id: string;
  firstName: string;
  lastName: string;
};

type Training = {
  id: string;
  title: string;
  category: string;
  level: string;
  durationMin: number;
  description?: string;
  image?: string;
  coach?: string;
  startAt?: string;
  trainerId?: string;
  minSubscription?: "free" | "plus" | "premium";
};

const initialForm: Omit<Training, "id"> = {
  title: "",
  category: "cardio",
  level: "beginner",
  durationMin: 45,
  description: "",
  trainerId: "",
  startAt: "",
  minSubscription: "" as "" | "free" | "plus" | "premium",
};

export default function AdminTrainingsPage() {
  const [trainings, setTrainings] = useState<Training[]>([]);
  const [trainers, setTrainers] = useState<Trainer[]>([]);
  const [form, setForm] = useState(initialForm);
  const [editingId, setEditingId] = useState<string | null>(null);
  const [file, setFile] = useState<File | null>(null);
  const [preview, setPreview] = useState<string | null>(null);
  const [loading, setLoading] = useState(false);

  const loadData = async () => {
    try {
      const [trainingsRes, trainersRes] = await Promise.all([
        fetch("/api/trainings"),
        fetch("/api/trainers"),
      ]);
      const trainingsData = await trainingsRes.json();
      const trainersData = await trainersRes.json();
      setTrainings(
        trainingsData.map((t: any) => ({
          id: String(t._id ?? t.id),
          title: t.title,
          category: t.category,
          level: t.level,
          durationMin: t.durationMin,
          description: t.description,
          image: t.image,
          startAt: t.startAt,
          trainerId: t.trainer?._id || t.trainerId,
          minSubscription: t.minSubscription || "free",
```

```
          coach:
            t.coach ||
            (t.trainer ? `${t.trainer.firstName ?? ""} ${t.trainer.lastName ?? ""}`.trim() :
""),
        }))
      );
      setTrainers(
        trainersData.map((tr: any) => ({
          id: String(tr._id ?? tr.id),
          firstName: tr.firstName,
          lastName: tr.lastName,
        }))
      );
    } catch (e) {
      console.error(e);
    }
  };

  useEffect(() => {
    loadData();
  }, []);

  const handleChange = (
    e: React.ChangeEvent<HTMLInputElement | HTMLTextAreaElement | HTMLSelectElement>
  ) => {
    const { name, value } = e.target;
    setForm((prev) => ({
      ...prev,
      [name]: name === "durationMin" ? Number(value) || 0 : value,
    }));
  };

  function handleFileChange(e: React.ChangeEvent<HTMLInputElement>) {
    const f = e.target.files?.[0] || null;
    setFile(f);
    if (f) setPreview(URL.createObjectURL(f));
  }

  const handleEdit = (t: Training) => {
    setEditingId(t.id);
    setForm({
      title: t.title,
      category: t.category,
      level: t.level,
      durationMin: t.durationMin,
      description: t.description || "",
      trainerId: t.trainerId || "",
      startAt: t.startAt ? t.startAt.slice(0, 16) : "",
      minSubscription: (t.minSubscription as Training["minSubscription"]) || "",
    });
    setPreview(t.image || null);
    setFile(null);
  };

  const resetForm = () => {
    setForm(initialForm);
    setEditingId(null);
    setPreview(null);
    setFile(null);
  };

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setLoading(true);

    try {
      let imageUrl: string | undefined = preview || undefined;
```

```
      if (file) {
        const fd = new FormData();
        fd.append("file", file);
        const upload = await fetch("/api/upload", {
          method: "POST",
          body: fd,
        });
        if (!upload.ok) {
          const err = await upload.json().catch(() => ({}));
          throw new Error(err.error || "Cannot upload image");
        }
        const data = await upload.json();
        imageUrl = data.url;
      }

      const payload = { ...form, image: imageUrl };

      if (editingId) {
        const res = await fetch(`/api/trainings/${editingId}`, {
          method: "PATCH",
          headers: { "Content-Type": "application/json" },
          body: JSON.stringify(payload),
        });
        if (!res.ok) {
          throw new Error(await res.text());
        }
      } else {
        const res = await fetch("/api/trainings", {
          method: "POST",
          headers: { "Content-Type": "application/json" },
          body: JSON.stringify(payload),
        });
        if (!res.ok) {
          throw new Error(await res.text());
        }
      }

      await loadData();
      resetForm();
    } catch (error) {
      console.error(error);
      alert("Не вдалося зберегти тренування");
    } finally {
      setLoading(false);
    }
  };

  const handleDelete = async (id: string) => {
    if (!confirm("Видалити тренування?")) return;
    try {
      const res = await fetch(`/api/trainings/${id}`, { method: "DELETE" });
      if (!res.ok) throw new Error();
      setTrainings((prev) => prev.filter((t) => t.id !== id));
    } catch (e) {
      console.error(e);
      alert("Не вдалося видалити");
    }
  };

  return (
    <div className="mx-auto max-w-6xl px-4 py-8 space-y-6">
      <h1 className="mb-2 text-3xl font-extrabold">Адмін · Тренування</h1>

      <form
        onSubmit={handleSubmit}
        className="space-y-4 rounded-3xl bg-white p-6 shadow-[0_18px_40px_rgba(0,0,0,0.12)]
border border-black/5"
      >
```

```
<div className="flex items-center justify-between">
  <h2 className="text-xl font-semibold">
    {editingId ? "Редагувати тренування" : "Нове тренування"}
  </h2>
  {editingId && (
    <button
      type="button"
      onClick={resetForm}
      className="text-xs text-slate-600 underline"
    >
      Скасувати редагування
    </button>
  )}
</div>

<div className="grid gap-4 md:grid-cols-2">
  <div>
    <label className="mb-1 block text-sm font-medium">Назва</label>
    <input
      name="title"
      value={form.title}
      onChange={handleChange}
      className="w-full rounded-lg border px-3 py-2 text-sm"
      required
    />
  </div>
  <div>
    <label className="mb-1 block text-sm font-medium">Категорія</label>
    <select
      name="category"
      value={form.category}
      onChange={handleChange}
      className="w-full rounded-lg border px-3 py-2 text-sm"
    >
      <option value="cardio">Кардіо</option>
      <option value="functional">Функціональне</option>
      <option value="strength">Силове</option>
    </select>
  </div>
</div>

<div className="grid gap-4 md:grid-cols-3">
  <div>
    <label className="mb-1 block text-sm font-medium">Рівень</label>
    <select
      name="level"
      value={form.level}
      onChange={handleChange}
      className="w-full rounded-lg border px-3 py-2 text-sm"
    >
      <option value="beginner">Початковий</option>
      <option value="intermediate">Середній</option>
      <option value="advanced">Просунутий</option>
    </select>
  </div>
  <div>
    <label className="mb-1 block text-sm font-medium">Тривалість (хв)</label>
    <input
      type="number"
      name="durationMin"
      value={form.durationMin}
      onChange={handleChange}
      className="w-full rounded-lg border px-3 py-2 text-sm"
    />
  </div>
</div>

<div className="grid gap-4 md:grid-cols-2">
```

```
            <div>
              <label className="mb-1 block text-sm font-medium">Дата і час</label>
              <input
                type="datetime-local"
                name="startAt"
                value={form.startAt}
                onChange={handleChange}
                className="w-full rounded-lg border px-3 py-2 text-sm"
                required
              />
            </div>
            <div>
              <label className="mb-1 block text-sm font-medium">Мін. абонемент</label>
              <select
                name="minSubscription"
                value={form.minSubscription}
                onChange={handleChange}
                className="w-full rounded-lg border px-3 py-2 text-sm"
                required
              >
                <option value="">— Оберіть абонемент —</option>
                <option value="free">Free</option>
                <option value="plus">Plus</option>
                <option value="premium">Premium</option>
              </select>
            </div>
          </div>
          <div>
            <label className="mb-1 block text-sm font-medium">Опис</label>
            <textarea
              name="description"
              value={form.description}
              onChange={handleChange}
              className="w-full rounded-lg border px-3 py-2 text-sm"
              rows={3}
            />
          </div>
          <div>
            <label className="mb-1 block text-sm font-medium">Тренер</label>
            <select
              name="trainerId"
              value={form.trainerId || ""}
              onChange={handleChange}
              className="w-full rounded-lg border px-3 py-2 text-sm"
            >
              <option value="">— Оберіть тренера —</option>
              {trainers.map((tr) => (
                <option key={tr.id} value={tr.id}>
                  {tr.firstName} {tr.lastName}
                </option>
              ))}
            </select>
          </div>

          <div>
            <label className="mb-1 block text-sm font-medium">Обкладинка</label>
            <input
              type="file"
              accept="image/*"
              onChange={handleFileChange}
              className="w-full text-sm"
            />
            {preview && (
              <div className="mt-3 h-32 w-full overflow-hidden rounded-md border border-black/10
bg-black/5">
                {/* eslint-disable-next-line @next/next/no-img-element */}
                <img src={preview} alt="preview" className="h-full w-full object-cover" />
              </div>
```

```
          )}
        </div>

        <button
          type="submit"
          disabled={loading}
          className="rounded-full bg-primary px-6 py-2 text-sm font-semibold text-black
shadow-md hover:bg-primary/80 disabled:opacity-60"
        >
          {loading ? "Зберігаємо..." : editingId ? "Оновити тренування" : "Додати тренування"}
        </button>
      </form>

      <div className="space-y-3">
        <h2 className="text-xl font-semibold">Усі тренування</h2>
        {trainings.length === 0 ? (
          <p className="text-sm text-slate-600">Поки немає тренувань.</p>
        ) : (
          <div className="grid gap-4 md:grid-cols-2 lg:grid-cols-3">
            {trainings.map((t) => (
              <div
                key={t.id}
                className="rounded-2xl bg-white shadow px-4 py-3 space-y-2 border border-
black/5"
              >
                <div className="h-32 w-full overflow-hidden rounded-xl bg-black/5">
                  {/* eslint-disable-next-line @next/next/no-img-element */}
                  <img
                    src={t.image || "/img/hero-gym.jpg"}
                    alt={t.title}
                    className="h-full w-full object-cover"
                  />
                </div>
                <div className="text-sm font-semibold">{t.title}</div>
                <div className="text-xs text-slate-600">
                  {t.category} · {t.level} · {t.durationMin} хв
                </div>
                {t.startAt && (
                  <div className="text-xs text-slate-600">
                    {new Date(t.startAt).toLocaleString("uk-UA")}
                  </div>
                )}
                {t.minSubscription && (
                  <div className="text-xs text-slate-600">
                    Мін. абонемент: {t.minSubscription}
                  </div>
                )}
                {t.coach && <div className="text-xs text-slate-600">Тренер: {t.coach}</div>}
                {t.description && (
                  <p className="text-xs text-slate-700 line-clamp-3">{t.description}</p>
                )}
                <div className="flex gap-2 pt-1 text-xs">
                  <button
                    onClick={() => handleEdit(t)}
                    className="rounded-full border px-3 py-1 hover:bg-black/5"
                  >
                    Редагувати
                  </button>
                  <button
                    onClick={() => handleDelete(t.id)}
                    className="rounded-full border px-3 py-1 text-red-600 hover:bg-red-50"
                  >
                    Видалити
                  </button>
                </div>
              </div>
            ))}
          </div>
```

```
        )}
      </div>
    </div>
  );
}
```

Додаток Е

Powergym/app/profile/page.tsx

```
"use client";

import { useEffect, useState } from "react";
import { useRouter } from "next/navigation";
import Navbar from "@/components/Navbar";
import Footer from "@/components/Footer";
import Image from "next/image";

type AppUser = {
    id?: string;
    email?: string;
    name?: string;
    avatar?: string;
    points?: number;
    rank?: number;
    totalUsers?: number;
    subscription?: "free" | "plus" | "premium";
    subscriptionUntil?: string | null;
  };

  type BookingItem = {
    id: string;
    status: "active" | "cancelled" | "completed";
    createdAt?: string;
    training?: {
      id: string;
      title: string;
      category?: string;
      coachName?: string;
    } | null;
  };
type AiPlan = {
  id: string;
  text: string;
  createdAt?: string;
};
type PersonalTrainingItem = {
  id: string;
  title?: string;
  plan?: string;
  date?: string;
  status?: string;
  trainer?: { id?: string; name?: string } | null;
};

// Дні тижня
const weekDays = ["Пн.", "Вт.", "Ср.", "Чт.", "Пт.", "Сб.", "Нд."];

// ======================
//  КОМПОНЕНТ
// ======================
export default function ProfilePage() {

  const router = useRouter();
  const [user, setUser] = useState<AppUser | null>(null);
  const [weeklyPoints, setWeeklyPoints] = useState<number[]>(Array(7).fill(0));
  const [bookings, setBookings] = useState<BookingItem[]>([]);
  const [aiPlan, setAiPlan] = useState<AiPlan | null>(null);
  const [personalTrainings, setPersonalTrainings] = useState<PersonalTrainingItem[]>([]);
```

```
  const [selectedTraining, setSelectedTraining] = useState<PersonalTrainingItem | null>(null);

  // ======================
  //  useEffect
  // ======================
  useEffect(() => {


    if (typeof window === "undefined") return;

    const stored = localStorage.getItem("powergymUser");
    if (!stored) {
      router.push("/login");
      return;
    }

    const localUser = JSON.parse(stored) as AppUser;
    setUser(localUser);

    if (!localUser.id) return;

    // 1 — отримуємо дані користувача
    fetch(`/api/user/me?id=${localUser.id}`)
      .then((res) => res.json())
      .then((data) => {
        setUser((prev) => ({
          ...(prev ?? {}),
          ...data,
        }));
      });

    // 2 — рейтинг
    fetch(`/api/rating?id=${localUser.id}`)
      .then((res) => res.json())
      .then((ratingData) => {
        setUser((prev) => ({
          ...(prev ?? {}),
          rank: ratingData.rank,
          totalUsers: ratingData.totalUsers,
        }));
      });

    // 3 — статистика тижня
    fetch(`/api/user/stats?id=${localUser.id}`)
      .then((res) => res.json())
      .then((stats) => {
        if (Array.isArray(stats.weeklyPoints)) {
          setWeeklyPoints(stats.weeklyPoints);
        }
      });
      // 4 — мої записи на тренування
// 4 - мої записи на тренування
fetch(`/api/bookings?userId=${localUser.id}`)
  .then(async (res) => {
    if (!res.ok) {
      console.error("Bookings response not ok:", res.status);
      return [];
    }

    const text = await res.text();

    // якщо тіло порожнє - повертаємо пустий масив, щоб не ламати json()
    if (!text) return [];

    try {
      return JSON.parse(text);
    } catch (err) {
      console.error("Cannot parse bookings JSON:", err, text);
```

```
        return [];
      }
    })
    .then((list: any[]) => {
      const mapped: BookingItem[] = list.map((b) => ({
        id: b.id,
        status: b.status,
        createdAt: b.createdAt,
        training: b.training
          ? {
              id: b.training.id,
              title: b.training.title,
              category: b.training.category,
              coachName: b.training.coachName,
            }
          : null,
      }));
      setBookings(mapped);
    })
    .catch((e) => console.error("Error fetching bookings:", e));

  // 5 — AI-план тренувань
  fetch(`/api/ai-plan?userId=${localUser.id}`)
    .then((res) => res.json())
    .then((data) => {
      if (data && data.text) {
        setAiPlan({
          id: data.id,
          text: data.text,
          createdAt: data.createdAt,
        });
      }
    })
    .catch((e) => console.error("Error fetching AI plan:", e));

  // 6 — персональні тренування
  fetch(`/api/personal-trainings?userId=${localUser.id}`)
    .then((res) => res.json())
    .then((list: any[]) => {
      if (!Array.isArray(list)) return;
      const mapped: PersonalTrainingItem[] = list.map((pt) => ({
        id: pt.id,
        title: pt.title,
        plan: pt.plan,
        date: pt.date,
        status: pt.status,
        trainer: pt.trainer
          ? { id: pt.trainer.id, name: pt.trainer.name }
          : null,
      }));
      setPersonalTrainings(mapped);
    })
    .catch((e) => console.error("Error fetching personal trainings:", e));

  }, [router]);

  if (!user) return null;

  // ДЕСТРУКТУРИЗАЦІЯ
  const name = user.name || "Користувачу";
  const points = user.points ?? 0;
  const rank = user.rank ?? 0;
  const totalUsers = user.totalUsers ?? 0;
  const subscription = user.subscription ?? "free";
  const subscriptionUntil = user.subscriptionUntil
    ? new Date(user.subscriptionUntil)
    : null;
  const maxWeekly = Math.max(...weeklyPoints);
```

```
  const hasAnyWeeklyPoints = maxWeekly > 0;

  // Мапа підписок
  const subscriptionInfo =
  subscription === "premium"
    ? {
        label: "Повний доступ",
        description: "Ваш преміум активний",
        price: "800 грн",
        isActive: !!subscriptionUntil,
      }
    : subscription === "plus"
    ? {
        label: "Абонемент Плюс",
        description: "Доступ до тренажерів та групових тренувань",
        price: "600 грн",
        isActive: !!subscriptionUntil,
      }
    : {
        label: "Без абонементу",
        description: "Оберіть абонемент, щоб розблокувати всі можливості залу",
        price: "",
        isActive: false,
      };

  const handleBuySubscription = () => router.push("/subscriptions");

  return (
    <>
      <main className="min-h-screen bg-[#F4F7F6] text-black flex flex-col">
        <Navbar />

        <div className="flex-1">
          <section className="mx-auto max-w-5xl px-4 py-8">

            {/* Заголовок */}
            <h1 className="text-center text-3xl md:text-4xl font-extrabold mb-2">
              Особистий кабінет
            </h1>
            <p className="text-center text-sm md:text-base text-slate-700">
              Привіт, {name}! Гарного тренування сьогодні ☺
            </p>
{/* ПАНЕЛЬ КОРИСТУВАЧА */}
<div className="mt-6 rounded-3xl bg-white shadow-[0_18px_40px_rgba(0,0,0,0.15)] px-6 py-4 flex
flex-col md:flex-row items-center gap-4">
  <div className="flex items-center gap-3 w-full md:w-auto">
    <div className="relative h-14 w-14 rounded-full overflow-hidden bg-slate-200">
      <Image
        src={user.avatar || "/img/default-avatar.png"}
        alt={name}
        fill
        className="object-cover"
      />
    </div>
    <div>
      <p className="text-sm font-semibold">{name}</p>
      <p className="text-xs text-slate-600">{user.email}</p>
    </div>
  </div>

  <div className="flex-1 flex justify-end w-full">
    <button
      onClick={() => router.push("/trainings")}
      className="rounded-full bg-[#8DD9BE] px-5 py-1.5 text-xs font-semibold text-black shadow
hover:bg-[#7ACDAE]"
    >
      Відкрити список тренувань
    </button>
```

```
  </div>
</div>

            {/* ============================
               КАРТКА 1 — БАЛИ + ГРАФІК
            ============================ */}
            <div className="mt-8 rounded-3xl bg-white shadow-xl px-6 py-5 flex flex-col
md:flex-row gap-6">
              <div className="flex flex-col items-center w-full md:w-1/3">
                <div className="text-3xl mb-1">🏋️‍♂️</div>
                <p className="text-4xl font-extrabold">{points}</p>
                <p className="text-sm text-slate-700 mt-1">
                  Цього тижня {hasAnyWeeklyPoints ? "" : "ще"} 0 балів
                </p>
              </div>

              <div className="w-full md:w-2/3">
                {!hasAnyWeeklyPoints ? (
                  <div className="flex h-40 items-center justify-center text-xs text-slate-
500">
                    За цей тиждень ще не нараховано балів
                  </div>
                ) : (
                  <div className="flex items-end justify-between h-40 px-3">
                    {weeklyPoints.map((v, idx) => (
                      <div key={idx} className="flex flex-col items-center">
                        <div
                          className="w-6 rounded-t-lg bg-[#8DD9BE]"
                          style={{ height: `${(v / maxWeekly) * 100}%` }}
                        />
                        <span className="text-[10px] text-slate-700">
                          {weekDays[idx]}
                        </span>
                      </div>
                    ))}
                  </div>
                )}
              </div>
            </div>

            {/* ============================
               КАРТКА 2 — АБОНЕМЕНТ
            ============================ */}
            <div className="mt-6 rounded-3xl bg-white shadow-xl px-6 py-5 flex flex-col
md:flex-row gap-6">
              <div className="flex flex-col w-full md:w-1/2">
                <div className="text-3xl mb-1">$</div>
                <p className="text-2xl md:text-3xl font-extrabold">
  {subscriptionInfo.label}
</p>
<p className="mt-1 text-sm text-slate-700">
  {subscriptionInfo.isActive && subscriptionUntil
    ? `Дійсний до: ${subscriptionUntil.toLocaleDateString("uk-UA", {
        day: "numeric",
        month: "long",
        year: "numeric",
      })}`
    : subscriptionInfo.description}
</p>

                <button
                  onClick={handleBuySubscription}
                  className="mt-4 w-fit rounded-full bg-[#8DD9BE] px-5 py-1.5 text-xs font-
semibold text-black"
                >
                  Купити / змінити абонемент
                </button>
              </div>
```

```
<div className="w-full md:w-1/2 flex justify-center">
                <div className="relative h-40 w-full max-w-xs rounded-2xl overflow-hidden
shadow-lg">
                  {/* Картинка */}
                  <Image
                    src="/img/hero-gym.jpg"
                    alt="sub"
                    fill
                    className="object-cover"
                  />

                  {/* Темний градієнт зверху для читабельності тексту */}
                  <div className="absolute inset-0 bg-gradient-to-t from-black/70 to-
transparent" />

                  {/* Текст зверху зліва */}
                  <div className="absolute left-3 top-3 text-xs font-semibold text-white">
                    {subscriptionInfo.isActive ? subscriptionInfo.label : "Повний доступ"}
                    {subscriptionInfo.isActive && subscriptionUntil && (
                      <div className="text-[10px] text-slate-200">
                        Дійсний до{" "}
                        {subscriptionUntil.toLocaleDateString("uk-UA", {
                          day: "numeric",
                          month: "long",
                          year: "numeric",
                        })}
                      </div>
                    )}
                  </div>

                  {/* Ціна знизу справа */}
                  {subscriptionInfo.price && (
                    <div className="absolute right-3 bottom-3 text-sm font-bold text-white">
                      {subscriptionInfo.price}
                    </div>
                  )}
                </div>
              </div>

            </div>

            {/* ============================
               КАРТКА 3 — РЕЙТИНГ
            ============================ */}
            <div className="mt-6 rounded-3xl bg-white shadow-xl px-6 py-5 flex flex-col
md:flex-row gap-6">
              <div className="flex flex-col items-center w-full md:w-1/2">
                <div className="text-3xl mb-1">🎖</div>
                <p className="text-3xl font-extrabold">{rank}</p>
                <p className="text-sm text-slate-700 mt-1">
                  Твоє місце: {rank} з {totalUsers} учасників
                </p>
              </div>

              <div className="w-full md:w-1/2 flex items-center justify-center">
                <div className="flex h-28 w-full max-w-xs items-center justify-center rounded-
2xl bg-slate-50 border border-dashed text-sm text-slate-500">
                  Рейтинг серед усіх користувачів
                </div>
              </div>
            </div>
{/* МОЇ ЗАПИСИ НА ТРЕНУВАННЯ */}
<div className="mt-8 rounded-3xl bg-white shadow-[0_18px_40px_rgba(0,0,0,0.15)] px-6 py-5">
  <h2 className="text-base md:text-lg font-extrabold mb-3">
    Мої записи на тренування
  </h2>
```

```
  {bookings.length === 0 ? (
    <p className="text-xs text-slate-600">
      Ви ще не записувались на тренування. Відкрийте список тренувань і
      оберіть те, що вам підходить.
    </p>
  ) : (
    <ul className="space-y-2">
      {bookings.map((b) => {
        const date =
          b.createdAt &&
          new Date(b.createdAt).toLocaleString("uk-UA", {
            day: "2-digit",
            month: "2-digit",
            year: "numeric",
            hour: "2-digit",
            minute: "2-digit",
          });
        const statusLabel =
          b.status === "completed"
            ? "Завершено"
            : b.status === "cancelled"
            ? "Скасовано"
            : "Активний";

        return (
          <li
            key={b.id}
            className="flex flex-col md:flex-row md:items-center md:justify-between rounded-
2xl bg-slate-50 px-3 py-2 text-xs"
          >
            <div>
              <p className="font-semibold">
                {b.training?.title || "Тренування"}
              </p>
              <p className="text-slate-600">
                {b.training?.category}{" "}
                {b.training?.coachName &&
                  `• тренер: ${b.training.coachName}`}
              </p>
            </div>
            <div className="mt-1 md:mt-0 text-right">
              {date && (
                <p className="text-[11px] text-slate-500 mb-0.5">{date}</p>
              )}
              <p className="text-[11px] font-semibold text-slate-700">
                Статус: {statusLabel}
              </p>
            </div>
          </li>
        );
      })}
    </ul>
  )}
</div>
{/* ПЕРСОНАЛЬНІ ТРЕНУВАННЯ */}
<div className="mt-6 rounded-3xl bg-white shadow-[0_18px_40px_rgba(0,0,0,0.15)] px-6 py-5">
  <h2 className="text-base md:text-lg font-extrabold mb-3">
    Персональні тренування
  </h2>

  {personalTrainings.length === 0 ? (
    <p className="text-xs text-slate-600">
      Персональних тренувань поки немає. Запишіться у тренера, щоб побачити їх тут.
    </p>
  ) : (
    <ul className="space-y-2">
      {personalTrainings.map((pt) => {
        const dateStr =
```

```
            pt.date &&
            new Date(pt.date).toLocaleString("uk-UA", {
              day: "2-digit",
              month: "2-digit",
              year: "numeric",
              hour: "2-digit",
              minute: "2-digit",
            });

          const statusLabel =
            pt.status === "done"
              ? "Завершено"
              : pt.status === "cancelled"
              ? "Скасовано"
              : "Заплановано";
          const planText = pt.plan ? ` • План: ${pt.plan}` : "";
          return (
            <li
              key={pt.id}
              className="flex flex-col md:flex-row md:items-center md:justify-between rounded-
2xl bg-slate-50 px-3 py-2 text-xs"
            >
              <div className="space-y-0.5">
                <p className="font-semibold">
                  {pt.title || "Персональне тренування"}
                </p>
                <p className="text-slate-600">
                  {pt.trainer?.name && `Тренер: ${pt.trainer.name}`}
                  {planText}
                </p>
              </div>
              <div className="mt-1 md:mt-0 text-right space-y-1">
                {dateStr && (
                  <p className="text-[11px] text-slate-500 mb-0.5">{dateStr}</p>
                )}
                <p className="text-[11px] font-semibold text-slate-700">
                  Статус: {statusLabel}
                </p>
                <button
                  onClick={() => setSelectedTraining(pt)}
                  className="inline-block rounded-full border border-slate-300 px-3 py-1 text-
[11px] font-semibold text-slate-700 hover:bg-slate-100"
                >
                  Переглянути
                </button>
              </div>
            </li>
          );
        })}
      </ul>
    )}
  </div>
  {/* AI-план тренувань */}
  <div className="mt-6 rounded-3xl bg-white shadow-[0_18px_40px_rgba(0,0,0,0.15)] px-6 py-5">
    <h2 className="text-base md:text-lg font-extrabold mb-3">
      AI-план тренувань
    </h2>
    {!aiPlan ? (
      <p className="text-xs text-slate-600">
        Ви ще не зберігали AI-план. Створіть його на сторінці AI-тренера.
      </p>
    ) : (
      <>
        {aiPlan.createdAt && (
          <p className="text-[11px] text-slate-500 mb-2">
            Останнє оновлення:{" "}
            {new Date(aiPlan.createdAt).toLocaleString("uk-UA", {
              day: "2-digit",
```

```
            month: "2-digit",
            year: "numeric",
            hour: "2-digit",
            minute: "2-digit",
          })}
        </p>
      )}
      <div className="rounded-2xl bg-slate-50 px-3 py-3 text-xs whitespace-pre-wrap text-
slate-800">
        {aiPlan.text}
      </div>
    </>
  )}
</div>
          </section>
        </div>
        <Footer />
      </main>
      {selectedTraining && (
        <div className="fixed inset-0 z-50 flex items-center justify-center bg-black/50 px-4">
          <div className="w-full max-w-md rounded-2xl bg-white p-5 shadow-2xl">
            <div className="flex items-start justify-between gap-3">
              <div>
                <p className="text-xs uppercase tracking-wide text-slate-500">Персональне
тренування</p>
                <h3 className="text-lg font-extrabold mt-1">
                  {selectedTraining.title || "Персональне тренування"}
                </h3>
              </div>
              <button
                onClick={() => setSelectedTraining(null)}
                className="rounded-full bg-slate-100 px-2 py-1 text-xs font-semibold text-
slate-700 hover:bg-slate-200"
              >
                Закрити
              </button>
            </div>
            <div className="mt-4 space-y-2 text-sm text-slate-700">
              {selectedTraining.date && (
                <p>
                  <span className="font-semibold">Дата:</span>{" "}
                  {new Date(selectedTraining.date).toLocaleString("uk-UA", {
                    day: "2-digit",
                    month: "2-digit",
                    year: "numeric",
                    hour: "2-digit",
                    minute: "2-digit",
                  })}
                </p>
              )}
              {selectedTraining.trainer?.name && (
                <p>
                  <span className="font-semibold">Тренер:</span>
{selectedTraining.trainer.name}
                </p>
              )}
              {selectedTraining.plan && (
                <p>
                  <span className="font-semibold">План:</span> {selectedTraining.plan}
                </p>
              )}
              <p>
                <span className="font-semibold">Статус:</span>{" "}
                {selectedTraining.status === "done"
                  ? "Завершено"
                  : selectedTraining.status === "cancelled"
                  ? "Скасовано"
                  : "Заплановано"}
```

```
            </p>
          </div>
        </div>
      </div>
    )}
  </>
 );
}
```

Додаток Ж

Powergym/app/ schedule/page.tsx

```
"use client";

import { useEffect, useState } from "react";
import { useRouter } from "next/navigation";
import Navbar from "@/components/Navbar";
import Footer from "@/components/Footer";
import Image from "next/image";

type LeaderItem = {
  id: string;
  name: string;
  avatar?: string | null;
  points: number;
  rank: number;
};

type BookingItem = {
  id: string;
  createdAt?: string;
  training?: {
    startAt?: string;
  };
};

type LocalUser = {
  id?: string;
  name?: string;
};

function getMonthDays(year: number, month: number) {
  const daysInMonth = new Date(year, month + 1, 0).getDate();
  const days: Date[] = [];
  for (let d = 1; d <= daysInMonth; d++) days.push(new Date(year, month, d));
  return days;
}

export default function SchedulePage() {
  const router = useRouter();

  const [hallCount, setHallCount] = useState<number | null>(null);
  const [leaders, setLeaders] = useState<LeaderItem[]>([]);
  const [user, setUser] = useState<LocalUser | null>(null);
  const [bookingDates, setBookingDates] = useState<Set<string>>(() => new Set());
  const [trainingDates, setTrainingDates] = useState<Set<string>>(() => new Set());
  const [loadingCalendar, setLoadingCalendar] = useState(false);

  const today = new Date();
  const currentYear = today.getFullYear();
  const currentMonth = today.getMonth();
  const monthDays = getMonthDays(currentYear, currentMonth);

  const dateKey = (d: Date) =>
    `${d.getFullYear()}-${String(d.getMonth() + 1).padStart(2, "0")}-${String(
      d.getDate()
```

```
    ).padStart(2, "0")}`;

  useEffect(() => {
    const stored = typeof window !== "undefined" ? localStorage.getItem("powergymUser") :
null;
    if (stored) {
      try {
        setUser(JSON.parse(stored) as LocalUser);
      } catch (e) {
        console.error("Cannot parse powergymUser:", e);
      }
    }
  }, []);

  useEffect(() => {
    fetch("/api/gym-load")
      .then(async (res) => (res.ok ? res.json() : null))
      .then((data) => {
        if (data && typeof data.count === "number") setHallCount(data.count);
      })
      .catch((e) => console.error("Error fetching gym-load:", e));

    fetch("/api/rating?top=5")
      .then(async (res) => (res.ok ? res.json() : []))
      .then((list: any[]) => {
        setLeaders(
          list.map((u) => ({
            id: u.id,
            name: u.name,
            avatar: u.avatar,
            points: u.points,
            rank: u.rank,
          }))
        );
      })
      .catch((e) => console.error("Error fetching rating leaderboard:", e));

    fetch("/api/trainings")
      .then(async (res) => (res.ok ? res.json() : []))
      .then((list: any[]) => {
        const dates = new Set<string>();
        list.forEach((t) => {
          if (t.startAt) dates.add(dateKey(new Date(t.startAt)));
        });
        setTrainingDates(dates);
      })
      .catch((e) => console.error("Error fetching trainings:", e));
  }, []);

  useEffect(() => {
    if (!user?.id) return;

    setLoadingCalendar(true);
    fetch(`/api/bookings?userId=${user.id}`)
      .then(async (res) => {
        if (!res.ok) return [];
        const text = await res.text();
        if (!text) return [];
        try {
          return JSON.parse(text);
        } catch (error) {
          console.error("Cannot parse bookings JSON:", error, text);
          return [];
        }
      })
      .then((list: any[]) => {
        const dates = new Set<string>();
        (list as BookingItem[]).forEach((b) => {
```

```
          const dStr = b.training?.startAt || b.createdAt;
          if (!dStr) return;
          const d = new Date(dStr);
          dates.add(dateKey(d));
        });
        setBookingDates(dates);
      })
      .catch((e) => console.error("Error fetching bookings:", e))
      .finally(() => setLoadingCalendar(false));
  }, [user?.id]);

  const monthName = today.toLocaleString("uk-UA", {
    month: "long",
    year: "numeric",
  });

  const maxHall = 60;
  const hallPercent =
    hallCount != null ? Math.min(100, Math.round((hallCount / maxHall) * 100)) : 0;

  return (
    <main className="min-h-screen bg-[#F4F7F6] text-black flex flex-col">
      <Navbar />

      <div className="flex-1">
        <section className="mx-auto max-w-5xl px-4 py-10">
          <h1 className="text-center text-3xl md:text-4xl font-extrabold mb-2">
            Розклад тренувань
          </h1>
          <p className="text-center text-sm md:text-base text-slate-700 mb-8">
            Плануйте відвідування, бронюйте тренування та слідкуйте за активністю в клубі.
          </p>

          <div className="mx-auto max-w-xl rounded-3xl bg-white shadow-
[0_18px_40px_rgba(0,0,0,0.15)] px-6 py-5 mb-8">
            <h2 className="text-sm md:text-base font-extrabold mb-3">
              Поточне завантаження залу (симуляція)
            </h2>
            <p className="text-xs text-slate-600 mb-3">
              Дані умовні, показують загальну кількість людей у залі зараз.
            </p>

            <div className="flex items-center gap-4">
              <div className="flex-1 h-3 rounded-full bg-slate-100 overflow-hidden">
                <div
                  className="h-full bg-[#8DD9BE] transition-all"
                  style={{ width: `${hallPercent}%` }}
                />
              </div>
              <div className="text-sm font-semibold min-w-[80px] text-right">
                {hallCount != null ? `${hallCount} осіб` : "—"}
              </div>
            </div>

            <p className="mt-1 text-[11px] text-slate-500">Максимальна місткість: {maxHall}
осіб.</p>
          </div>

          <div className="grid gap-6 md:grid-cols-[minmax(0,1.4fr)_minmax(0,1.8fr)]">
            <div className="rounded-3xl bg-white shadow-[0_18px_40px_rgba(0,0,0,0.15)] px-6
py-5">
              <h2 className="text-sm md:text-base font-extrabold mb-3">
                Лідери (за заробленими балами)
              </h2>
              {leaders.length === 0 ? (
                <p className="text-xs text-slate-600">Ще немає даних. Спробуйте пізніше.</p>
              ) : (
                <ul className="space-y-2 text-xs">
```

```
                {leaders.map((u) => (
                  <li
                    key={u.id}
                    className="flex items-center justify-between rounded-2xl bg-[#E6FFF3]
px-3 py-2"
                  >
                    <div className="flex items-center gap-3">
                      <span className="font-semibold">{u.rank}.</span>
                      <div className="relative h-8 w-8 rounded-full overflow-hidden bg-
slate-200">
                        {u.avatar ? (
                          <Image src={u.avatar} alt={u.name} fill className="object-cover"
/>
                        ) : (
                          <div className="flex h-full w-full items-center justify-center
text-[10px] text-slate-600">
                            {u.name.charAt(0).toUpperCase()}
                          </div>
                        )}
                      </div>
                      <span className="font-semibold">{u.name}</span>
                    </div>
                    <span className="text-[11px] text-slate-700">{u.points} балів</span>
                  </li>
                ))}
              </ul>
            )}
          </div>

          <div className="rounded-3xl bg-white shadow-[0_18px_40px_rgba(0,0,0,0.15)] px-6
py-5">
            <h2 className="text-sm md:text-base font-extrabold mb-3">
              Календар ваших відвідувань
            </h2>

            {!user?.id ? (
              <div className="rounded-2xl bg-[#F4F7F6] px-4 py-6 text-center">
                <p className="text-xs text-slate-700 mb-4">
                  Увійдіть, щоб бачити свої відмічені дні та бронювання.
                </p>
                <button
                  onClick={() => router.push("/login")}
                  className="rounded-full bg-[#8DD9BE] px-6 py-2 text-xs font-semibold text-
black shadow hover:bg-[#7ACDAE]"
                >
                  Увійти
                </button>
              </div>
            ) : (
              <div>
                <p className="text-xs text-slate-600 mb-2">
                  {monthName.charAt(0).toUpperCase() + monthName.slice(1)}
                </p>

                {loadingCalendar && (
                  <p className="text-[11px] text-slate-500 mb-2">Оновлюємо дані...</p>
                )}

                <div className="grid grid-cols-7 gap-1 text-[11px]">
                  {["Пн", "Вт", "Ср", "Чт", "Пт", "Сб", "Нд"].map((d) => (
                    <div
                      key={d}
                      className="text-center font-semibold text-slate-600 mb-1"
                    >
                      {d}
                    </div>
                  ))}
```

```
                    {(() => {
                      const first = new Date(currentYear, currentMonth, 1).getDay();
                      const shift = first === 0 ? 6 : first - 1;
                      return Array.from({ length: shift }).map((_, i) => (
                        <div key={`empty-${i}`} />
                      ));
                    })()}

                    {monthDays.map((d) => {
                      const key = dateKey(d);
                      const hasTraining = bookingDates.has(key) || trainingDates.has(key);
                      const isToday =
                        d.getDate() === today.getDate() &&
                        d.getMonth() === today.getMonth() &&
                        d.getFullYear() === today.getFullYear();

                      return (
                        <div
                          key={key}
                          className={[
                            "flex h-8 w-8 items-center justify-center rounded-full mx-auto
text-[11px]",

                            hasTraining
                              ? "bg-[#8DD9BE] text-black font-semibold"
                              : "text-slate-700",
                            isToday && !hasTraining ? "border border-[#8DD9BE]" : "",
                          ]
                            .filter(Boolean)
                            .join(" ")}
                        >
                          {d.getDate()}
                        </div>
                      );
                    })}
                  </div>

                  <p className="mt-3 text-[10px] text-slate-500">
                    Зеленим позначені дні з бронюваннями або запланованими тренуваннями.
                  </p>
                </div>
              )}
            </div>
          </div>
        </section>
      </div>

      <Footer />
    </main>
  );
}
```

Додаток 3

Powergym/.env.local

```
MONGO_URI="mongodb+srv://admin:admin@cluster0.vivsuwl.mongodb.net/?retryWrites=true&w=majority
&appName=Cluster0"

NEXTAUTH_SECRET="your-secret-here"
JWT_SECRET="your-jwt-secret"

NEXTAUTH_URL=http://localhost:3000
```

Додаток И

Powergym/app/page.tsx

```
import Image from "next/image";
import Navbar from "@/components/Navbar";
import BenefitCard from "@/components/BenefitCard";
import NewsCard from "@/components/NewsCard";
import NewsSection, { NewsItem } from "@/components/NewsSection";
import Footer from "@/components/Footer";
import { Hero } from "@/components/Hero";

async function getNews(): Promise<NewsItem[]> {
  const res = await fetch("http://localhost:3000/api/news", { cache: "no-store" });
  if (!res.ok) return [];
  return res.json();
}

export default async function HomePage() {
  const news = await getNews();

  return (
    <main className="min-h-screen bg-[#f2f5f7] text-black flex flex-col">
      <Navbar />

      {/* HERO */}
      <Hero/>


      {/* НАШІ ПЕРЕВАГИ */}
      <section className="mx-auto max-w-6xl px-4 pb-6 pt-4">
        <h2 className="mb-6 text-center text-2xl font-bold tracking-wide">
          НАШІ ПЕРЕВАГИ
        </h2>
        <div className="grid gap-5 md:grid-cols-3">
          <BenefitCard
            icon={<span>🏋️‍♂️</span>}
            title="Сучасне обладнання"
            description="Професійні тренажери нового покоління, що відповідають міжнародним
стандартам."
          />
          <BenefitCard
            icon={<span>🤖</span>}
            title="AI-Тренер"
            description="Штучний інтелект створює персональну програму тренувань під твій ритм
життя."
          />
          <BenefitCard
            icon={<span>🏆</span>}
            title="Система досягнень"
            description="Збирай бали, підвищуй рейтинг і обмінюй їх на бонуси та знижки."
          />
        </div>
      </section>
      {/* НОВИНИ ТА АКЦІЇ */}
      <section className="mx-auto max-w-6xl px-4 pb-16 pt-6">
        {/* НОВИНИ ТА АКЦІЇ */}
<NewsSection items={news} />

      </section>

      <Footer />
    </main>
  );
}
```