

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Додаток для аналізу тренувальних показників та харчування»

Виконав студент групи ПЗм-23-1
Лимарь Дмитро Андрійович

Керівник роботи:
Власенко Олег Васильович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2024

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

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«1» жовтня 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **Лимарь Дмитро Андрійович**

Керівник роботи: **Власенко Олег Васильович**

Тема роботи: **«Додаток для аналізу тренувальних показників та харчування».**

затверджена Наказом закладу вищої освіти від «1» жовтня 2024 р., №520с

Вихідні дані для роботи (зазначається предмет і об'єкт дослідження):
інформаційні технології для розробки додатка та використання веб-технології для розробки.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Власенко О.В.	01.10.2024р.	01.10.2024р.
2	Власенко О.В.	01.10.2024р.	01.10.2024р.
3	Власенко О.В.	01.10.2024р.	01.10.2024р.

РЕФЕРАТ

Метою роботи є розробка інформаційного додатка для аналізу тренувальних показників і харчування, що сприяє профілактиці хронічних захворювань, таких як ожиріння та діабет, через інтеграцію сучасних наукових підходів до планування раціону та фізичної активності. Додаток дозволяє користувачам ефективно керувати своїм здоров'ям, підтримуючи баланс макро- і мікроелементів та оптимізуючи тренувальні програми.

У роботі визначено основні завдання: проведено аналіз сучасних наукових досліджень щодо впливу харчування та фізичної активності на розвиток і управління хронічними захворюваннями; обґрунтовано архітектуру додатка, що враховує індивідуальні потреби користувачів; розроблено структуру бази даних та реалізовано ключові функціональні модулі.

Особливістю розробки є інтеграція функціоналу для:

- автоматичного розрахунку КБЖУ (калорій, білків, жирів, вуглеводів) із урахуванням науково обґрунтованих норм;
- генерації збалансованих раціонів для покращення стану здоров'я;
- моніторингу тренувального прогресу з візуалізацією змін ваги та основних показників.

Для реалізації використано сучасні технології: мову програмування PHP, фреймворк Laravel, систему управління базами даних MySQL, а також сучасні веб-інструменти (HTML, CSS).

Наукова новизна роботи полягає у створенні інформаційного додатка, який поєднує рекомендації щодо харчування та тренувань для профілактики ожиріння й діабету. Інтеграція цих аспектів в одну платформу дозволяє не лише досягти спортивних цілей, але й підтримувати загальний стан здоров'я користувачів.

Випускна робота складається з програми інформаційного додатка та пояснювальної записки. Пояснювальна записка містить 89 сторінки, 27 ілюстрацій, 15 таблиць.

КЛЮЧОВІ СЛОВА: L11, MVC, БД, ФД, ВД, КБЖВ, BMR.

ABSTRACT

The aim of the work is to develop an information application for the analysis of training indicators and nutrition, which contributes to the prevention of chronic diseases, such as obesity and diabetes, through the integration of modern scientific approaches to diet planning and physical activity. The application allows users to effectively manage their health, maintaining a balance of macro- and microelements and optimizing training programs.

The main tasks are defined in the work: an analysis of modern scientific research on the impact of nutrition and physical activity on the development and management of chronic diseases was conducted; the architecture of the application was substantiated, taking into account the individual needs of users; the database structure was developed and key functional modules were implemented.

A feature of the development is the integration of functionality for:

- automatic calculation of KBJU (calories, proteins, fats, carbohydrates) taking into account scientifically based norms;
- generation of balanced diets to improve health;
- monitoring training progress with visualization of weight changes and key indicators.

For implementation, modern technologies were used: PHP programming language, Laravel framework, MySQL database management system, as well as modern web tools (HTML, CSS).

The scientific novelty of the work lies in the creation of an information application that combines recommendations on nutrition and training for the prevention of obesity and diabetes. The integration of these aspects into one platform allows not only to achieve sports goals, but also to maintain the general health of users.

The thesis consists of an information application program and an explanatory note. The explanatory note contains 89 pages, 27 illustrations, 15 tables.

KEYWORDS: L11, MVC, DATABASE, FD, VD, KBJ, BMR.

ЗМІСТ

РЕФЕРАТ	2
ABSTRACT	5
ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ІНФОРМАЦІЙНОГО ДОДАТКА	10
1.1 ПОСТАНОВКА ЗАДАЧІ	10
1.2 АНАЛІЗ АНАЛОГІВ ПРОГРАМНОГО ПРОДУКТУ	11
1.3 ОБҐРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ТА ВИМОГИ ДО АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ.....	18
1.4 АНАЛІЗ МЕТОДІВ РОЗРАХУНКУ КЛЮЧОВИХ ПАРАМЕТРІВ ДЛЯ РОБОТИ ДОДАТКА.....	20
Висновки до розділу 1.....	24
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ІНФОРМАЦІЙНОГО ДОДАТКУ ..	26
2.1 ВИЗНАЧЕННЯ ВАРІАНТІВ ВИКОРИСТАННЯ ТА СТРУКТУРИ СИСТЕМИ.....	26
2.2 РОЗРОБКА БАЗИ ДАНИХ СИСТЕМИ.....	32
2.3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ АЛГОРИТМІВ РОБОТИ СИСТЕМИ	42
2.4 РЕАЛІЗАЦІЯ ФУНКЦІОНАЛУ ІНФОРМАЦІЙНОГО ДОДАТКУ.....	43
Висновки до розділу 2.....	54
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКА	56
3.1 ПОРЯДОК ВСТАНОВЛЕННЯ ТА НАЛАШТУВАННЯ СИСТЕМИ	56
3.2 СТРУКТУРА ІНТЕРФЕЙСУ. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ДОДАТКОМ ..	57
3.3 ТЕСТУВАННЯ РОБОТИ ПРОГРАМНОГО КОМПЛЕКСУ	70
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	73
ДОДАТКИ.....	75
ДОДАТОК А.....	76

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

L11 – Laravel 11 версії.

БД – База даних.

MVC – Model–View–Controller (з англ. Модель–Представлення–Контролер).

ФД – Щоденник харчування.

ВД – Щоденник тренувань.

КБЖВ – калорії, білки, жири та вуглеводи.

BMR - basal metabolic rate (з англ. Базова швидкість метаболізму).

ВСТУП

Актуальність теми. Швидкий ритм життя сучасної людини та нестача часу сприяє розвиненню неправильних харчових звичок, відсутності фізичних навантажень в цілому негативно позначається на здоров'ї. При тривалому не правильному харчування та не дотриманні базових норм можуть розвиватися хронічні захворювання, такі як: ожиріння, діабет і загальні проблеми з шлунково-кишковим трактом. За даними Всесвітньої організації охорони здоров'я, понад 650 мільйонів людей у світі мають ожиріння, а близько 450 мільйонів страждають на діабет II типу. Основними причинами цих захворювань є незбалансоване харчування, відсутність фізичної активності та обмеженість інструментів для комплексного моніторингу здоров'я

Особливу роль у вирішенні цієї проблеми відіграють цифрові технології, які дозволяють інтегрувати індивідуалізовані підходи до харчування та тренувань у повсякденне життя. Наукові дослідження підтверджують, що контроль за споживанням калорій і нутрієнтів, а також регулярна фізична активність сприяють зниженню ризиків хронічних захворювань. Водночас існуючі додатки здебільшого не охоплюють комплексного підходу до моніторингу харчування і тренувань.

Мета дослідження. Створення інформаційного додатка для аналізу тренувальних показників і харчування, який інтегрує сучасні наукові підходи до моніторингу здоров'я та підтримки спортивного прогресу.

Завдання дослідження:

- Провести аналіз існуючих аналогів і визначити їхні недоліки;
- Розробити алгоритми для розрахунку КБЖВ та генерації раціонів;
- Реалізувати функціонал для відслідковування тренувального прогресу;
- Провести тестування додатка на основі реальних даних.

Об'єкт дослідження. Інформаційні технології для моніторингу харчування та тренувального прогресу.

Предмет дослідження. Методи розробки додатків для інтеграції функцій моніторингу харчування і тренувань.

Практичне значення. Створений додаток дозволяє користувачам:

- Планувати харчування з урахуванням індивідуальних потреб;
- Відстежувати тренувальний прогрес за допомогою інтерактивної візуалізації;
- Знижувати ризики хронічних захворювань через контроль за здоровим способом життя.

Апробація результатів роботи. Матеріали кваліфікаційної роботи доповідалися на VII Всеукраїнської науково-технічної конференції «Математичне моделювання та розробка програмного забезпечення» 02-03 грудня 2024 року.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ІНФОРМАЦІЙНОГО ДОДАТКА

1.1 Постановка задачі

Головною причиною створення додатка є забезпечення людей інструментом. Інструментом, який дозволить аналізувати харчування та тренування для покращення здоров'я. Профілактики хронічних захворювань, таких як діабет та ожиріння, а також підвищення спортивного прогресу.

Сучасний світ стикається з викликами, пов'язаними з погіршенням харчових звичок та недостатньою фізичною активністю. За даними Всесвітньої організації охорони здоров'я, понад 13% населення світу має надмірну вагу або ожиріння, що суттєво збільшує ризик розвитку діабету II типу та серцево-судинних захворювань. У цьому контексті стає необхідним створення цифрових рішень, що дозволяють користувачам автоматизувати процес планування раціону, відстеження калорій та контроль фізичної активності.

Запропонований додаток має забезпечити:

Автоматизацію розрахунку індивідуальних потреб у калоріях, білках, жирах та вуглеводах (КБЖВ). Формування збалансованих раціонів, які сприяють досягненню спортивних і профілактичних цілей.

Відстеження прогресу користувачів за допомогою інтерактивних графіків змін ваги, параметрів тіла та досягнень у тренуваннях.

Для досягнення мети необхідно вирішити такі завдання:

1. Розробити дизайн користувацького інтерфейсу, що забезпечить зручну взаємодію користувача з додатком.

2. Інтегрувати алгоритми автоматичного розрахунку КБЖВ та рекомендацій для раціонів.

3. Впровадити функціонал для відслідковування тренувань, включаючи динаміку змін ваги, об'єму тіла та досягнень.

4. Забезпечити можливість адаптації раціону залежно від змін у прогресі користувача.

5. Реалізувати архітектуру бази даних для збереження користувацьких даних та історії прогресу.

Запропоновані задачі відповідають потребам сучасного суспільства, інтегруючи персоналізацію харчування і тренувань для профілактики хронічних захворювань та покращення фізичної форми користувачів.

1.2 Аналіз аналогів програмного продукту

В наш час ідеї переповнюють світ, тому створення чогось абсолютно нового стає викликом. У процесі проєктування застосунків важливо вивчати наявні аналоги, щоб краще зрозуміти вимоги користувачів та сформувані унікальну ціннісну пропозицію для нашого продукту, завдяки якій наше рішення зможе створити конкуренцію на ринку. Для порівняння були обрані наступні застосунки:

1. Trainerize [4].
2. Cronometer [5].
3. FitnessServer

Trainerize — це сучасна онлайн-платформа, створена для тренерів, які займаються розробкою, управлінням і моніторингом тренувальних програм своїх клієнтів. Платформа поєднує в собі функціонал для створення індивідуальних тренувань, комунікації між тренером і клієнтом та відстеження прогресу. Trainerize має веб-версію та мобільний додаток, що робить її зручною як для тренерів, так і для користувачів.

Основними характеристиками та можливостями платформи є створення індивідуальних програм тренувань для клієнтів, враховуючи їхні цілі, рівень фізичної підготовки та доступне обладнання. Інструменти для налаштування включають вибір вправ, встановлення кількості повторень, підходів та часу відпочинку. Користувачі можуть відстежувати свій прогрес через додаток, включаючи зміни ваги, об'ємів тіла та тренувальних досягнень. Тренери мають

доступ до цих даних у реальному часі, що дозволяє коригувати програми за потреби. Інтегрований чат і функція відео-чату дозволяють тренерам підтримувати постійний зв'язок із клієнтами, мотивуючи їх досягати поставлених цілей. Додаток підтримує інтеграцію з популярними трекерами активності. Це дозволяє отримувати додаткові дані про активність клієнтів поза тренуваннями.

Платформа надає базові можливості для моніторингу харчування. Клієнти можуть реєструвати свої страви та отримувати дані про спожиті калорії, але функціонал харчування є обмеженим у порівнянні з платформами, спеціалізованими на дієтології. Trainerize дозволяє тренерам створювати персоналізовані програми та шаблони тренувань для груп клієнтів, що особливо корисно для персональних тренерів.

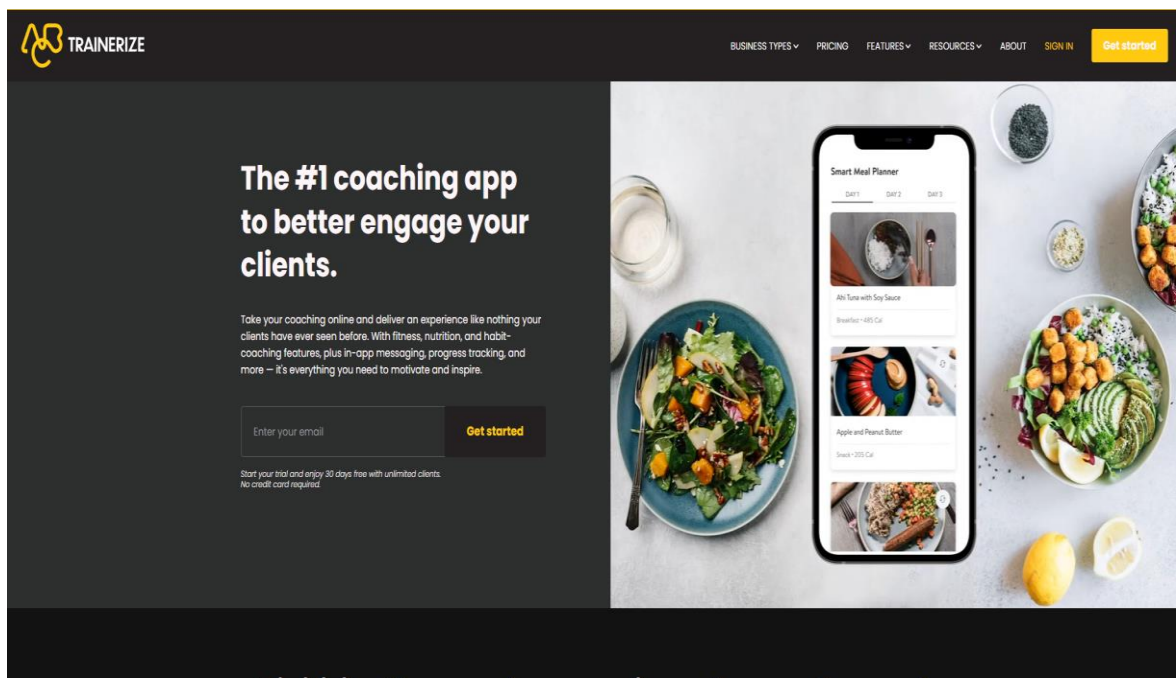


Рисунок 1.1 - Головна сторінка додатка **trainerize.com**

До переваг Trainerize можна віднести комплексне управління тренуваннями, додаток пропонує всі необхідні інструменти для створення, моніторингу та адаптації тренувальних програм. Простий інтерфейс, який буде зручний у використанні як для тренерів, так і для клієнтів. Інтеграція з популярними трекерами активності, що дозволяє збирати більш повні дані про

фізичну активність клієнтів. Постійна комунікація між тренером і клієнтом, за допомогою чату та відеоповідомленням.

Недоліками Trainerize є обмежений функціонал харчування, платформа дозволяє лише базове відстеження калорій та раціону, що не задовольняє потреби користувачів, для яких важливий детальний аналіз харчування. Висока вартість, доступ до повного функціоналу потребує значних фінансових витрат. Фокус на тренерах, додаток більше орієнтований на професійне використання, що обмежує його привабливість для звичайних користувачів, які тренуються самостійно.

Trainerize є потужним інструментом для тренерів, які працюють із клієнтами дистанційно. Основний акцент робиться на управлінні тренуваннями, проте обмежений функціонал харчування може не відповідати потребам користувачів, які прагнуть комплексного підходу до здорового способу життя. У порівнянні з іншими платформами Trainerize забезпечує зручність використання та широкий спектр функцій, але вартість підписки може бути бар'єром для деяких користувачів.

FitnessServer - це платформа, орієнтована на створення та налаштування індивідуальних планів харчування та тренувань. Її основна мета допомогти користувачам ефективно досягати своїх цілей за допомогою персоналізованого підходу. Fitness-Server підходить як для тренерів, які працюють з клієнтами, так і для індивідуальних користувачів, які бажають самостійно управляти своїми програмами.

Основними характеристиками та можливостями платформи є планування харчування: Платформа дозволяє створювати індивідуальні раціони харчування, враховуючи потреби користувача в калоріях, макроелементах та загальні цілі.

Fitness-Server має гнучкий функціонал по забезпеченню планування тренувань. Fitness-Server пропонує функціонал для розробки тренувальних програм із налаштуванням інтенсивності, кількості повторень, підходів та періодів відпочинку.

Є можливість персоналізації програм, тобто всі плани можна адаптувати під індивідуальні потреби користувачів, включаючи обмеження в раціоні.

Аналітика платформи надає користувачам звіти про їхній прогрес, дозволяючи відстежувати динаміку змін ваги, об'ємів тіла, виконання тренувань та споживання калорій.

Достатньо зручний для тренерів. Додаток дає змогу тренерам працювати з декількома клієнтами одночасно, створювати шаблони програм і швидко їх адаптувати.

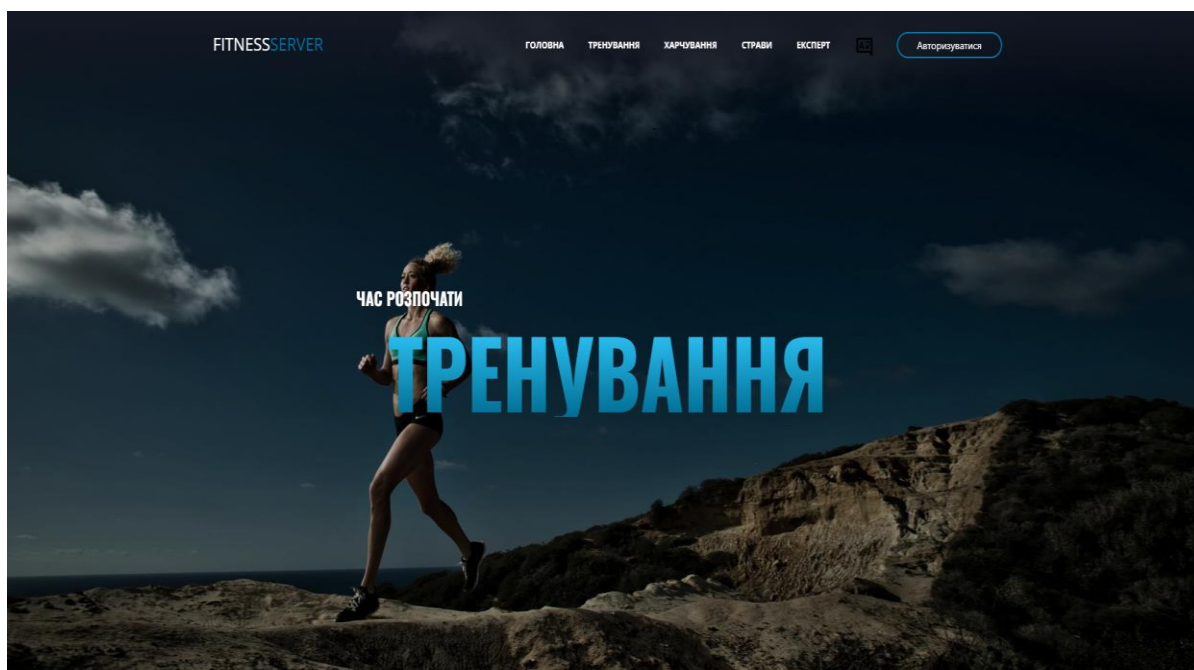


Рисунок 1.2 - Головна сторінка додатка fitness-server.com

Перевагами FitnessServer можна визначити функціонал для планування харчування та тренувань, платформа поєднує обидва напрямки, що робить її універсальним рішенням для управління здоров'ям і фізичною формою. Зручність тренерів, вони отримують потужні інструменти для управління клієнтськими програмами. Можливість персоналізації для користувачів, усі програми адаптуються під конкретні потреби користувача, враховуючи цілі, фізичну підготовку та харчові обмеження.

Недоліками FitnessServer є складність використання для новачків: Інтерфейс і функціонал можуть бути занадто складними для звичайних користувачів без спеціальної підготовки. Більшість функцій доступні лише за

підпискою, що обмежує доступність платформи для широкого кола користувачів.

Таким чином, FitnessServer це платформа, яка поєднує функціонал для створення харчових і тренувальних програм, що робить її корисною для тренерів і професійних користувачів. Проте її складність може відлякати новачків, які шукають більш інтуїтивно зрозуміле рішення. Висока вартість послуг також є бар'єром для користувачів із обмеженим бюджетом. Fitness-Server буде найкращим вибором для професійного використання, але для звичайних користувачів може вимагати адаптації та спрощення функціоналу.

Cronometer - це популярна онлайн-платформа та мобільний додаток, орієнтовані на моніторинг харчування. Додаток дозволяє користувачам відстежувати споживання макро- та мікроелементів, калорій і навіть вітамінів, що робить його одним із найдетальніших інструментів для аналізу раціону. Його аудиторія — це користувачі, які прагнуть покращити своє здоров'я, дотримуючись збалансованого харчування, зокрема спортсмени, дієтологи та люди, які активно стежать за своїм здоров'ям.

Основними характеристиками та можливостями платформи є детальний аналіз харчування: Cronometer дозволяє реєструвати споживання їжі та отримувати повну інформацію про її склад. Додаток відображає дані про калорії, білки, жири, вуглеводи, вітаміни та мінерали.

У Cronometer доступна обширна база продуктів, яка включає як звичайну їжу, так і популярні брендові продукти. Це дозволяє користувачам легко додавати елементи до свого раціону.

Додаток інтегрується з багатьма фітнес-трекерами, що дозволяє поєднувати дані про фізичну активність із харчовими звичками.

Користувачі можуть встановлювати персоналізовані цілі щодо калорійності, споживання макроелементів (білків, жирів, вуглеводів) та вітамінів залежно від своїх потреб (наприклад, схуднення, набір маси або підтримання ваги). Простий та інтуїтивний дизайн полегшує реєстрацію продуктів і перегляд даних.

Додаток дозволяє вносити дані про вагу, активність, рівень глюкози, холестерину тощо, забезпечуючи комплексний підхід до моніторингу здоров'я.

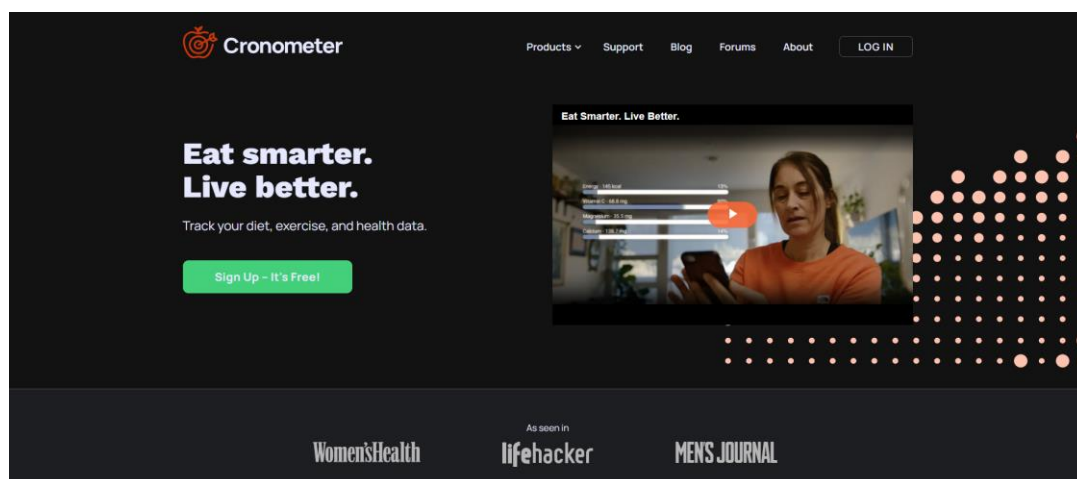


Рисунок 1.3 - Головна сторінка додатка **Cronometer**

До переваг Cronometer входить детальний аналіз макро- і мікроелементів. Cronometer є одним із небагатьох додатків, що пропонує повний аналіз раціону з урахуванням вітамінів та мінералів. Велика база продуктів, яка включає як загальні продукти, так і брендовані, що полегшує користувачам додавання своїх улюблених страв. Додаток доступний кількома мовами, що розширює його аудиторію. Висока зручність для дієтологів і спортсменів. Додаток дозволяє професіоналам та ентузіастам отримувати точну інформацію для досягнення своїх цілей.

Основними недоліками Cronometer є відсутність функціоналу для тренувань, платформа не надає можливості створення або моніторингу тренувальних програм, що обмежує її використання для комплексного аналізу здоров'я. Присутні обмеження безкоштовної версії. Основні функції доступні безкоштовно, але для детальнішого аналізу та додаткових налаштувань потрібна преміум-підписка.

Таким чином, Cronometer — це потужний інструмент для аналізу раціону, що підходить для користувачів, які потребують детального моніторингу харчування. Його основна перевага — це глибокий аналіз мікроелементів, що робить його популярним серед спортсменів, людей із особливими потребами в харчуванні та дієтологів. Проте відсутність функціоналу для тренувань обмежує

його використання як комплексного рішення. Для тих, хто шукає зручний і точний спосіб відстеження харчування, Cronometer стане чудовим вибором, але для повного управління здоров'ям може знадобитися додатковий функціонал.

Таблиця 1.1

Характеристика	Cronometer	FitnessServer	Trainerize	Власний додаток
Цільова аудиторія	Люди, які хочуть слідкувати за раціоном харчування	Тренери та клієнти	Персональні тренери та їх клієнти	Спортсмени, які слідкують за харчуванням та тренуваннями
Функціонал харчування	Відстеження калорійності, макро- та мікроелементів	Не включено	Обмежений	Автоматична генерація раціону, контроль макроелементів
Функціонал тренувань	Немає	Планування тренувань	Комплексне управління тренуваннями	Базовий функціонал для тренувань
Персоналізація	Частково (налаштування цілей)	Інструменти для тренерів	Індивідуальні тренувальні програми	Повна персоналізація харчування та тренувань
Інтеграція з трекерами	Підтримує	Підтримує	Немає	Інтеграція з трекерами
Складність інтерфейсу	3/5	2/5	3/5	2/5
Основні переваги	Зручний для тренерів, інтеграція з пристроями	Детальний аналіз харчування, велика база продуктів	Комбінований підхід харчування та тренувань	Детальний аналіз харчування і тренувань. Комбінований підхід до аналізу
Основні недоліки	Обмежений функціонал харчування, висока вартість	Відсутність тренувального функціоналу, преміум-доступ	Складність використання для новачків, платний доступ	Відсутність підключення сторонніх пристроїв.

Проаналізувавши аналоги додатків для аналізу харчування та тренувань, було виявлено, що більшість із аналогів представляють схожий функціонал: відстеження калорійності, тренувального прогресу та створення індивідуальних планів. Проте функціонал орієнтований здебільшого на вузьку аудиторію, тобто лише на тренерів або харчування.

Судячи з представлених аналогів, основними функціями системи повинні бути: простий і привабливий інтерфейс, автоматизація процесів планування

харчування, персоналізація даних під потреби користувача та можливість відстеження результатів у зручному форматі.

Загалом, вибір MVC для створення додатка є раціональним, адже цей підхід забезпечує чітке розділення логіки, гнучкість, тестованість і зручність розширення, що є важливими аспектами для розробки сучасних інформаційних систем.

1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для створення вебсайту, доцільно застосувати: мову розмітки HTML, каскадні таблиці стилів CSS, мову програмування JavaScript, бібліотеку jQuery, серверну мову PHP, вебсервер Apache, базу даних MySQL та редактор коду JetBrains PhpStorm.

HTML (HyperText Markup Language) - це основа більшості вебсайтів, яка використовується для створення структури сторінок, відображення тексту, зображень і базового контенту. HTML виступає каркасом для розробки функціонала та інтерфейсу користувача[4].

CSS (Cascading Style Sheets) - мова стилів, що використовується разом із HTML для візуального оформлення вебсторінок. CSS дозволяє задавати стиль тексту, кольори, розташування елементів та загальну естетику сторінок. Її інтеграція з HTML забезпечує комплексний підхід до створення привабливого дизайну[5].

JavaScript - мова програмування, яка використовується переважно у браузерних додатках. Являється мовою з ООП і використовується для виконання скриптів які своєю чергою допомагають «оживити» сайт [6].

jQuery — популярна JavaScript-бібліотека, яка спрощує роботу з DOM, обробку подій, анімацію та AJAX-запити. Її використання забезпечує прискорення розробки та полегшує створення інтерактивних компонентів на вебсторінці.

PHP (Hypertext Preprocessor) - серверна мова програмування, яка забезпечує логіку роботи додатка. PHP дозволяє обробляти дані форм, взаємодіяти з базою даних та створювати динамічний контент. Використання PHP у версії 8.2 забезпечує високу продуктивність, підтримку сучасних стандартів та наявність великої кількості бібліотек і модулів. Актуальна версія - 8.2 [7].

JetBrains PhpStorm - середовище розробки, з напрямом на розробку на мові PHP. Середовище, яке допомагає працювати з кодом за допомогою інтелектуального редактора, рефакторингу, підтримки генераторів і поліпшення синтаксису, підтримка фреймворків на PHP, гнучке налаштування плагінів, візуальна стилізація під користувача, редактор мови розмітки (HTML, CSS), слідування за стилями коду (PSR1/PSR2), потужні інструменти для роботи з базами даних [7].

Після створення можливих реляційних баз даних (MySQL, PostgreSQL) рішення було прийнято на базі MySQL, оскільки MySQL відрізняється високою швидкістю та ефективністю, що дозволяє оптимально обробляти навіть більші обсяги даних [13]. Вона підтримує розподілені транзакції та має потужну систему кешування для підвищення продуктивності. Крім того, MySQL має велику кількість користувачів спільноти та широку підтримку, що робить її простою у використанні та вивченні. Однак на ринку є ще одна популярна система — PostgreSQL. Вона відома своєю розширеністю і повною функціональністю. PostgreSQL надає безліч додаткових можливостей, таких як видалення дублікатів, операції з даними JSON і географічні запити. Ця система також має розширені можливості управління доступом і реалізації ролей користувачів.

У порівнянні з іншими СУБД, такими як PostgreSQL, MySQL була обрана через її високу продуктивність і легкість інтеграції в проєкти середнього рівня складності. PostgreSQL, хоча й забезпечує розширені можливості для аналізу даних, більше підходить для проєктів із великим обсягом нестандартних операцій.

Розглянувши варіанти технологій було обрано:

- Мову розмітки документів HTML5;
- стилі CSS3;
- мову програмування JavaScript та бібліотеку jQuery;
- фреймворк Laravel 11 (PHP8.2);
- середовище керування базами даних MySQL;
- Windows 10, Linux, MacOS;
- 32-розрядний або 64-розрядний процесор з тактовою частотою 1 ГГц та

вище;

- 4 Гбайти ОЗП;
- 4 Гбайти, за можливості більше вільного місця на жорсткому диску;
- комп'ютер користувача виконує роль сервера.

1.4 Аналіз методів розрахунку ключових параметрів для роботи додатка

У даному розділі описано алгоритми та формули, які використовуються для розрахунку базових параметрів у додатку. Ці алгоритми забезпечують автоматизацію процесів оцінки калорійних норм, співвідношення макроелементів (КБЖВ), відсотку жирової тканини, а також рекомендацій щодо вітамінів і мінералів.

Розрахунок відсотка жирової тканини

Першим з розглянутих методів стане розрахунок відсотка жирової тканини. Метод розрахунку відсотка жирової тканини базується на антропометричних замірах: обхвату талії, шиї, стегон (для жінок) та росту. Він дозволяє отримати орієнтовні дані про склад тіла користувача з похибкою в межах 2–4% порівняно з результатами, отриманими за допомогою спеціалізованого обладнання.

1. Для чоловіків:

$$\% \text{жиру} = 86.01 * \log_{10}(\text{талія(см)} - \text{шия(см)}) - 70.041 * \log_{10}(\text{зріст(см)}) + 30.76$$

2. Для жінок:

$$\begin{aligned} \% \text{жиру} = & 163.205 * \log_{10}(\text{талія(см)} + \text{стегна(см)} - \text{шия(см)}) - 70.041 \\ & * \log_{10}(\text{зріст(см)}) + 30.76 \end{aligned}$$

Результат розрахунку відсотка жирової тканини дозволяє оцінити залежність змін у жировій масі та м'язах, що є важливим для аналізу ефективності тренувань і раціону.

Розрахунок базального метаболізму (BMR)

Базальний метаболізм (BMR) визначає кількість енергії, необхідної для підтримки основних функцій організму в стані спокою (дихання, робота серця тощо). У додатку використовуються три загальновизнані формули для підвищення точності розрахунків:

1. Гарріса-Бенедикта:

- BMR для чоловіка:

$$BMR = 88.362 + (13.397 * \text{вага(кг)}) + (4.799 * \text{зріст(см)}) - (5.677 * \text{вік})$$

- BMR для жінок:

$$BMR = 447.593 + (9.247 * \text{вага(кг)}) + (3.098 * \text{зріст(см)}) - (4.330 * \text{вік})$$

2. Міффліна-Сен Жеора:

- BMR для чоловіка:

$$BMR = (10 * \text{вага(кг)}) + (6.25 * \text{зріст(см)}) - (5 * \text{вік}) + 5$$

- BMR для жінок:

$$BMR = (10 * \text{вага(кг)}) + (6.25 * \text{зріст(см)}) - (5 * \text{вік}) - 161$$

3. Кетча-МакАрдла (залежно від м'язової маси):

$$BMR = 370 + (21.6 * (1 - (\frac{\% \text{жиру}}{100})))$$

Середнє значення BMR, отримане з цих формул, забезпечує більш точний розрахунок.

Розглянемо графічне відображення залежності BMR від ваги для кожної формули.

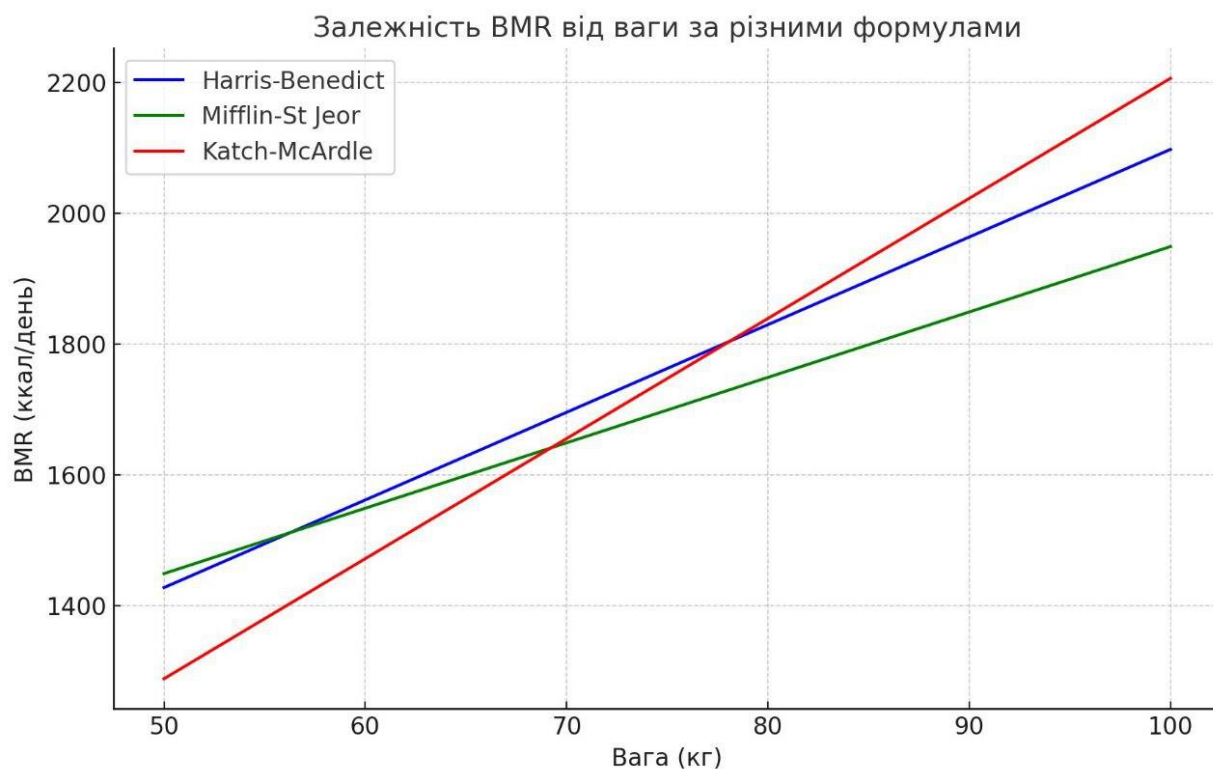


Рисунок 1.4 – Графік зміни BMR

Добова норма калорій (TDEE)

Добова норма калорій розраховується на основі BMR з урахуванням рівня фізичної активності (коефіцієнт активності):

Базова норма:

$$\text{TDEE} = \text{BMR} * \text{коефіцієнт активності.}$$

Де:

- 1.2 – мінімальна активність (сидячий спосіб життя);
- 1.375 – легка активність (легкі вправи 1–3 рази на тиждень);
- 1.55 – помірна активність (вправи 3–5 разів на тиждень);
- 1.725 – висока активність (інтенсивні вправи 6–7 разів на тиждень);
- 1.9 – дуже висока активність (фізична праця або часті тренування).

З отриманим результатом ми працюємо розраховуючи норму для вибраної цілі користувача.

При зниженні ваги ми зменшуємо TDEE на 22% калорій, та збільшуємо на 22% при наборі ваги. Необхідні не великі зміни у базових калорій для безпечної

зміни ваги та підтримання результатів. Розрахунок БЖВ відбувається індивідуально враховуючи ціль користувача та персональні дані.

Персоналізовані рекомендації

Окрім розрахунків калорійності, у додатку реалізовано рекомендації щодо споживання вітамінів і мінералів, адаптовані до статі, віку та фізичної активності користувача.

Вітаміни та мінерали відіграють ключову роль у підтримці загального здоров'я та метаболічних процесів організму. Рекомендовані добові норми споживання цих нутрієнтів залежать від віку, статі та фізіологічного стану людини.

Рекомендовані добові норми споживання вітамінів та мінералів

Згідно з даними Офісу дієтичних добавок Національного інституту здоров'я США[1], рекомендовані добові норми споживання (RDA) для основних вітамінів та мінералів варіюються залежно від віку та статі. Наприклад, для дорослих чоловіків і жінок віком 19–50 років рекомендовані норми споживання деяких ключових нутрієнтів такі:

- Вітамін С: 90 мг/день для чоловіків; 75 мг/день для жінок.
- Вітамін D: 600 МО/день для обох статей.
- Кальцій: 1,000 мг/день для обох статей.
- Залізо: 8 мг/день для чоловіків; 18 мг/день для жінок.

Ці значення можуть змінюватися залежно від віку, фізіологічного стану (наприклад, вагітність) та рівня фізичної активності.

Адаптація споживання вітамінів та мінералів при зміні ваги

Під час процесу зниження або набору ваги важливо забезпечити достатнє споживання вітамінів та мінералів, оскільки обмеження калорій може призвести до дефіциту деяких нутрієнтів. Дослідження показують, що дефіцит вітамінів групи В, вітаміну D та магнію може негативно впливати на метаболізм та процеси зниження ваги.

Тому під час корекції ваги рекомендується:

Збалансоване харчування: включати в раціон продукти, багаті на необхідні вітаміни та мінерали, щоб запобігти їх дефіциту. Додатковий прийом добавок: за рекомендацією лікаря або дієтолога, особливо якщо існує ризик дефіциту певних нутрієнтів.

Варто зазначити, що надмірне споживання деяких вітамінів та мінералів може бути шкідливим. Наприклад, Європейське агентство з безпеки харчових продуктів встановило верхню допустиму межу споживання вітаміну B6 на рівні 12.5 мг/день для дорослих, оскільки надмірне споживання може призвести до периферичної нейропатії.

Таким чином, під час планування дієти для зниження або набору ваги важливо враховувати не лише калорійність, але й забезпечення організму необхідними вітамінами та мінералами для підтримки оптимального здоров'я.

Висновки до розділу 1

У даному розділі було здійснено всебічний аналіз актуальності, задачі та методів, необхідних для створення інформаційного додатка, спрямованого на інтеграцію функціоналу моніторингу харчування та тренувального прогресу. Проведений аналіз включав:

Постановку задачі розробки додатка, з визначенням ключових функціональних вимог. Огляд існуючих аналогів програмних продуктів з їх перевагами та недоліками. Детальне дослідження алгоритмів і методів розрахунку базових параметрів.

З огляду на глобальні виклики, пов'язані з ростом рівня ожиріння та недостатньою фізичною активністю, було визначено необхідність створення інструменту, який допомагав би користувачам у:

Автоматизації розрахунку індивідуальних норм калорій, білків, жирів і вуглеводів (КБЖВ).

Формуванні збалансованих раціонів, адаптованих до цілей (зниження ваги, набір маси, підтримка форми).

Моніторингу складу тіла (відсотка жирової тканини) та аналізу прогресу тренувань.

Зазначені завдання забезпечують комплексний підхід до підтримки здорового способу життя та профілактики хронічних захворювань, таких як ожиріння і діабет. Таким чином, було сформовано чітке уявлення про архітектуру та інструменти, які необхідні для створення функціонального додатка, що дозволить якісно вирішувати поставлені задачі та відповідатиме потребам користувачів.

Оцінка аналогів дала змогу сформулювати ключові вимоги до розробки додатка, зокрема: створення універсального інструменту, який інтегрує аналіз харчування, моніторинг тренувань і персоналізовані рекомендації.

Загалом, проведений аналіз підтверджує актуальність створення додатка, який інтегрує сучасні наукові підходи до розрахунків і враховує потреби користувачів у персоналізації. Виявлені недоліки існуючих програмних рішень стали основою для формування вимог до розробки інноваційного інструменту. Запропоновані методи та алгоритми забезпечують високу точність розрахунків і адаптацію до індивідуальних параметрів кожного користувача.

Отримані результати стануть базою для реалізації функціоналу додатка, що дозволить користувачам досягати своїх цілей у харчуванні та тренуваннях ефективніше.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ІНФОРМАЦІЙНОГО ДОДАТКУ

2.1 Визначення варіантів використання та структури системи

Інформаційний додаток має за мету спрощення та автоматизацію процесів, пов'язаних із аналізом харчування та тренувальних показників. Основними завданнями додатка є автоматизація створення індивідуальних планів харчування та тренувань, відстеження прогресу користувачів у досягненні цілей (наприклад, схуднення чи набору м'язової маси), а також інтеграція сучасних інструментів для аналізу даних.

Додаток забезпечує можливість ефективного моніторингу змін у раціоні, аналізу макро- та мікроелементів і взаємодії з користувачем через зручний та інтуїтивний інтерфейс. Основною метою є покращення доступності та якості інструментів для персоналізованого підходу до здорового способу життя.

Вимоги користувачів.

На рисунку 2.1 відображено діаграму активності не авторизованого користувача. Гостю доступні наступні функції:

1. До інформації з головної сторінки;
2. до взаємодії з калькуляторами;
3. до перегляду довідника з продуктів та вправ;
4. до реєстрації або авторизації на сайті.

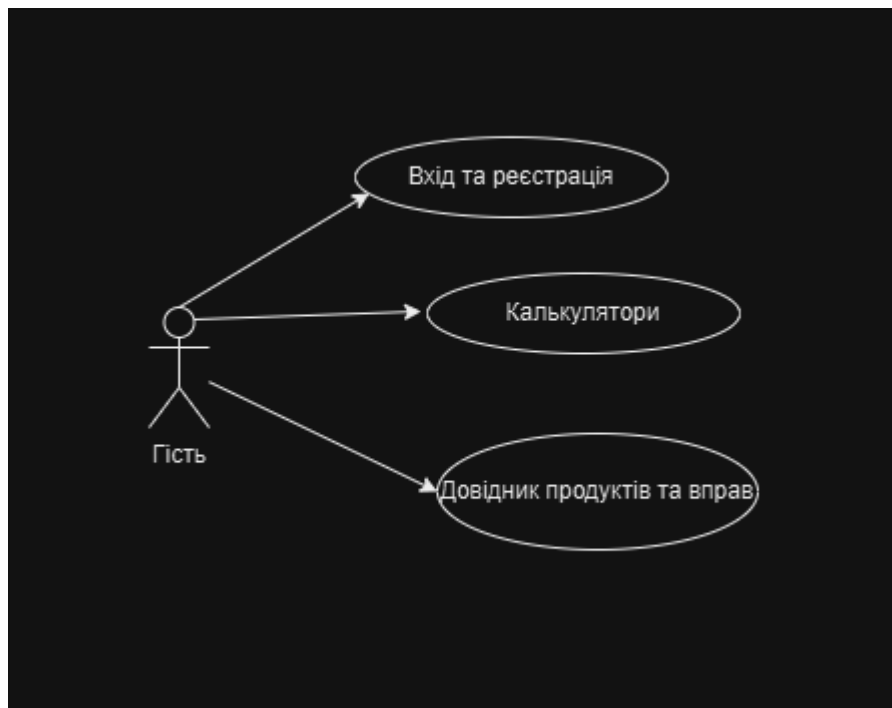


Рисунок 2.1 – Діаграма активності гостя

На рисунку 2.2 діаграма зареєстрованого користувача. Авторизований користувач має доступ до наступних функцій:

1. До використання калькуляторів;
2. до сторінки з персональним щоденником харчування;
3. до списку бажаних продуктів;
4. до графіку ймовірної зміни ваги;
5. до особистого кабінету;
6. до графіку зі змінами;
7. до персонального щоденника тренувань;
8. до списку проблем зі здоров'ям;
9. до графіка з результатами тренувань;
10. вихід.

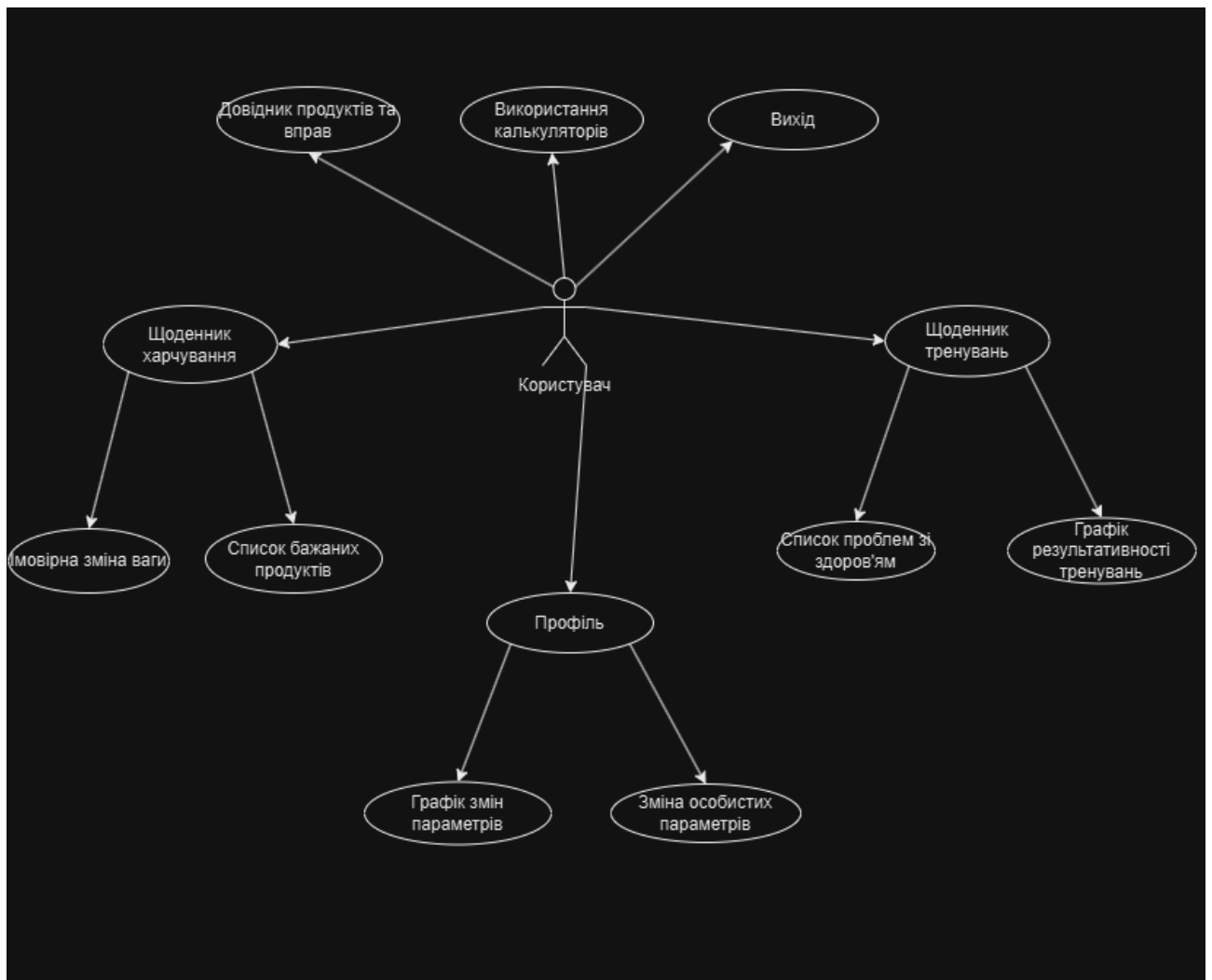


Рисунок 2.2 - Діаграма використання авторизованого користувача

Функціональні вимоги:

1. Авторизація. Для користувачів необхідна можливість реєстрації та подальшого входу в аккаунт.
2. Додавання, видалення, та редагування щоденника харчування та тренувань.
3. Система повинна зберігати конфіденційність користувачів.
4. Система повинна коректно та автоматично відображати необхідну інформацію.

Нефункціональні вимоги:

1. Сприйняття:

- Час, потрібний для початку роботи - 2 хв.
- Час відповіді системи для звичайних запитів не повинен перевищувати

0.340 мс, а для більш складних запитів - 0.840 мс.

– Інтерфейс повинен бути зрозумілим для користувачів.

2. Надійність:

– Середній час безперервної роботи - 55 год.

– Максимальна норма помилок та дефектів в роботі програми - 1 помилка на 3000 запитів користувача.

3. Швидкість. Сайт повинен швидко виконувати запити користувачів.

4. Можливості використання:

– Масштабування - після додавання функціонала не повинно бути фатальних помилок.

Аналіз функціональних вимог дозволив виявити наступні ключові елементи, що забезпечать розробку програмного комплексу для цієї системи. На рис. 2.3 наведена діаграма класів, що представляє класи та контролери системи. Детальна структура представлень розглянута у третьому розділі.

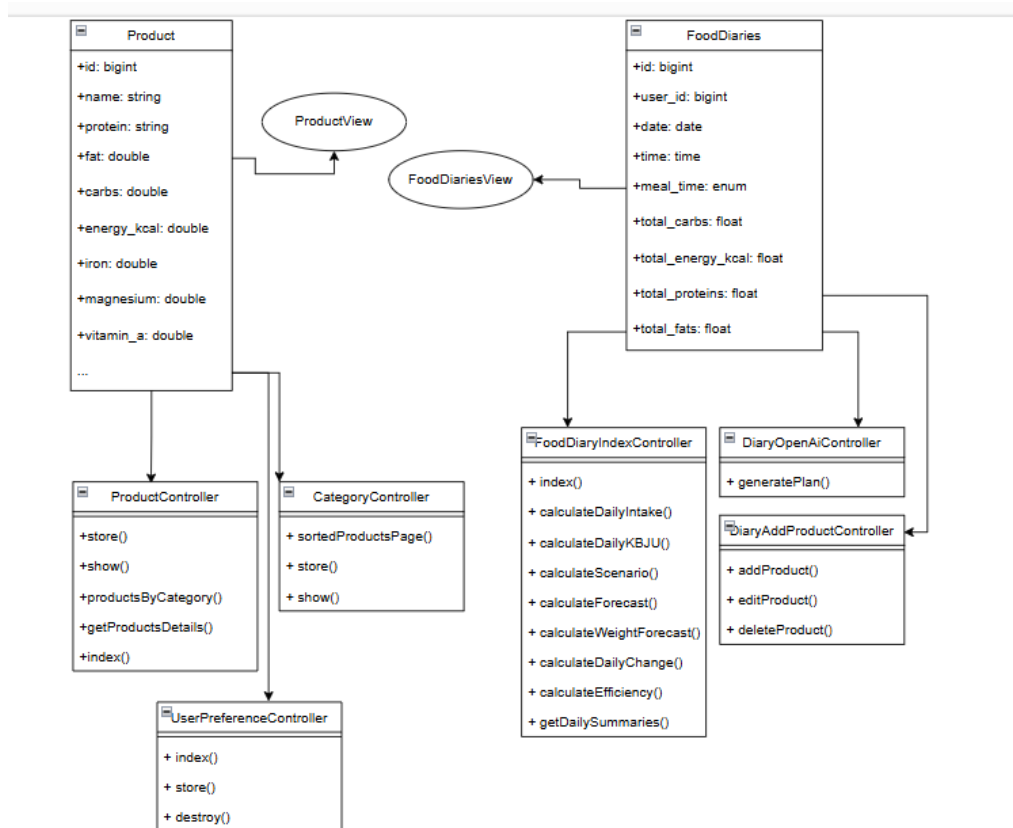


Рисунок 2.3 - Діаграма класів

На рисунку 2.3 відображені основні класи для роботи з харчуванням. Основним з них є клас Product. Клас є основним у роботі проекту, містить властивості та методи для їх обробки. Реалізує операції з продуктами, такі як: додавання продукту в базу даних, відображення конкретного продукту, перегляд детального складу продукту, сортування продуктів за категоріями. Це базовий клас, який наслідується контролерами для роботи з харчуванням. Дані з Product використовується у контролерах категорій, бажаних продуктів користувача, щоденника харчування.

Методи :

- store () – додавання продукту;
- show() – перегляд продукту;
- getProductsDetails() – детальна інформація про продукт;
- index() – перегляд всіх продуктів;
- productsByCategory() – сортування продуктів за категорією

Наступним класом на малюнку 2.3 є клас FoodDiary, що відповідає за роботу щоденника харчування. Клас має в собі декілька контролерів, які виконують функції CRUD з продуктами раціону, генерацією харчування за допомогою ШІ, відображення раціону та добових норм нутрієнтів користувача. Є головним класом при роботі з щоденником, для успішного виконання всіх властивостей і методів. Для роботи класу необхідні зв'язки з моделлю user для отримання необхідних базових норм нутрієнтів, КБЖВ. Для успішної генерації раціону за допомогою ШІ необхідно отримати список бажаних продуктів користувача. Для ручного створення раціону необхідний зв'язок з базовим класом Product.

Методи:

- index() –показ щоденника;
- generatePlan() – генерація харчування ШІ, на 1 день;
- addProduct() – додавання продукту до щоденника;
- editProduct () – редагування продукту в щоденнику;
- deleteProduct () –видалення продукту зі щоденника;

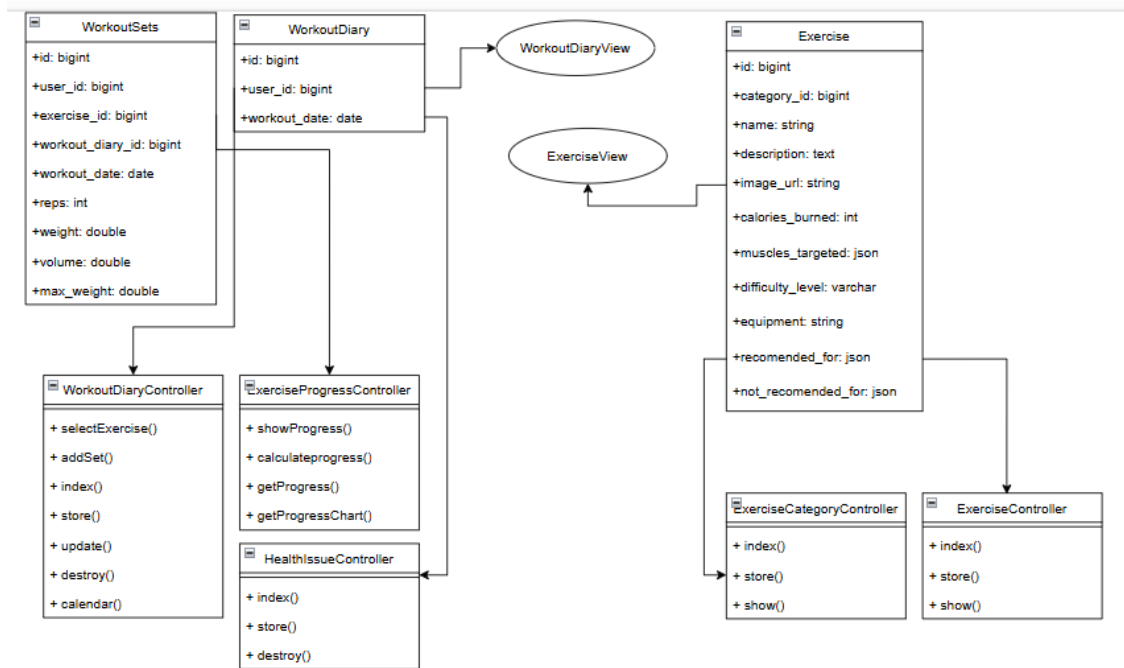


Рисунок 2.4 – Діаграма класів

Розглянемо базовий клас для роботи з вправами – exercise, зрисунка 2.4. Клас є основним для роботи з вправами, містить методи для їх обробки. Виконує дії з вправами такі як: додавання в базу даних, редагування та видалення. Клас використовується при роботі всіх методів та властивостей щоденника тренувань, для створення тренування, для обчислення прогресу користувача та обчислення прогресу користувача по конкретній вправі.

Методи:

- index() – перегляд всіх вправ категорії;
- show() – перегляд конкретної вправи;
- store() – додавання вправи;

На рисунку 2.4 відображено класи для роботи з вправами та щоденником тренувань. Поєднані між собою класи WorkoutDiary та WorkoutSets, виконують усі властивості та методи необхідні для коректної роботи щоденника тренувань, такі як: crud тренування, розрахунок прогресу по конкретних вправах та загального прогресу користувача. Клас WorkoutDiary є базовою та початковим вектором для початку роботи з щоденником тренувань, відбувається створення та збереження добового тренування. В той час, як клас WorkoutSets є основним

класом для наповнення тренування та відслідковування прогресу користувача. За допомогою методів цього класу відбувається додавання вправ, підходів та повторень, тому для роботи класу необхідно мати зв'язок з базовим класом `exercise`.

Методи:

- `index()` – перегляд щоденника тренувань;
- `addSet()` – додавання підходу;
- `store()` – створення щоденника;
- `update()` – редагування тренування;
- `destroy()` – видалення тренування;
- `showProgress()` – перегляд прогресу.
- `calculateProgress()` – розрахунок прогресу

2.2 Розробка бази даних системи

Для збереження та подальшого використання даних було розроблено таблиці бази даних для кожного зі змінюваних сутностей, таких як:

- `users` – таблиця в якій знаходиться інформація про користувачів;
- `exercise` – таблиця в якій знаходиться інформація про вправи;
- `categories` – таблиця в якій знаходиться інформація про категорії продуктів;
- `exercise_categories` – таблиця в якій знаходиться інформація про категорії вправ;
- `exercise_progress` – таблиця в якій зберігається прогрес користувачів;
- `food_diaries` – таблиця для збереження щоденника харчування;
- `food_diaries_items` – таблиця в якій зберігаються продукти і інформація про спожите КБЖВ;
- `health_issues` – таблиця проблем зі здоров'ям;
- `products` – таблиця в якій знаходиться інформація про продукти;
- `user_health_issues` – таблиця яка пов'язує `users` та `health_issues`;

- user_mensurements– таблиця в якій зберігається інформація про заміри користувача;
 - user_product_preference – таблиця в зберігаються бажані продукти користувача;
 - workout_diaries – таблиця, яка зберігає щоденники тренувань;
 - workout_sets – таблиця, яка зберігає дані про тренування.
- Тож, база даних складається з 14 таблиць, розглянемо детально кожную.

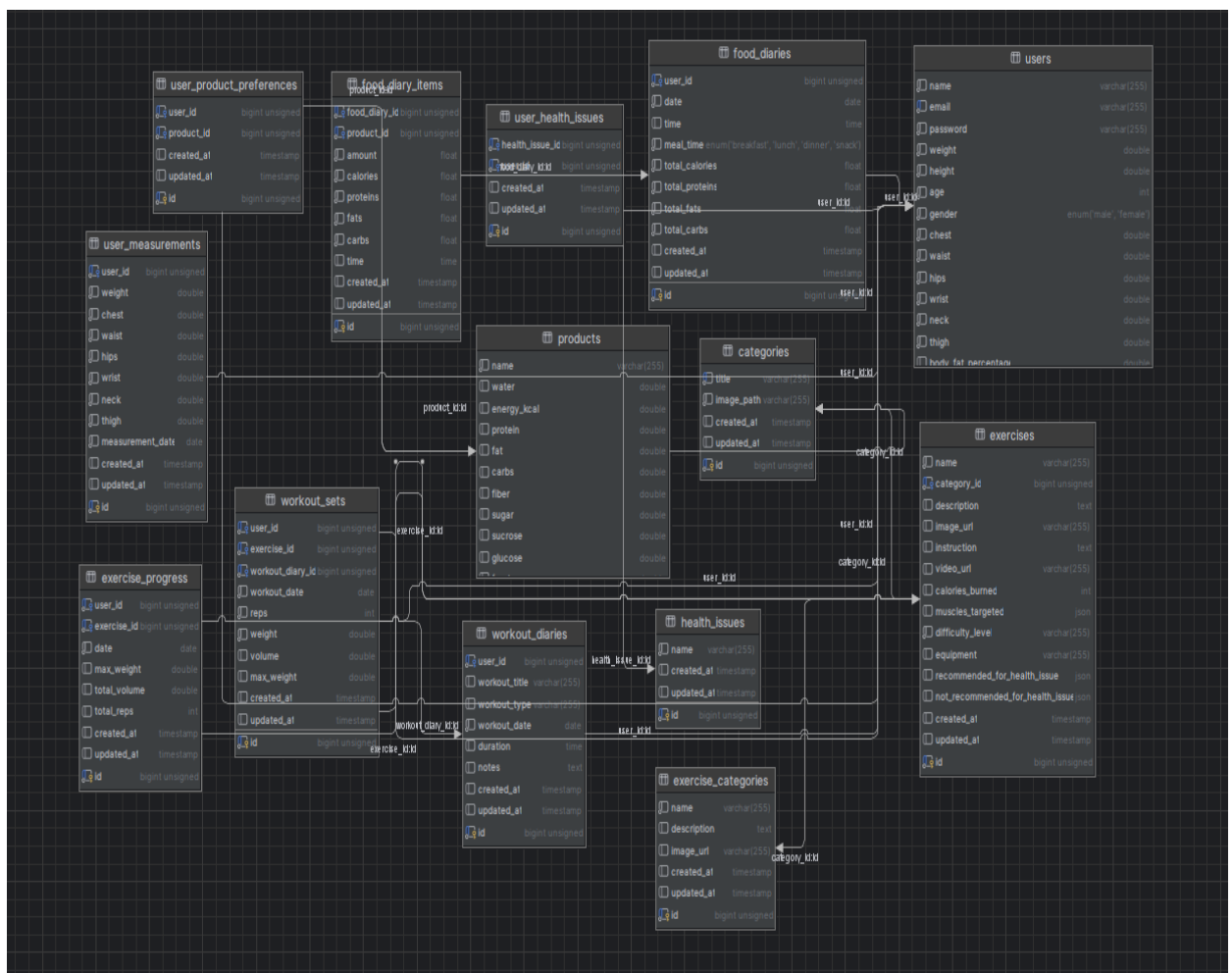


Рисунок 2.5- Діаграма бази даних

Нижче буде наведено опис кожної з таблиць бази даних (див. табл. 2.1 - 2.14)

Таблиця user_product_preferences представляє собою список обраних користувачем продуктів. Список необхідний для генерації харчування за допомогою ШІ. Звідна таблиця для роботи якої потрібні зв'язки з таблицями users та product.

Таблиця 2.1

user_product_preferences		
Поле	Тип	Призначення
Id	BIGINT, PK	Унікальний ідентифікатор
user_id	BIGINT, FK	Унікальний ідентифікатор користувача
product_id	BIGINT, FK	Унікальний ідентифікатор продукту

Таблиця user_measurements необхідна для збереження оновлених замірів користувача. Заміри використовуються для зміни BMR та побудови графіків змін. Для коректної роботи необхідний зв'язок тільки з таблицею users.

Таблиця 2.2

user_measurements		
Поле	Тип	Призначення
id	Uniqueidentifier, PK	Унікальний ідентифікатор
user_id	BIGINT,FK	Унікальний ідентифікатор користувача
weight	FLOAT	Вага
chest	FLOAT	Замір грудей
waist	FLOAT	Замір талії
hips	FLOAT	Замір бедер
wrist	FLOAT	Замір зап'ястка
neck	FLOAT	Замір шиї
thigh	FLOAT	Замір бедра
measurement_date	DATE	Дата оновлення замірів

Таблиця exercise_progress зберігає дані користувача для створення та відображення графіків прогресу конкретної вправи користувача. Для роботи необхідні зв'язки з таблицями users та exercises.

Таблиця 2.3

exercise_progress		
Поле	Тип	Призначення
Id	BIGINT, PK	Унікальний ідентифікатор
user_id	BIGINT, FK	Унікальний ідентифікатор користувача
exercise_id	BIGINT, FK	Унікальний ідентифікатор вправи
date	DATE	Дата тренування
max_weight	FLOAT	Максимальна вага
total_volume	FLOAT	Загальний об'єм
total_reps	INT	Загально повторень

Таблиця food_diary_items зберігає дані про додані до добового раціону продуктів та розраховує отримані КБЖВ. Для роботи потрібні зв'язки з таблицями food_diaries та products.

Таблиця 2.4

food_diary_items		
Поле	Тип	Призначення
Id	Uniqueidentifier, PK	Унікальний ідентифікатор
food_diary_id	BIGINT, FK	Унікальний ідентифікатор щоденника
product_id	BIGINT, FK	Унікальний ідентифікатор продукту
amount	FLOAT	Кількість
calories	FLOAT	Калорії
proteins	FLOAT	Білки
fats	FLOAT	Жири
carbs	FLOAT	Вуглеводи
time	TIME	Час

Таблиця user_health_issues є звідною таблицею для збереження та подальшого відображення проблем зі здоров'ям користувача. Для роботи необхідні зв'язки з таблицями users та health_issues.

Таблиця 2.5

user_health_issues		
Поле	Поле	Поле
id	BIGINT, RK	Унікальний ідентифікатор
user_id	BIGINT, FK	Унікальний ідентифікатор користувача
health_issue_id	BIGINT, FK	Унікальний ідентифікатор проблеми

Таблиця products є основною таблицею проекту в якій зберігається детальний опис продуктів.

Таблиця 2.6

products		
Поле	Тип	Призначення
id	BIGINT, PK	Унікальний ідентифікатор
name	STRING	Назва продукту
water	FLOAT	Вміст води
energy_kcal	FLOAT	Калорійність
protein	FLOAT	Вміст білків
fat	FLOAT	Вміст жирів
carbs	FLOAT	Вміст вуглеводів
fiber	FLOAT	Вміст клітковини
sugar	FLOAT	Загальний вміст цукрів
sucrose	FLOAT	Вміст сахарози
glucose	FLOAT	Вміст глюкози
fructose	FLOAT	Вміст фруктози
lactose	FLOAT	Вміст лактози
maltose	FLOAT	Вміст мальтози
galactose	FLOAT	Вміст галактози
starch	FLOAT	Вміст крохмалю
calcium	FLOAT	Вміст кальцію
iron	FLOAT	Вміст заліза

Продовження таблиці 2.6

magnesium	FLOAT	Вміст магнію
phosphorus	FLOAT	Вміст фосфору
potassium	FLOAT	Вміст калію
sodium	FLOAT	Вміст натрію
zinc	FLOAT	Вміст цинку
copper	FLOAT	Вміст міді
manganese	FLOAT	Вміст марганцю
selenium	FLOAT	Вміст селену
thiamine	FLOAT	Вміст тіаміну (вітамін B1)
riboflavin	FLOAT	Вміст рибофлавіну (вітамін B2)
niacin	FLOAT	Вміст ніацину (вітамін B3)
pantothenic_acid	FLOAT	Вміст пантотенової кислоти (B5)
vitamin_b6	FLOAT	Вміст вітаміну B6
folic_acid	FLOAT	Вміст фолієвої кислоти (B9)
vitamin_b12	FLOAT	Вміст вітаміну B12
vitamin_a	FLOAT	Вміст вітаміну A
vitamin_e	FLOAT	Вміст вітаміну E
saturated_fatty_acids	FLOAT	Вміст насичених жирних кислот
monounsaturated_fatty_acids	FLOAT	Вміст мононенасичених жирних кислот
polyunsaturated_fatty_acids	FLOAT	Вміст поліненасичених жирних кислот
category_id	BIGINT, FK	Зв'язок із таблицею категорій

Таблиця workout_diaries звідна таблиця для роботи щоденника тренувань. Для роботи необхідний зв'язок з таблицею users.

Таблиця 2.7

workout_diaries		
Поле	Тип	Призначення
id	BIGINT, PK	Унікальний ідентифікатор
user_id	BIGINT, FK	Зв'язок із таблицею користувачів
workout_date	DATE	Дата тренування

Таблиця workout_sets необхідна для збереження інформації про виконанні підходи та повторення у щоденнику тренувань. Для роботи потрібні зв'язки з таблицями users, exercises, workout_diaries.

Таблиця 2.8

workout_sets		
Поле	Тип	Призначення
id	BIGINT, PK	Унікальний ідентифікатор
user_id	BIGINT, FK	Зв'язок із таблицею користувачів
exercise_id	BIGINT, FK	Зв'язок із таблицею вправ
workout_diary_id	BIGINT, FK	Зв'язок із таблицею тренувальних щоденників
workout_date	DATE	Дата тренування
reps	INTEGER	Кількість повторень
weight	FLOAT	Вага (кг)
volume	FLOAT	Загальний обсяг
max_weight	FLOAT	Максимальна вага

Таблиця health_issues необхідна для збереження списку доступних для вибору проблем зі здоров'ям.

Таблиця 2.9

health_issues		
Поле	Тип	Призначення
id	BIGINT, PK	Унікальний ідентифікатор
name	STRING	Назва проблеми зі здоров'ям
created_at	TIMESTAMP	Дата створення запису
updated_at	TIMESTAMP	Дата оновлення запису

Таблиця users необхідна для збереження та оновлення інформації про користувача.

Таблиця 2.10

users		
Поле	Тип	Призначення
id	BIGINT, PK	Унікальний ідентифікатор
name	STRING	Ім'я користувача
email	STRING	Електронна пошта
password	STRING	Пароль
weight	FLOAT	Вага користувача
height	FLOAT	Зріст користувача
age	INTEGER	Вік користувача
gender	ENUM	Стать (male, female)
chest	FLOAT	Обхват грудей
waist	FLOAT	Обхват талії
hips	FLOAT	Обхват стегон
wrist	FLOAT	Обхват зап'ястя

Продовження таблиці 2.10

neck	FLOAT	Обхват шиї
thigh	FLOAT	Обхват стегна
body_fat_percentage	FLOAT	Відсоток жиру в організмі
goal	ENUM	Ціль (weight_loss, weight_gain, weight_maintenance)
activity_coefficient	ENUM	Рівень активності

Таблиця exercise необхідна для збереження списку вправ з детальним описом. Для роботи необхідний зв'язок із таблицею exercise_categories.

Таблиця 2.11

exercise		
Поле	Тип	Призначення
id	BIGINT, PK	Унікальний ідентифікатор
name	STRING	Назва вправи
category_id	BIGINT, FK	Зв'язок із таблицею категорій вправ
description	TEXT	Опис вправи
image_url	STRING	URL до зображення вправи
instruction	TEXT	Інструкція до виконання вправи
video_url	STRING	URL до відео вправи
calories_burned	INTEGER	Кількість спалених калорій
muscles_targeted	JSON	Цільові м'язи
difficulty_level	STRING	Рівень складності (default: "середній")
equipment	STRING	Необхідне обладнання

Продовження таблиці 2.11

recommended_for_health_issues	JSON	Рекомендації для стану здоров'я
not_recommended_for_health_issues	JSON	Протипоказання для стану здоров'я

Таблиця categories необхідна для збереження категорій для продуктів.

Таблиця 2.12

categories		
Поле	Тип	Призначення
id	BIGINT, PK	Унікальний ідентифікатор
title	STRING	Назва категорії (унікальна)
image_path	STRING	Шлях до зображення категорії

Таблиця food_diaries необхідна для створення щоденника харчування та обрахунку отриманого КБЖВ. Для роботи потрібний зв'язок з таблицею users.

Таблиця 2.13

food_diaries		
Поле	Тип	Призначення
id	BIGINT, PK	Унікальний ідентифікатор
user_id	BIGINT, FK	Зв'язок із таблицею користувачів
date	DATE	Дата запису
time	TIME	Час запису
meal_time	ENUM	Час прийому їжі (сніданок, обід, вечеря, перекус)
total_calories	FLOAT	Загальна кількість калорій
total_proteins	FLOAT	Загальна кількість білків
total_fats	FLOAT	Загальна кількість жирів
total_carbs	FLOAT	Загальна кількість вуглеводів

Таблиця exercise_categories необхідна для збереження категорій вправ.

Таблиця 2.14

exercise_categories		
Поле	Тип	Призначення
id	BIGINT, PK	Унікальний ідентифікатор
name	STRING	Назва категорії вправ
description	TEXT	Опис категорії
image_url	STRING	URL до зображення

Реалізувавши базу даних, користувач може з легкістю використовувати всі необхідні функції додатка.

2.3 Проектування та реалізація алгоритмів роботи системи

Головними компонентами в додатку є маніпуляції з продуктами та вправами. Розглянемо алгоритм роботи користувача, який представлено на діаграмі рисунка 2.6:

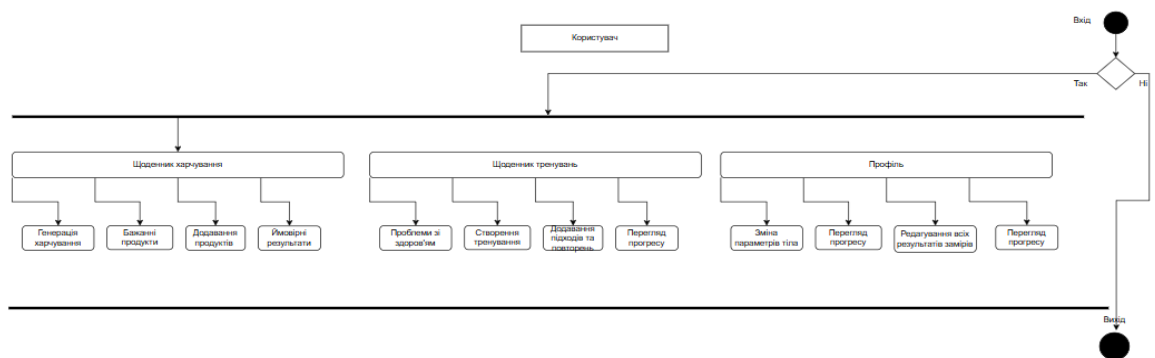


Рисунок 2.6 - Активність користувача

Аутентифікація в обліковому записі. Метою є забезпечення персоналізованого доступу до функцій системи.

Користувач вводить свої облікові дані (електронну пошту та пароль...). Система перевіряє достовірність даних через захищений протокол. У разі успіху користувач отримує доступ до своїх профільних даних.

Реалізація аутентифікації базується на сучасних рекомендаціях з безпеки веб-додатків (OWASP). Застосовуються токени для захисту сесій користувача.

Дії зі щоденником харчування. Додавання продуктів: Користувач обирає продукт із бази даних. Система автоматично розраховує калорії та КБЖВ, інтегруючи дані з алгоритмів (описаних у попередніх підрозділах). Редагування записів. Внесення змін до раніше доданих продуктів або їх кількості. Генерація раціону. Використання алгоритмів персоналізованого підбору продуктів і страв.

Розрахунки калорійності та співвідношення макроелементів базуються на даних ВООЗ, а алгоритм генерації раціонів враховує науково обґрунтовані формули з оптимізації харчування.

Дії зі щоденником тренувань. Додавання вправ. Користувач може створювати індивідуальні тренування, обираючи вправи з каталогу. Перегляд прогресу. Візуалізація змін у виконаних вправах (тривалість, кількість підходів, вага). Редагування тренувань, можливість вносити корективи до запланованих тренувань.

Алгоритми розроблені на основі рекомендацій ACSM (American College of Sports Medicine) для моніторингу фізичного прогресу.

Взаємодія в профілі. Замір параметрів, користувач вносить дані (вага, об'єми тіла), які система аналізує для визначення динаміки змін. Редагування даних, можливість оновлювати заміри чи цілі.

Для аналізу змін використовуються графічні інтерпретації та лінійні моделі прогнозування. Це дозволяє оцінювати прогрес у зручній візуальній формі.

Було розглянути можливі варіанти взаємодії з додатком зі сторони користувача.

2.4 Реалізація функціоналу інформаційного додатку

Для реалізації додатка було вибрано фреймворк Laravel. Laravel використовує розширення залежностей, такі як: маршрути, контролери, моделі

та представлення [16]. Компоненти взаємодіють між собою за допомогою вбудованих методів.

Першим кроком до побудови додатка буде реалізація маршрутів. Код представлено у ДОДАТКУ А. Побудувавши ієрархію маршрутів переходимо до створення контролерів. Розглянемо детально основні методи, які виконують логіку додатка.

Базові розрахунки для коректної роботи додатка проводяться у моделі User. При реєстрації користувач вводить данні про себе та заміри свого тіла, які необхідні для розрахунків.

Модель User використовується для роботи з даними користувача, включаючи розрахунки індивідуальних параметрів, таких як базальний метаболізм (BMR), відсоток жирової тканини, норми нутрієнтів і вітамінів. Модель інтегрує науково обґрунтовані формули для забезпечення точності та персоналізації.

Перший метод – розрахунок відсотка жирової тканини **calculateBodyFatPercentage()**. Розрахунок відсотка жирової тканини на основі антропометричних даних користувача. Метод приймає дані про обхват талії, шії, стегон (для жінок) і зріст. Залежно від статі користувача використовуються різні формули:

```
public function calculateBodyFatPercentage()
{
    if ($this->gender === 'male') {
        return 86.01 * log10($this->waist - $this->neck) - 70.041 * log10($this->height) + 30.76;
    } else {
        return 163.205 * log10($this->waist + $this->hips - $this->neck) - 97.684 * log10($this->height) - 106.387;
    }
}
```

Повертає відсоток жирової тканини у вигляді десяткового числа.

Персоналізація плану харчування, людям із вищим відсотком жиру рекомендується знижувати споживання жирів, щоб ефективніше втрачати вагу. Людям із нижчим відсотком жиру можуть бути рекомендовані вищі норми білків для підтримки м'язової маси. Відсоток жиру понад норми (20–25% для чоловіків і 30–35% для жінок) може сигналізувати про ризик захворювань, таких як

ожиріння, діабет або серцево-судинні захворювання. Додаток може попередити користувача та рекомендувати звернути увагу на зміну способу життя. Візуалізація прогресу для побудови графіків змін складу тіла, що дозволяє користувачам чітко бачити динаміку змін.

Наступний метод розраховує базальний метаболізм(BMR) **calculateBMR**. Розрахунок базального метаболізму (BMR) для визначення мінімальної кількості калорій, необхідних для підтримання життєдіяльності.

Метод використовує три формули:

```
public function calculateBMR()
{
    $weight = $this->weight;
    $height = $this->height;
    $age = $this->age;
    $gender = $this->gender;
    if ($this->weight <= 0 || $this->height <= 0 || $this->age <= 0) {
        return 0;
    }
    $bmrHarrisBenedict = $gender === 'male'
        ? 88.362 + (13.397 * $weight) + (4.799 * $height) - (5.677 * $age)
        : 447.593 + (9.247 * $weight) + (3.098 * $height) - (4.330 * $age);
    $bmrMifflinStJeor = $gender === 'male'
        ? (10 * $weight) + (6.25 * $height) - (5 * $age) + 5
        : (10 * $weight) + (6.25 * $height) - (5 * $age) - 161;
    $bodyFatPercentage = $this->calculateBodyFatPercentage();
    $leanMass = $weight * (1 - $bodyFatPercentage / 100);
    $bmrKatchMcArdle = 370 + (21.6 * $leanMass);
    $averageBMR = ($bmrHarrisBenedict + $bmrMifflinStJeor + $bmrKatchMcArdle) / 3;
    return $averageBMR;
}
```

Метод повертає середнє значення з трьох формул для підвищення точності.

Побудова раціону, залежно від BMR обчислюються щоденні потреби у калоріях та нутрієнтах. Дозволяє оптимізувати харчування під фізичну активність і цілі (зниження чи набір ваги). Усі розрахунки нутрієнтів (білки, жири, вуглеводи) базуються на TDEE, а TDEE залежить від BMR. Відстеження BMR у часі дозволяє оцінити вплив змін у вазі, складі тіла або рівні активності. Низький BMR може сигналізувати про проблеми з метаболізмом. Використовується для визначення дефіциту чи профіциту калорій, що веде до змін ваги.

Метод **calculateNutritionNorms()** проводить розрахунки добової норми нутрієнтів. Код методу наведено у ДОДАТКУ В. Розрахунок добової потреби в калоріях, білках, жирах та вуглеводах (КБЖВ) залежно від цілей користувача. Добова норма калорій (TDEE) розраховується на основі BMR і рівня активності: Ці значення використовуються для створення плану харчування та відстеження дотримання дієти.

Отримання базального метаболізму (BMR). Метод опирається на значення, розраховане функцією **calculateBMR()**. BMR є основою для розрахунку добової потреби в калоріях, оскільки враховує мінімальні енергетичні потреби організму.

Розрахунок загальної добової потреби в калоріях (TDEE). Формула:

$TDEE = BMR \cdot AF$, де AF — коефіцієнт фізичної активності, який може мати такі значення:

- 1.2: мінімальна активність (сидячий спосіб життя).
- 1.375: легка активність (легкі фізичні вправи 1-3 рази на тиждень).
- 1.55: середня активність (вправи 3-5 разів на тиждень).
- 1.725: висока активність (інтенсивні вправи 6-7 разів на тиждень).
- 1.9: дуже висока активність (фізична робота або дворазові тренування).

Коефіцієнти активності базуються на дослідженнях метаболічних витрат для різних рівнів фізичної активності. Вони дозволяють адаптувати розрахунок до реальних потреб користувача[17].

Адаптація до цілі користувача. Добова калорійність коригується залежно від цілі:

- **Зниження ваги:** дефіцит калорій у розмірі 22% від TDEE. Калорії для зниження ваги = $TDEE * 0.78$.
- **Набір ваги:** профіцит калорій у розмірі 22% від TDEE. Калорії для набору ваги = $TDEE * 1.22$.
- **Підтримка ваги:** використовується значення TDEE без змін.

Зміна калорійності на 20-25% є безпечним і ефективним підходом для корекції ваги, рекомендованим науковими організаціями, такими як ВООЗ[18].

Розрахунок макронутрієнтів (КБЖВ). Кількість білків, жирів і вуглеводів розраховується наступним чином:

- Білки: залежно від цілі та складу тіла: Білки (г) = вага (кг)·коефіцієнт білків. Для людей із нижчим % жиру коефіцієнт білків вищий (до 2.1 г/кг).
- Жири: залежно від ваги та цілі: Жири (г)=вага (кг)·коефіцієнт жирів. Для зниження ваги кількість жирів зменшується.
- Вуглеводи: залишок калорій після розрахунку білків і жирів: Калорії для вуглеводів = Калорії – (Білки (г)·4)–(Жири (г)·9).

Білки необхідні для підтримки м'язової маси, особливо при зниженні ваги. Жири підтримують гормональну функцію, але їх обмежують для створення дефіциту калорій. Вуглеводи є джерелом енергії, тому їх кількість адаптується до потреб[19].

Для жінок мінімальне значення калорійності встановлено на рівні 1200 ккал для запобігання негативним наслідкам для здоров'я. Значення базується на рекомендаціях ВООЗ і медичних дослідженнях[21]. Формули розрахунку макронутрієнтів базуються на рекомендаціях ВООЗ, американських дієтологічних асоціацій (ADA) та інших авторитетних джерел[20]. Алгоритм враховує вагу, рівень фізичної активності, склад тіла та цілі користувача, що забезпечує індивідуальний підхід. Додаток запобігає надмірному дефіциту калорій, що може викликати втрату м'язової маси, гормональні порушення чи інші ускладнення.

calculateVitaminMineralNorms() – метод з обрахунку необхідних вітамінів та мінералів. Розрахунок норм вітамінів і мінералів залежно від статі та цілей користувача. Використовуються рекомендовані значення RDA (Recommended Dietary Allowance) для основних мікронутрієнтів:

```
public function calculateVitaminMineralNorms()
{
    $norms = [
        'calcium' => 1000, // мг
        'iron' => $this->gender === self::GENDER_FEMALE ? 18 : 8, // мг
        'magnesium' => $this->gender === self::GENDER_FEMALE ? 310 : 400, // мг
        'phosphorus' => 700, // мг
        'potassium' => 3500, // мг
    ]
}
```

```

'sodium' => 2300, // мг
'zinc' => $this->gender === self::GENDER_FEMALE ? 8 : 11, // мг
'copper' => 0.9, // мг
'selenium' => 55, // мкг
'vitamin_c' => 75, // мг
'vitamin_d' => 600, // МЕ
'vitamin_e' => 15, // мг
'vitamin_b6' => 1.3, // мг
'folic_acid' => 400, // мкг
];

switch ($this->goal) {
    case self::GOAL_WEIGHT_LOSS:
        $norms['magnesium'] += 10;
        $norms['vitamin_c'] += 15;
        break;
    case self::GOAL_WEIGHT_GAIN:
        $norms['calcium'] += 200;
        $norms['phosphorus'] += 100;
        break;
}

return $norms;
}

```

Для користувачів доступна генерація харчування за допомогою ШІ.

DiaryOpenAiController - контролер забезпечує автоматизовану генерацію плану харчування для користувачів на основі їхніх персональних норм калорій, білків, жирів і вуглеводів (КБЖВ) та списку продуктів, обраних користувачем.

Метод - **generatePlan()**. Код метода наведено у ДОДАТКУ Г. Генерація індивідуального плану харчування за допомогою OpenAI та інтеграція результатів у базу даних. Відправка списку бажаних продуктів до OpenAi, норм КБЖВ, отримання згенерованого раціону

Генерація плану харчування з урахуванням індивідуальних норм дозволяє користувачам дотримуватися персоналізованого підходу до раціону. OpenAI додає гнучкість і креативність у складання планів, враховуючи доступні продукти. Результатом роботи метода є збережений план харчування, включаючи розподіл страв по годинах (наприклад, сніданок, обід, вечеря).

Розглянемо контролер для розрахунку та відображення тренувального процесу – ExerciseProgressController.

Контролер забезпечує обробку та візуалізацію даних прогресу користувачів у виконанні вправ, включаючи: максимальну вагу, загальний обсяг тренування (вага * повторення), загальну кількість повторень.

Детально розглянемо кожен з методів контролера:

showProgress() – метод для відображення прогресу по конкретній вправі.

```
public function showProgress($exerciseId, Request $request)
{
    $userId = auth()->id(); $range = $request->input('range', 'all'); $exercise =
    Exercise::findOrFail($exerciseId); $progress = WorkoutSet::where('exercise_id', $exerciseId)
    ->whereHas('diary', function ($query) use ($userId) { $query->where('user_id', $userId);
    });
    if ($range !== 'all') {
        $days = (int)$range * 7;
        $progress = $progress->where('workout_date', '>=', now()->subDays($days));
    }
    $progress = $progress->get(); $exercises = Exercise::whereHas('workoutSets', function ($query)
    use ($userId) { $query->where('user_id', $userId);
    })->get();
    $chartData = [
        'labels' => $progress->pluck('workout_date')->map(function ($date) {
            return \Carbon\Carbon::parse($date)->toIso8601String();
        })->toArray(), 'max_weights' => $progress->pluck('weight')->toArray(),
        'volumes' => $progress->map(function ($set) {
            return $set->weight * $set->reps;
        })->toArray(),
        'reps' => $progress->pluck('reps')->toArray(),
    ]; $dailyProgress = $progress->groupBy(function ($item) {
        return $item->workout_date;
    })->map(function ($items) {
        return [
            'max_weight' => $items->max('weight'),
            'total_volume' => $items->sum(function ($item) {
                return $item->weight * $item->reps;
            }), // Суммарный объем
            'total_reps' => $items->sum('reps'),
            'max_1rm' => $items->map(function ($item) {
                return $item->weight * (1 + $item->reps / 30);
            })->max(),
        ];
    });
    return view('workout-diary.progress', compact('exercise', 'progress', 'chartData', 'exercises',
    'range', 'dailyProgress'));
}
```

Метод відображає прогрес користувача у конкретній вправі. Отримання історії виконання вибраної вправи, обчислення ключових параметрів,

формування даних для візуалізації прогресу. Дозволяє користувачам бачити динаміку їхніх досягнень і виявляти закономірності у тренуваннях.

calculateProgress() – метод для розрахунку прогресу у конкретній вправі.

```
public function calculateProgress($userId, $exerciseId)
{
    $sets = WorkoutSet::where('user_id', $userId)
        ->where('exercise_id', $exerciseId)
        ->get();
    if ($sets->isEmpty()) {
        return response()->json([
            'message' => 'Не достатньо даних для аналізу прогреса.',
        ], 404);
    }
    $maxWeight = $sets->max('weight');
    $totalVolume = $sets->sum(function ($set) {
        return $set->weight * $set->reps;
    });
    $totalReps = $sets->sum('reps');
    $progress = ExerciseProgress::updateOrCreate(
        [
            'user_id' => $userId,
            'exercise_id' => $exerciseId,
            'date' => now()->toDateString(),
        ],
        [
            'max_weight' => $maxWeight,
            'total_volume' => $totalVolume,
            'total_reps' => $totalReps,
        ]
    );
    return response()->json([
        'message' => 'Успішно.',
        'progress' => $progress,
    ]);
}
```

Метод обчислює поточний прогрес у конкретній вправі. Завантаження всіх підходів у вибраній вправі, визначення максимальної ваги та обчислення загального обсягу. Використовується для швидкого доступу до ключових показників прогресу.

getProgressChart() – метод для формування графічного представлення.

```
public function getProgressChart($userId, $exerciseId)
{
    $progress = ExerciseProgress::where('user_id', $userId)
        ->where('exercise_id', $exerciseId)
        ->orderBy('date')
        ->get(['date', 'max_weight', 'total_volume', 'total_reps']);
}
```

```

if ($progress->isEmpty()) {
    return response()->json([
        'message' => 'Нема даних.',
    ], 404);
}
$chartData = [
    'labels' => $progress->pluck('date'),
    'datasets' => [
        [
            'label' => 'Максимальна вага',
            'data' => $progress->pluck('max_weight'),
            'borderColor' => 'rgba(75, 192, 192, 1)',
            'borderWidth' => 2,
        ],
        [
            'label' => 'Загальний об'єм',
            'data' => $progress->pluck('total_volume'),
            'borderColor' => 'rgba(192, 75, 75, 1)',
            'borderWidth' => 2,
        ],
        [
            'label' => 'Загальна кількість повторень',
            'data' => $progress->pluck('total_reps'),
            'borderColor' => 'rgba(75, 75, 192, 1)',
            'borderWidth' => 2,
        ],
    ],
];
return response()->json([
    'message' => 'Данні для графіка отримані.',
    'chartData' => $chartData,
]);
}

```

Метод для формування графічних даних для візуалізації прогресу. Забезпечує наочне уявлення про прогрес користувача.

Розглянемо базовий контролер щоденника харчування — **FoodDiaryIndexController()**. Контролер забезпечує обробку даних щоденника, розраховуючи спожиті нутрієнти та вітаміни, аналізує ефективність харчування, прогнозує зміну ваги користувача. Розглянемо детально кожен з розрахункових методів контролера.

Метод **calculateEfficiency()** — для оцінки ефективності раціону.

```

public function calculateEfficiency($userId, $date)
{
    $user = Auth::user();
    $totals = $this->calculateDailyKBJU($userId, $date);
    $norms = $user->calculateNutritionNorms();
}

```

```

if (!$totals) {
    return 0;
}
$efficiency = 100;
$categoryWeight = 25
$goal = $user->goal;
foreach (['calories', 'proteins', 'fats', 'carbs'] as $key) {
    $actual = $totals[$key] ?? 0;
    $norm = $norms[$key] ?? 1;
    $deviation = abs($actual - $norm) / $norm * 100;
    if ($key === 'calories') {
        if ($goal === 'weight_loss') {
            if ($actual > $norm) {
                $efficiency -= $categoryWeight;
            }
        } elseif ($goal === 'weight_gain') {
            if ($actual < $norm) {
                $efficiency -= $categoryWeight;
            }
        }
    }
}
if ($deviation > 15) {
    if ($deviation <= 30) {
        $efficiency -= $categoryWeight * 0.2;
    } elseif ($deviation <= 50) {
        $efficiency -= $categoryWeight * 0.5;
    } else {
        $efficiency -= $categoryWeight;
    }
}
}

return max(0, $efficiency);
}

```

Метод допомагає оцінити ефективність харчування користувача на основі відхилень від добових норм калорій та макронутрієнтів (КБЖВ). Визначається відсоткове відхилення від норм для кожного параметра КБЖВ

- Якщо відхилення $\leq 15\%$, ефективність — 100%.
- Якщо відхилення від 15% до 30% — ефективність зменшується на 20%.
- Якщо відхилення від 30% до 50% — ефективність зменшується на 50%.
- Якщо відхилення понад 50% — ефективність мінімальна (20%).

Ефективність обчислюється окремо для калорій, білків, жирів і вуглеводів, а потім береться середнє значення.

Відхилення $\leq 15\%$ вважається допустимим, оскільки не впливає суттєво на прогрес. Значні відхилення (понад 50%) вказують на серйозні порушення

раціону, що знижує ефективність. Ефективність харчування є критичним параметром для оцінки, наскільки раціон відповідає метаболічним потребам користувача. Відхилення у межах 15% є допустимими, оскільки не викликають суттєвих порушень у метаболічних процесах. Це базується на дослідженнях про толерантність організму до короткострокових змін у раціоні. Відхилення понад 30% сигналізують про те, що раціон не збалансований, що може призводити до втрати м'язової маси, уповільнення метаболізму або накопичення жирових запасів. Метод забезпечує персоналізований підхід до моніторингу, оскільки враховує як калорійність, так і розподіл макронутрієнтів. Після обчислення користувач отримає ефективність у відсотках (від 20% до 100%).

Метод **calculateScenario()** – виконує розрахунок зміни ваги для трьох сценаріїв на основі дефіциту чи профіциту калорій. Код метода наведено у ДОДАТКУ Д.

Метод рохравує три сценарії змін ваги користувача (оптимістичний, песимістичний, поточний) на основі дефіциту/профіциту калорій. Обчислення добового дефіциту чи профіциту калорій, зміну ваги та побудування графіків сценаріїв. Обчислення добового дефіциту/профіциту калорій:

Сценарії:

- **Оптимістичний:** передбачає ідеальне дотримання норм та зменшення ваги на основі планового дефіциту.
- **Песимістичний:** враховує можливе переїдання та, відповідно, зростання ваги.
- **Поточний:** базується на фактичному дефіциті/профіциті калорій.

Розрахунок ваги на кілька днів: Вага на кожен день прогнозується шляхом додавання зміни ваги до початкового значення.

Прогноз зміни ваги на основі дефіциту або профіциту калорій є науково обґрунтованим методом, широко визнаним у спортивній дієтології. Значення 7200 ккал для втрати 1 кг жирової тканини базується на середніх показниках енергетичної цінності жиру (9 ккал/г). Ураховуються також витрати на

метаболізм самого процесу ліполізу. Цей підхід мотивує користувача до підтримки раціону, показуючи як позитивний, так і негативний потенціал змін.

Метод **calculateDailyChange()** – виконує розрахунок щоденної зміни ваги.

```
public function calculateDailyChange($userId, $date)
{
    $user = Auth::user();
    $totals = $this->calculateDailyKBJU($userId, $date);
    $norms = $user->calculateNutritionNorms();
    $goal = $user->goal;
    $dailyChange = 0;
    if ($goal === 'weight_loss') {
        $dailyChange = round(($totals['calories'] - $norms['calories']) / 7200, 3);
    } elseif ($goal === 'weight_gain') {
        $dailyChange = round(($totals['calories'] - $norms['calories']) / 7200, 3);
        if ($totals['calories'] < $norms['calories']) {
            $dailyChange = round(($totals['calories'] - $norms['calories']) / 7200, 3);
        }
    }
    return $dailyChange;
}
```

Метод розраховує щоденну зміну ваги користувача на основі спожитих калорій порівняно з його добовою нормою калорійності (TDEE). Обчислення добової зміни ваги відносно спожитого КБЖВ. Для розрахунку дані про КБЖВ отримуються з метода calculateDailyKBJU(), добові норми калорій розраховуються в методі calculateNutritionNorms().

Формула базується на принципах енергетичного балансу, де 7200 ккал еквівалентні втраті або набору 1 кг жирової маси. Метод враховує персональні норми калорійності та цілі користувача, що робить розрахунки більш точними та персоналізованими. Метод дозволяє користувачеві оцінити вплив конкретного дня на зміну ваги, що допомагає аналізувати раціон і коригувати поведінку.

Висновки до розділу 2

У другому розділі розглянуто проектування системи інформаційного додатка, що об'єднує індивідуальний підхід до харчування, тренувань і моніторингу здоров'я. У процесі проектування було створено концептуальну основу системи, яка поєднує високий рівень функціональності з науковим підходом до аналізу даних.

Розроблена архітектура бази даних охоплює всі необхідні аспекти роботи додатка, включаючи збереження інформації про користувачів, продукти харчування, тренування та прогрес. Особливий акцент зроблено на оптимізації доступу до даних, що гарантує швидкість та ефективність обробки великих обсягів інформації. Кожен елемент бази даних інтегровано у систему так, щоб забезпечити зручність використання та масштабованість у майбутньому.

У розділі також описані ключові алгоритми, які лежать в основі роботи додатка. Зокрема, це алгоритми розрахунку добової калорійності (BMR), норм макроеlementів (КБЖВ), а також оцінки ефективності раціону. Кожен алгоритм базується на сучасних дослідженнях у сфері дієтології та спортивної медицини, що підвищує точність розрахунків і відповідність рекомендацій реальним потребам користувачів.

Значну увагу приділено створенню функціоналу, який дозволяє користувачам автоматично відстежувати зміни у своєму раціоні та тренувальному прогресі. Використання інтеграції даних із алгоритмами прогнозування дозволяє формувати чіткі сценарії змін ваги, базуючись на дефіциті або профіциті калорій. Це сприяє не лише досягненню спортивних чи дієтичних цілей, але й підтриманню мотивації користувачів.

Розділ також демонструє важливість дотримання сучасних стандартів безпеки даних. Проектування бази даних і функціональних модулів враховує можливі ризики та реалізує захист персональної інформації користувачів, що відповідає рекомендаціям OWASP.

Таким чином, другий розділ закладає основу для створення інформаційного додатка, який забезпечує ефективний аналіз харчування, тренувань і здоров'я користувачів. Інтеграція наукових підходів, сучасних алгоритмів і гнучкого функціоналу робить систему універсальним інструментом для досягнення особистих цілей у сфері здорового способу життя.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКА

3.1 Порядок встановлення та налаштування системи

Для роботи з додатком на локальному сервері необхідно налаштувати певне середовище розробки. Для коректної роботи фреймворку Laravel і пов'язаних з ним інструментів необхідно встановити такі компоненти: npm/yarn, Node.js, PHP (версія 8+), Composer, MySQL, а також інтегроване середовище розробки, наприклад, PhpStorm. Розглянемо процес встановлення цих технологій детальніше:

1. Для встановлення **npm** на Windows необхідно завантажити архів з офіційного сайту [10] та слідувати інструкціям інстальатора.

2. Для встановлення **nodejs** потрібно перейти на сайт [11] для завантаження інстальатора та виконувати дії необхідні для встановлення.

3. Для встановлення **composer** необхідно слідувати інструкціями з встановлення на офіційному сайті [12].

4. Встановлення **PHP** виконується завантаженням архіву з офіційного сайту. Обравши необхідну версію, після завантаження, необхідно додати шлях до файлу php.exe у реєстр Windows [7].

5. Встановлення **mysql** виконується за допомогою вказівок у завантаженому інстальаторі з офіційного сайту [9].

Після встановлення всіх необхідних компонентів потрібно налаштувати середовище для запуску проєкту. У середовищі розробки (в даному випадку PhpStorm) слід відкрити проєкт і створити базу даних. Для цього можна скористатися MySQL Workbench або іншими інструментами для роботи з базами даних. Після створення бази потрібно налаштувати файл конфігурації .env, вказавши параметри підключення до бази даних, такі як ім'я хоста, назву бази, ім'я користувача та пароль.

```
DB_DATABASE=example
```

```
DB_USERNAME=root
```

```
DB_PASSWORD=root
```

Після змін необхідно проініціалізувати проект написавши команду **npm install**. Після успішного встановлення всіх залежностей, необхідно виконати міграції, відбувається це командою **php artisan migrate**.

Виконавши ці дії проект готовий до роботи та змін. Перевірити стан та запустити сервер можна за допомогою команди **php artisan serve**.

3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з додатком

Робота додатка буде проводитись з незареєстрованого акаунта.

Головна сторінка знаходиться за адресою <http://127.0.0.1:8000/home>. Користувач може бачити в шапці перелік основних посилань. Користувач має можливість переглянути зміст головної сторінки, перейти за посиланнями у шапці або увійти до облікового запису.

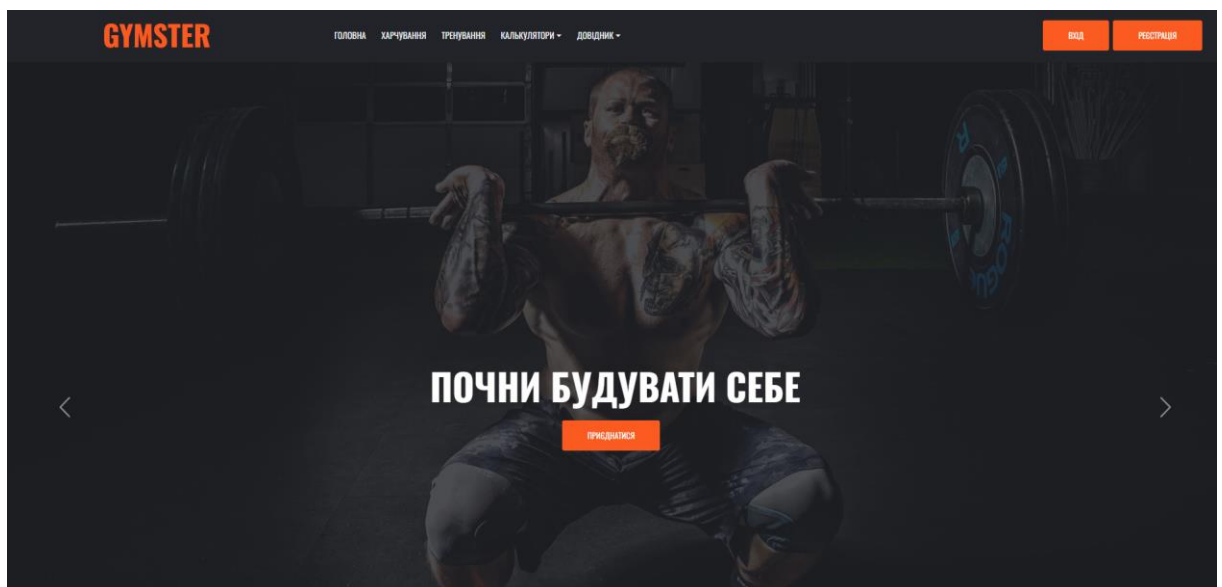


Рисунок 3.1 - Головна сторінка

Натискаючи на реєстрацію користувач повинен заповнити форму з інформацією про себе.

Реєстрація

admin	
admin2@gmail.com	
.....	
.....	
111	
178	
23	
Чоловік	
Зниження ваги	
1.375 - Легка активність	
115	115
115	60
20	40
ЗАРЕЄСТРУВАТИСЯ	

Рисунок 3.2 – Форма реєстрації

Після реєстрації користувачу будуть доступні всі функції додатка.
Розглянемо щоденник харчування.

Виберіть дату:

Дата:

03.12.2024

ПЕРЕЙТИ

ВИБРАТИ ПРОДУКТИ ДЛЯ ГЕНЕРАЦІЇ

Згенерувати план харчування:

ЗГЕНЕРУВАТИ

Щоденник харчування за 2024-12-03

Breakfast (час додавання:)

Продукт

Кількість (г)

Калорії

Білки

Жири

Вуглеводи

ДБ

Lunch (час додавання:)

Продукт

Кількість (г)

Калорії

Білки

Жири

Вуглеводи

ДБ

Dinner (час додавання:)

Продукт

Кількість (г)

Калорії

Білки

Жири

Вуглеводи

ДБ

Додати продукт

Прийом їжі

Сніданок

Продукт

Молоко коров'яче 3.2%

Кількість (г)

ДОДАТИ

Ваші показники КБЖУ:

Назва	Норма	Спожито	Покрито (%)
Calories	2,271.59 ккал	0.00 ккал	0%
Proteins	216.45 г	0.00 г	0%
Fats	77.70 г	0.00 г	0%
Carbs	176.62 г	0.00 г	0%

Вітаміни і мінерали

Назва	Норма	Спожито	Покрито (%)
Кальцій	1000 мг	0 мг	0%
Залізо	8 мг	0 мг	0%
Магній	400 мг	0 мг	0%
Фосфор	700 мг	0 мг	0%
Калій	3500 мг	0 мг	0%
Натрій	2300 мг	0 мг	0%
Цинк	11 мг	0 мг	0%
Мідь	0.9 мг	0 мг	0%
Селен	55 мкг	0 мкг	0%
Вітамін С	75 мг	0 мг	0%
Вітамін D	600 мг	0 мг	0%
Вітамін Е	15 мг	0 мг	0%
Вітамін В6	1.3 мг	0 мг	0%
Фолієва кислота	400 мг	0 мг	0%

Результативність харчування

Поточна результативність: 0 %

Зміна ваги

Імовірний результат: -0.315 кг

Переглянути детально

Виберіть початкову дату:

2024-12-03

Виберіть період прогнозу (тиждень):

1 тиждень

ПОРАХУВАТИ

Рисунок 3.3 – Вигляд щоденника харчування

Користувач може вибрати будь який день та заповнити його даними і в майбутньому повернутись до них. Праворуч користувач може бачити свої норми КБЖВ та вітамінів з мінералами. Користувач може переглянути ймовірний результат свого харчування.

Додаємо декілька продуктів.

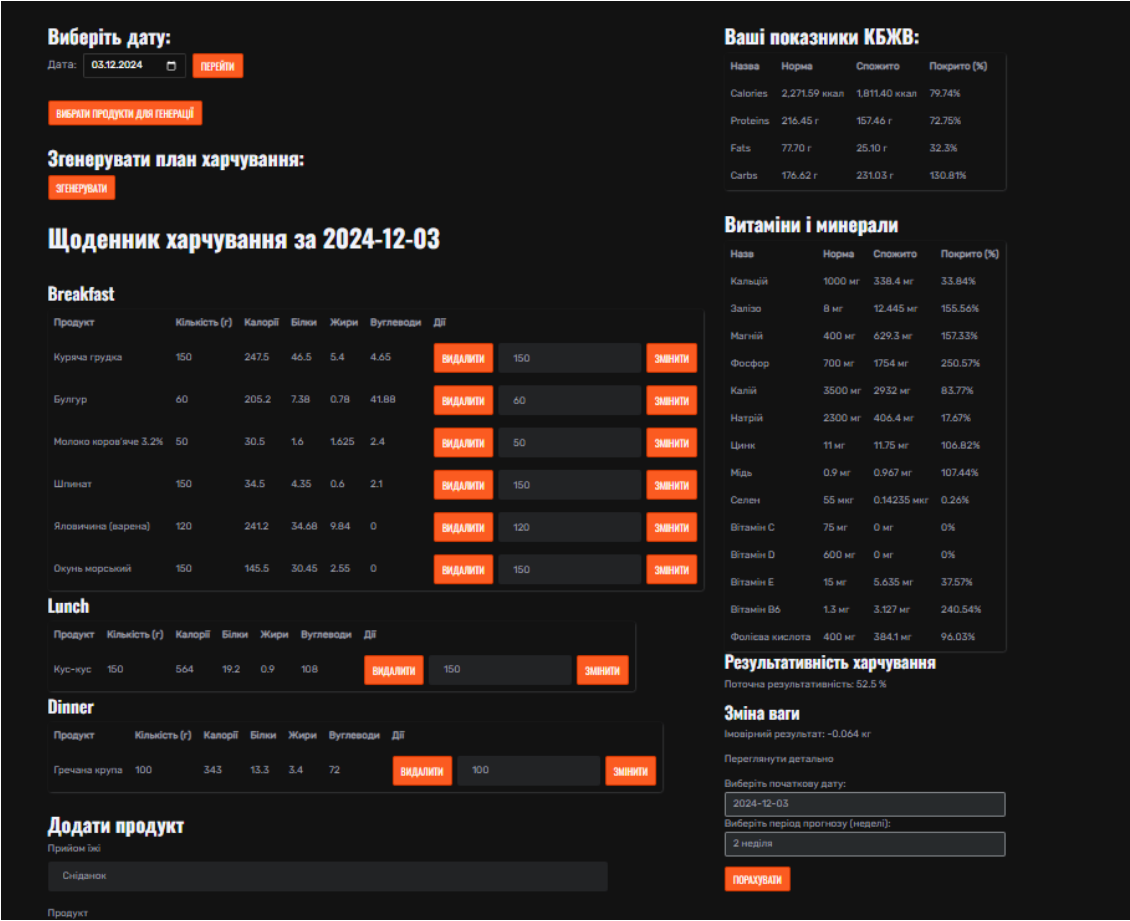


Рисунок 3.4 - Сторінка щоденника харчування

Користувач отримує список своїх прийомів їжі та їх харчову цінність. Користувач бачить скільки КБЖВ спожито та скільки необхідно для закриття добової норми. Переглянемо ймовірні результати на 1 тиждень.

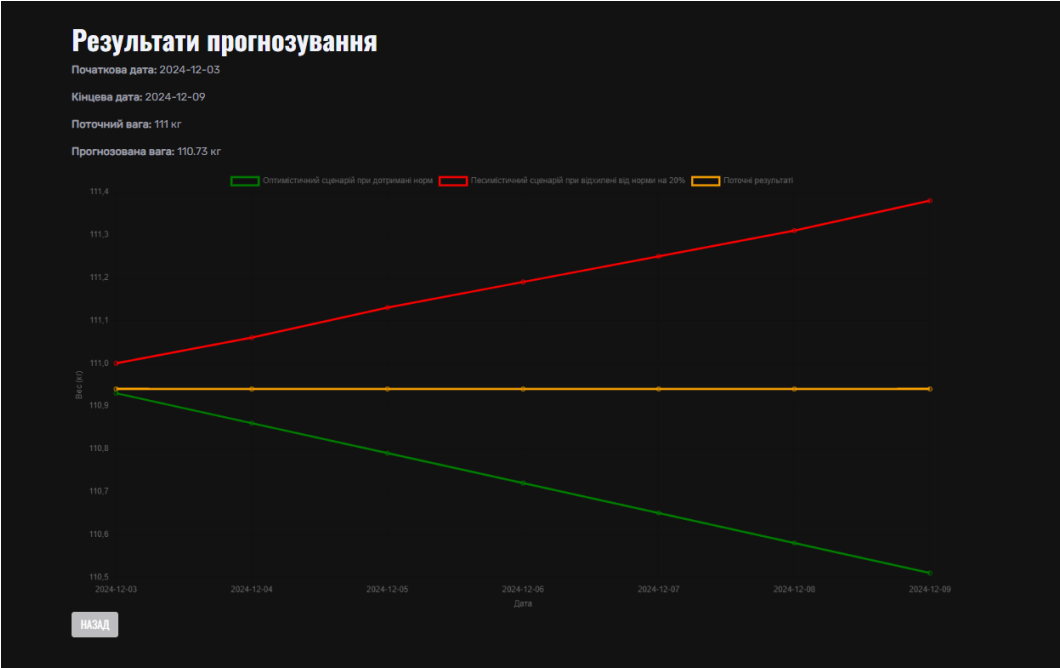
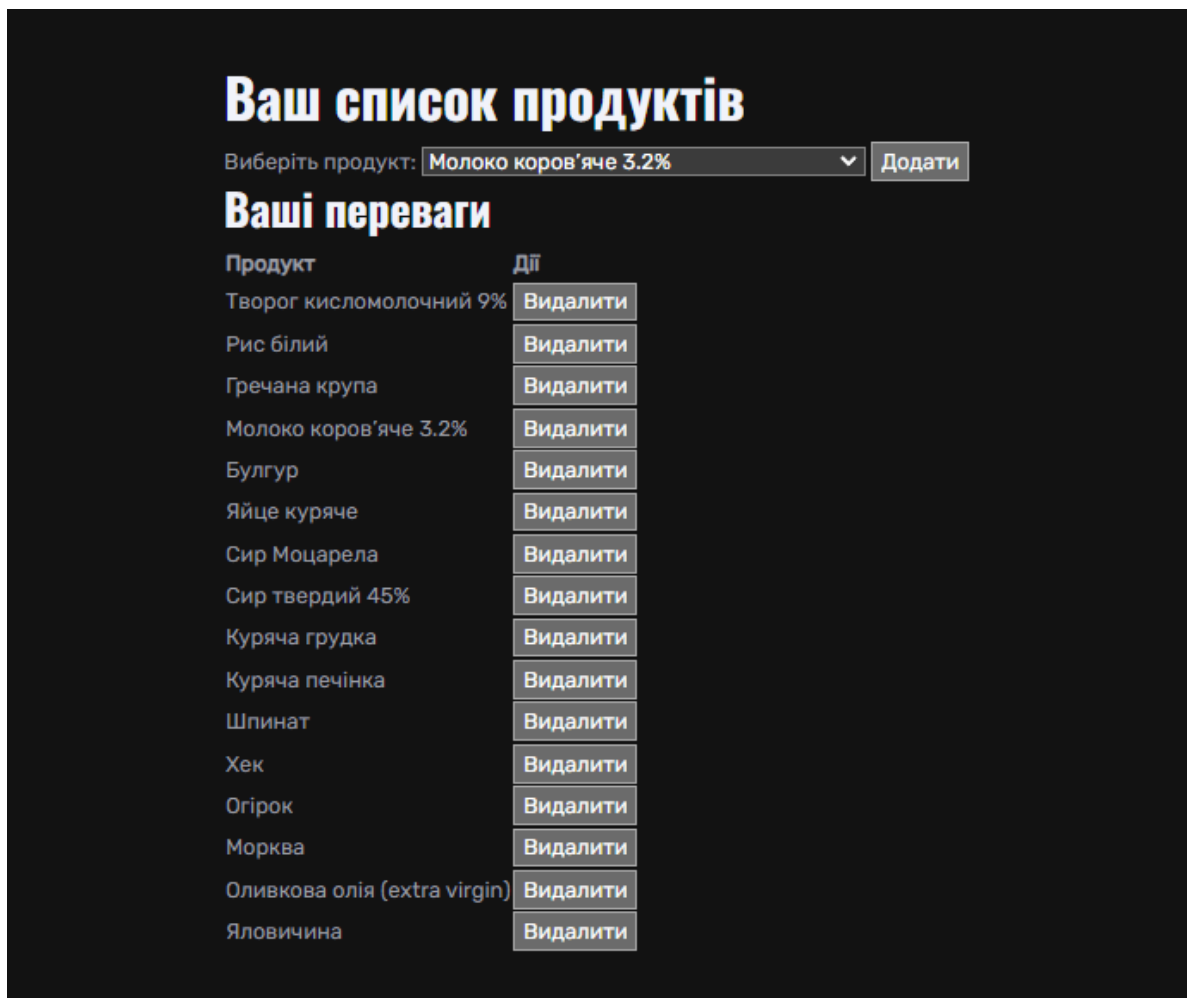


Рисунок 3.5 - Сторінка ймовірних результатів

Користувачу доступний графік на якому зображено 3 лінії – сценарії. Результати можуть бути не точними при малій кількості заповнених днів.

Повернемося до головної сторінки щоденника харчування. Користувач має можливість згенерувати харчування за допомогою ШІ. Проте для того, щоб згенерувати необхідно вибрати список продуктів для генерації.



Продукт	Дії
Творог кисломолочний 9%	Видалити
Рис білий	Видалити
Гречана крупа	Видалити
Молоко коров'яче 3.2%	Видалити
Булгур	Видалити
Яйце куряче	Видалити
Сир Моцарела	Видалити
Сир твердий 45%	Видалити
Куряча грудка	Видалити
Куряча печінка	Видалити
Шпинат	Видалити
Хек	Видалити
Огірок	Видалити
Морква	Видалити
Оливкова олія (extra virgin)	Видалити
Яловичина	Видалити

Рисунок 3.6 – Сторінка бажаних продуктів

Користувач може вибрати продукти з БД в свій список. Розглянемо генерацію харчування на 1 день. Перейдемо в інший день і згенеруємо харчування використовуючи продукти зі списку обраних.

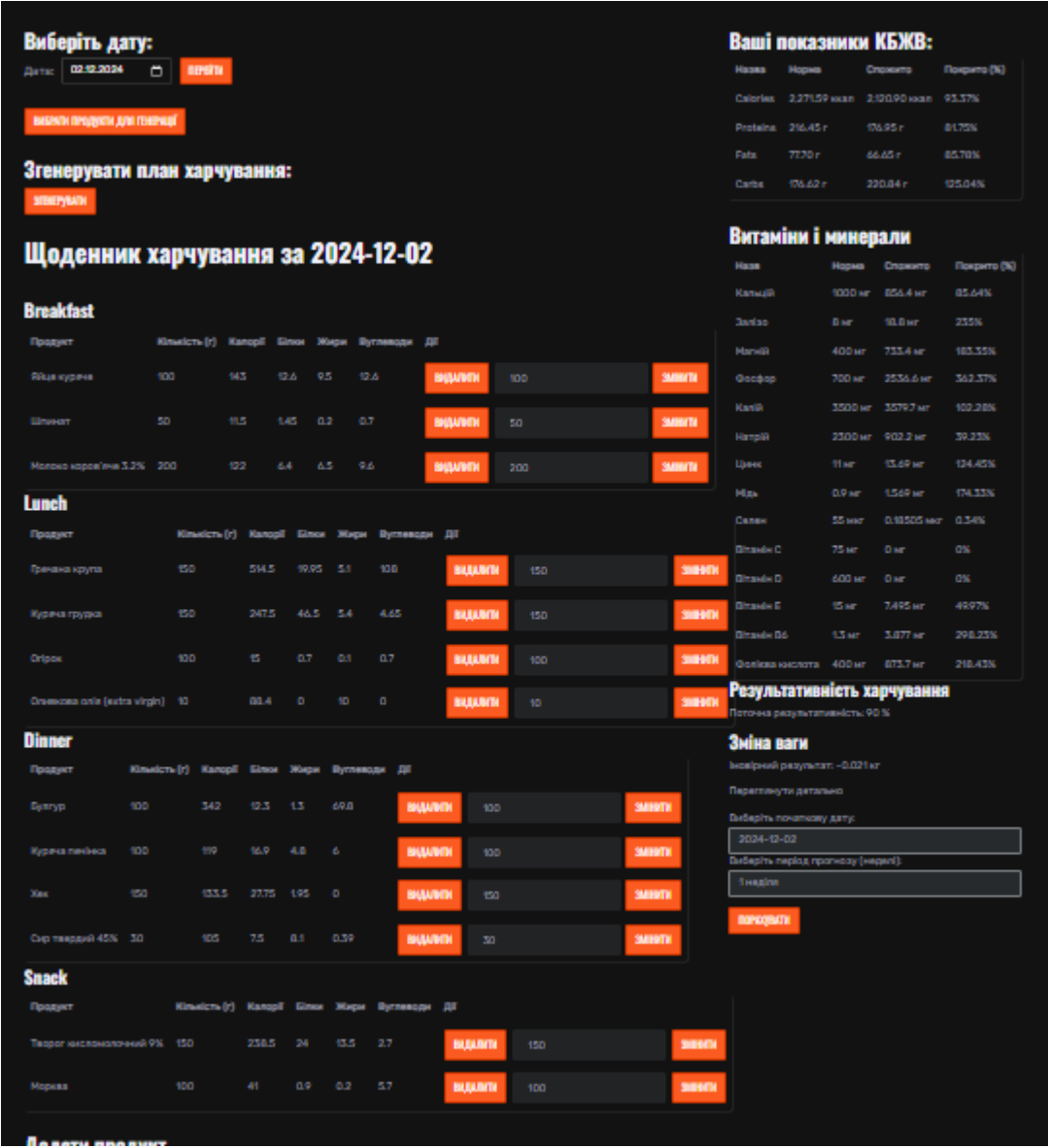


Рисунок 3.7 – Раціон згенерований за допомогою ШІ.

Перейдемо до розгляду роботи із щоденником тренувань.

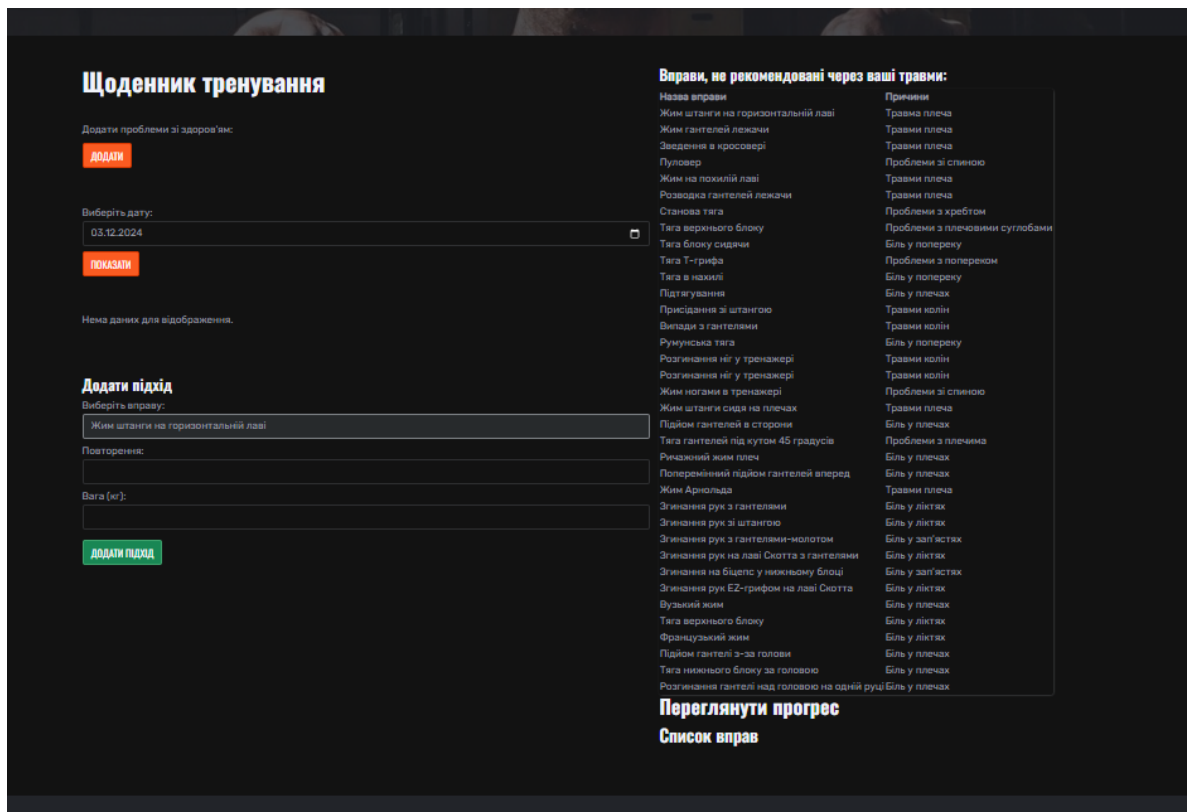


Рисунок 3.8 – Головна сторінка щоденника тренувань

Наразі відображаються всі вправи, як не рекомендовані. виправимо це додавши проблему зі здоров'ям на відповідній сторінці.

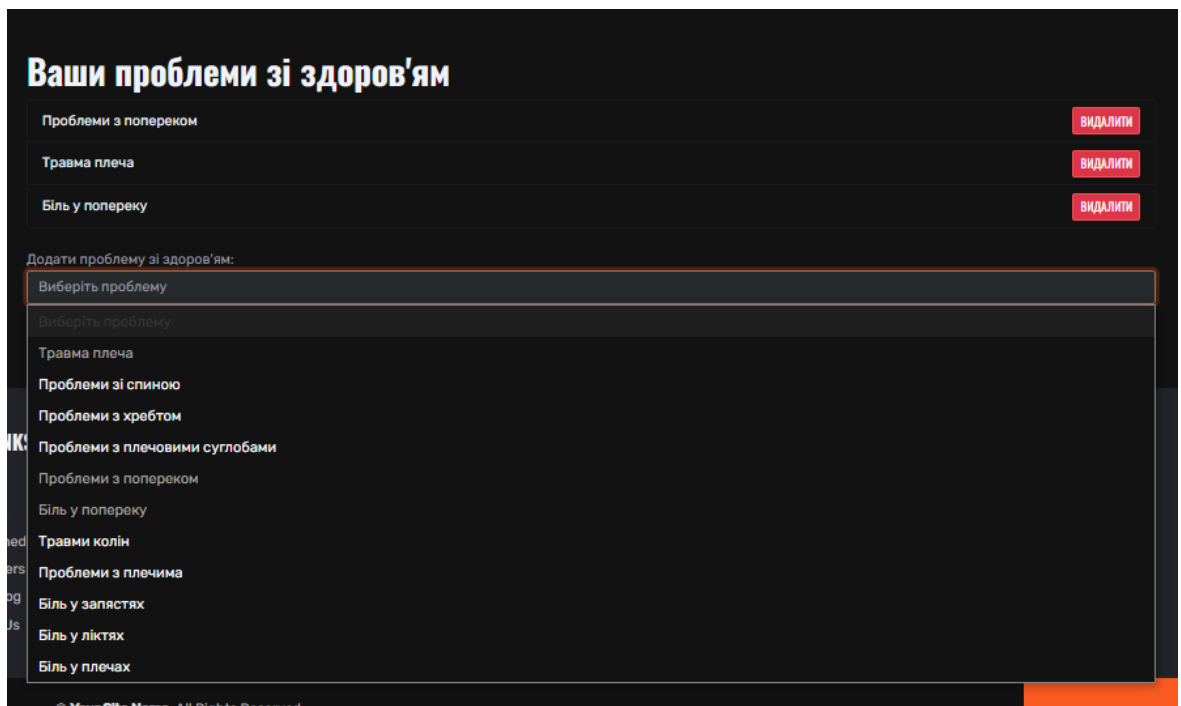


Рисунок 3.9 – Сторінка вибору проблем зі здоров'ям

Користувач має можливість вибрати зі списку те, що його турбує, за потреби видалити. Сторінка щоденника відображає не рекомендовані за станом здоров'я вправи, тому почнемо додавати наше тренування.

Щоденник тренування

Додати проблеми зі здоров'ям:

ДОДАТИ

Виберіть дату:

03.12.2024

ПОКАЗАТИ

Вправи, не рекомендовані через ваші травми:

Назва вправи	Причини
Жим штанги на горизонтальній лаві	Травма плеча
Тяга блоку сидючи	Біль у попереку
Тяга Т-грифа	Проблеми з попереком
Тяга в нахилі	Біль у попереку
Румунська тяга	Біль у попереку

Переглянути прогрес

Список вправ

Жим штанги на горизонтальній лаві [ПЕРЕГЛЯНУТИ ПРОГРЕС](#)

Жим гантелей лежачи [ПЕРЕГЛЯНУТИ ПРОГРЕС](#)

Жим штанги на горизонтальній лаві

Підхід	1	2	3	4	5	6	7	8	9	Підсумок
Повторення, к-сть	12	12	10	10	10	6	12	20	10	102
Вага снаряда, кг	50	80	100	100	100	100	80	50	30	690

Жим гантелей лежачи

Підхід	1	2	3	4	5	Підсумок
Повторення, к-сть	10	10	10	6	10	46
Вага снаряда, кг	20	25	30	30	15	120

Додати підхід

Виберіть вправу:

Жим штанги на горизонтальній лаві

Повторення:

Вага (кг):

ДОДАТИ ПІДХІД

Рисунок 3.10 – Запис тренування

Праворуч користувач може бачити список не рекомендованих для нього вправ.

Користувач обирає вправу зі списку та вказує дані, які виконав або планує виконати. Можливе редагування повторень та ваги. Користувач може побачити свою статистику по кожній вправі.

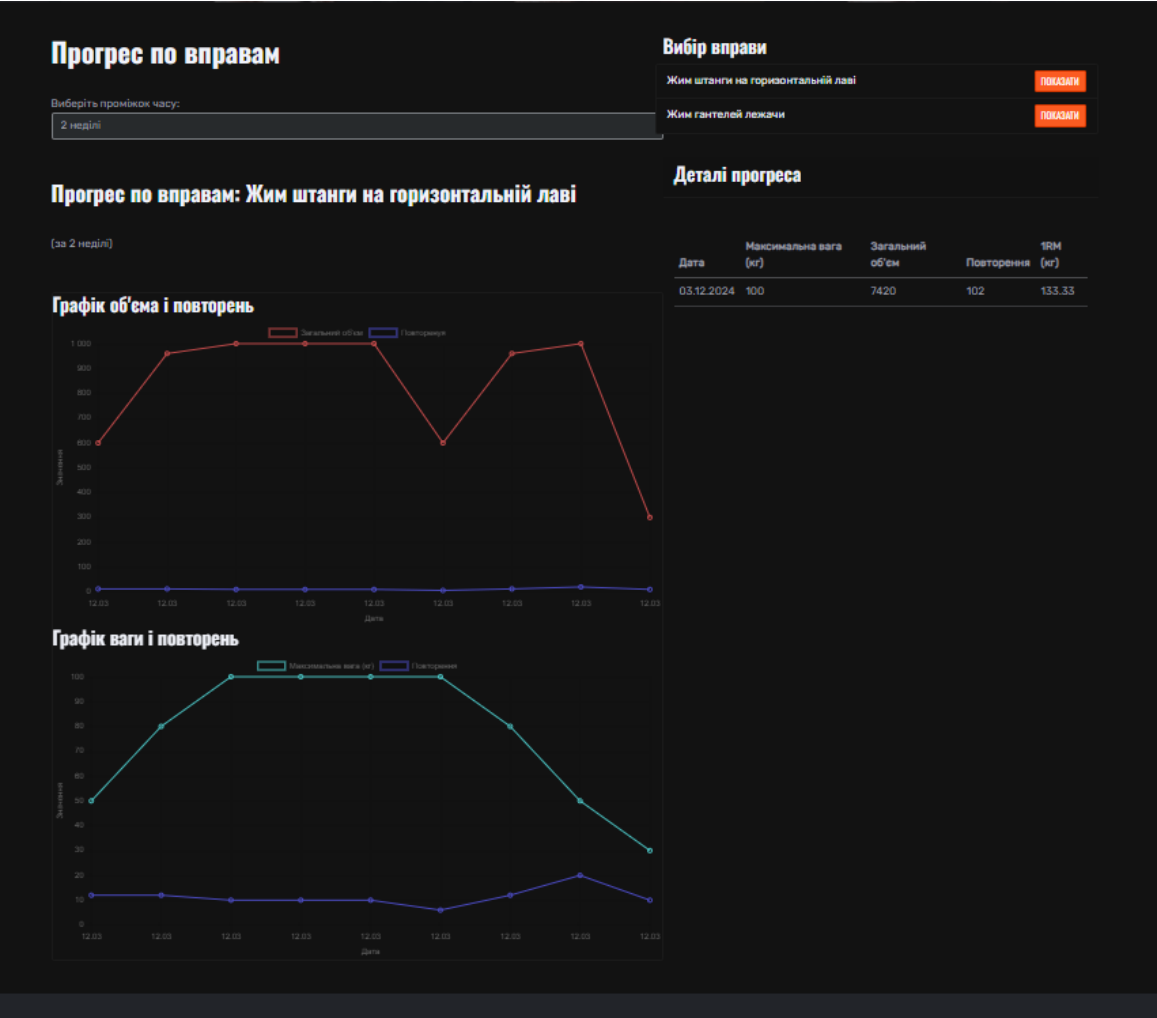


Рисунок 3.11 – Графік результативності вправ

Користувач може переглядати у «Довіднику» продукти та вправи. Розглянемо продукти, які розділені на категорії.

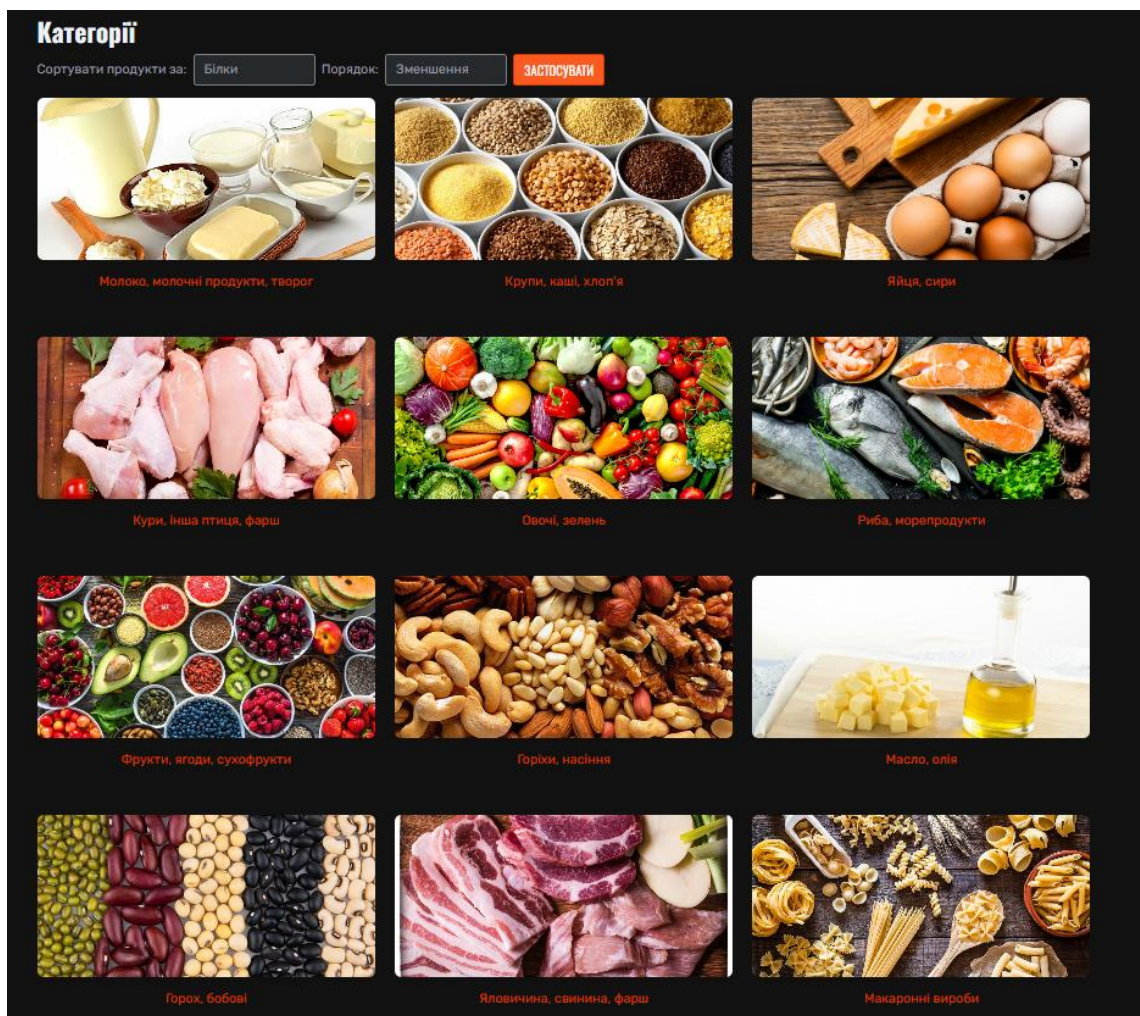


Рисунок 3.12 – Категорії продуктів

Щоб не шукати в кожній категорії потрібні продукти було зроблено сортування за необхідним користувачу компонентом, розглянемо сортування обравши білки. Отримаємо сторінку сортованих продуктів. Натиснувши на продукт користувач може побачити детальний опис його складу.

Список продуктів

Сортування: protein (по зменшенню)

Назва продукта	Калорії	Білки	Жири	Вуглеводи
Сир Пармезан	431	38	29	3.6
Соя (суха)	446	36.5	19.9	30.2
Куряча грудка	Сир Пармезан Калорії: 431 ккал Вода: 30 г Білки: 38 г Жири: 29 г Клітковина: 0 г Сахар: 0.9 г Мінерали: - Кальцій: 1184 мг - Залізо: 0.9 мг - Магній: 44 мг - Фосфор: 694 мг - Калій: 92 мг - Натрій: 1600 мг - Цинк: 3 мг Вітаміни: - Вітамін B6: 0.09 мг - Фолієва кислота: 7 мкг - Вітамін B12: 1.5 мкг - Вітамін E: 0.2 мкг Жири: - Насичені жирні кислоти: 18.9 мг - Мононенасичені жирні кислоти: 7.5 мкг - Поліненасичені жирні кислоти: 1.1 мкг Закрити	3.6	3.1	
Індичка філе		0	1	15.1
Насіння гарбуза		2.8	49.1	10
Яловичина (варена)		3.9	8.2	0
Яловичина (смажена)		7	15	0
Яловичина		5.1	17	0
Арахіс		5.8	49.2	16.1
Горох (сухий)		5.8	1.2	58
Сочевиця (суха)		3.8	1.1	53
Свинина		3.7	20.8	0
Сир твердий 45%		5	27	1.3
Куряче стегно		5	11	0
Свинина (смажена)		5	22.5	0
Яловичина (тушкована)		5	12	0
Сир Чеддер		1.9	33.1	24.9
Свинина (варена)		1.8	20.4	0
Шинка копчена (пармська)		1	15	0
Квасоля (суха)		5.4	0.8	54
Тунець		5.3	4.9	0
Ковбаса копчена (саламі)		5	46	1.2
Гусяче м'ясо	3/1	22.8	33.5	1
Ковбаса мисливська (копчена)	470	22	43	1
Сервелат копчений	510	22	45	1
Мигдаль	579	21.2	49.9	9.5
Насіння соняшника	584	20.8	51.5	13.4
Лосось	206	20.4	13.4	0
Фасоль червона	112	20.4	2.8	0

Рисунок 3.13 – Сторінка продуктів

Розглянемо сторінки з категоріями вправ та вправами.

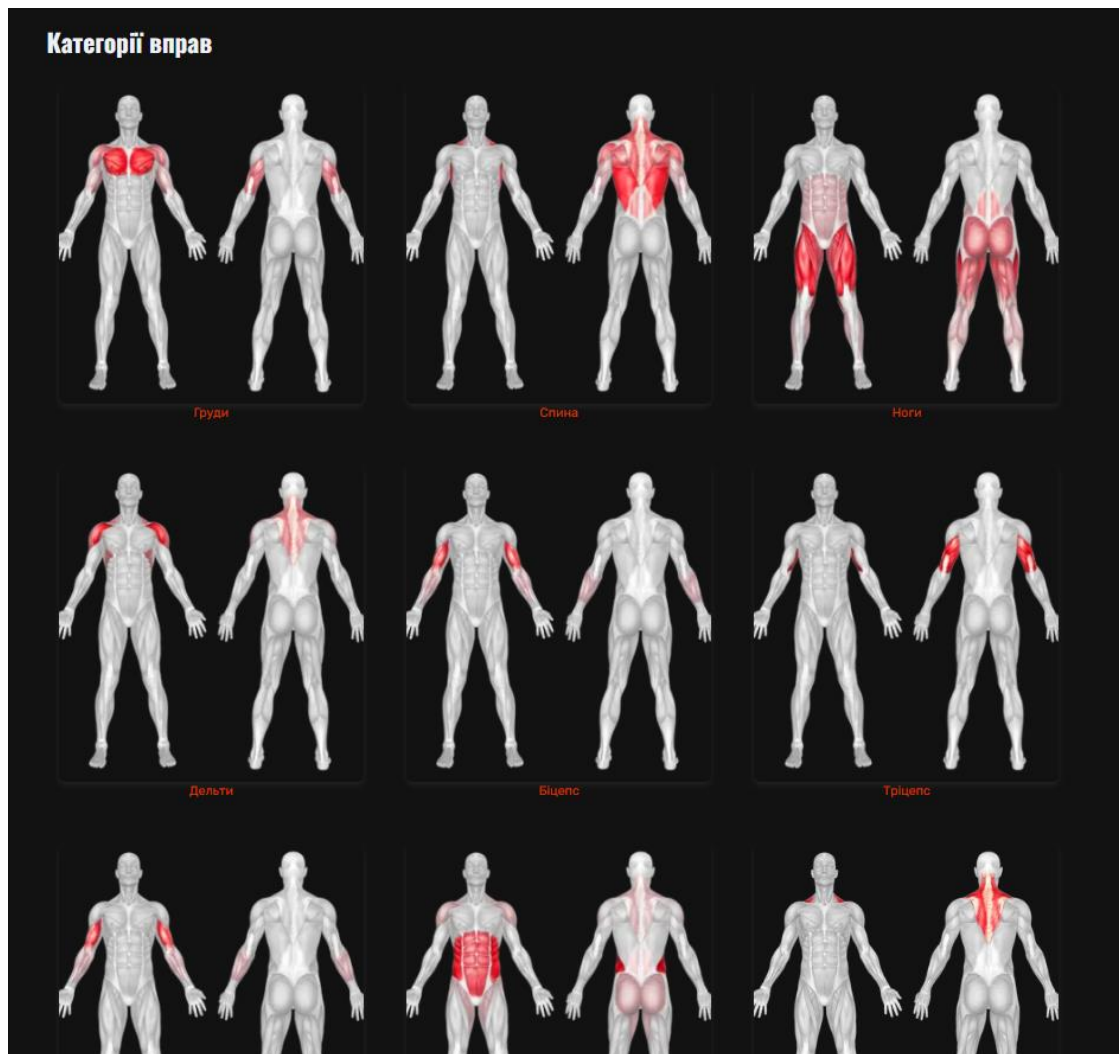


Рисунок 3.14 – Категорії вправ

Користувач обирає необхідно йому групу м'язів та переглядає список вправ. Розглянемо категорію «Спина».

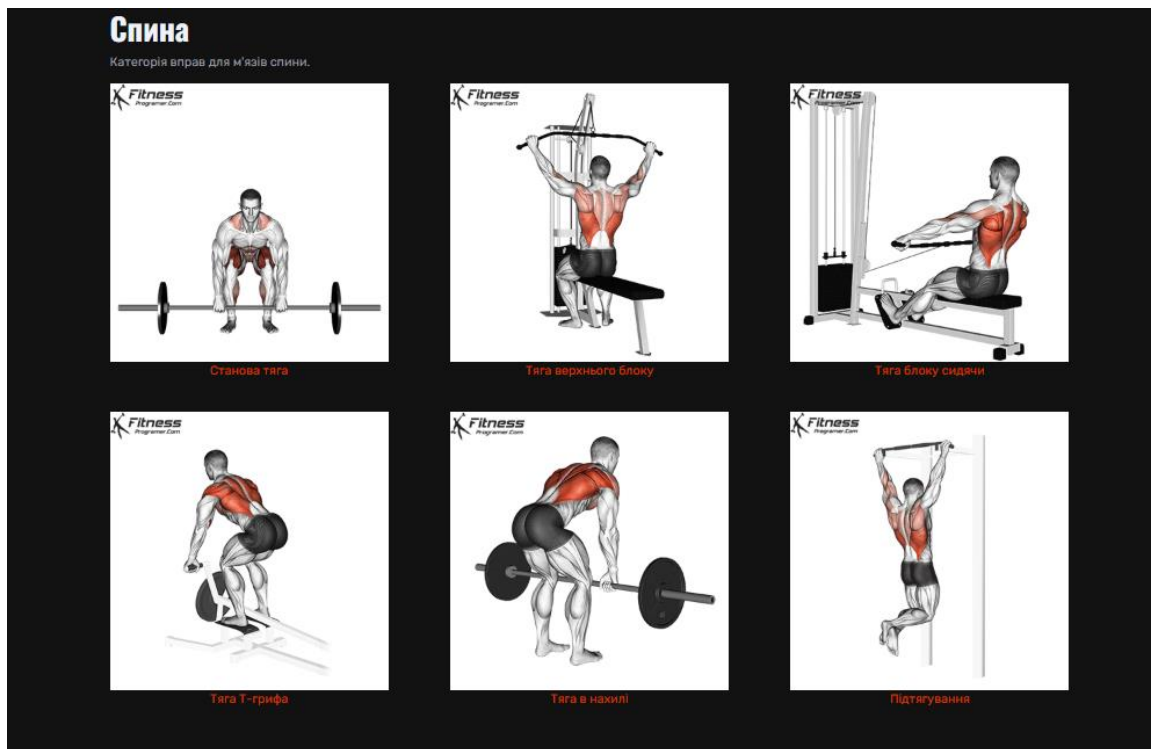


Рисунок 3.15 – Категорія вправ на м'язи спини

Користувач може вибрати вправу на задану категорії та переглянути вправу. Користувач бачить основну інформацію по вправі, інструкцію до виконання та відео-урок.

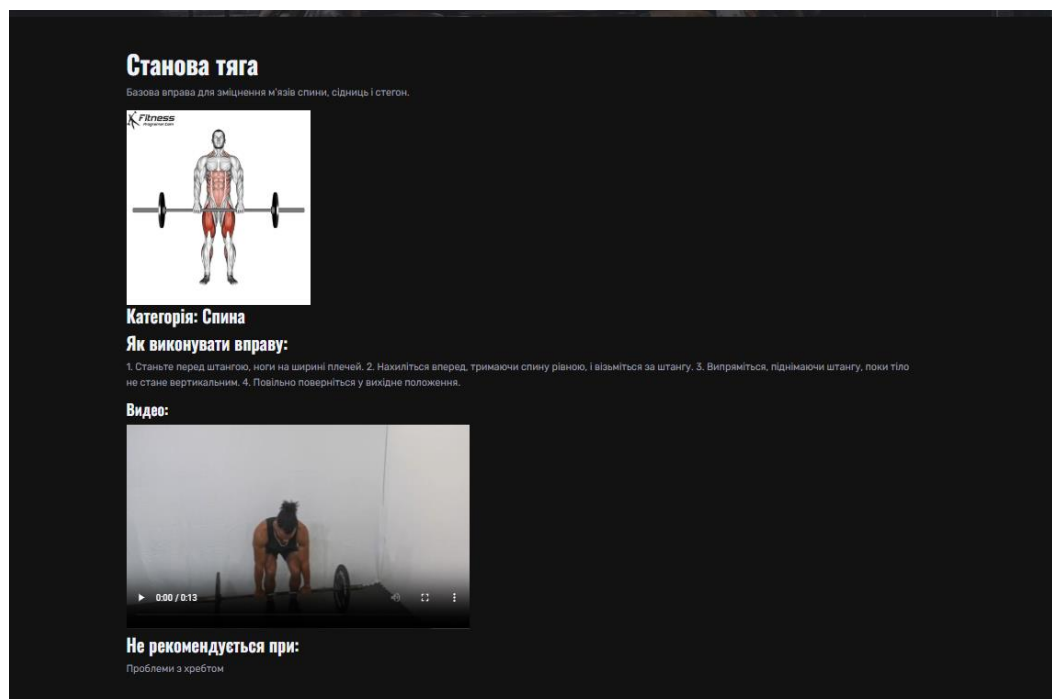


Рисунок 3.16 – Сторінка вправи

3.3 Тестування роботи програмного комплексу

Для коректної роботи додатка для аналізу тренувальних показників та харчування необхідно проводити ретельне тестування, яке допомагає виявити слабкі місця та можливі помилки в роботі системи. Для цього виконуємо основні дії, які повинен виконувати користувач.

Реєстрація облікового запису. Користувач, який потрапив на головну сторінку додатка, має можливість зареєструватись. Натиснувши кнопку “Зареєструватись”, користувач заповнює форму реєстрації, вказуючи дані (ім'я, email, пароль...), та надсилає її. Користувач успішно пройшов верифікацію та обліковий запис створено. Вхід до персонального кабінету: Користувач із зареєстрованим акаунтом переходить до свого профілю через головну сторінку. Користувач потрапляє до персонального кабінету, де може переглядати свої дані, харчові раціони, тренувальні плани та статистику. Користувач успішно перейшов до свого кабінету та має доступ до його функцій.

Взаємодія з функціоналом кабінету (Раціони, Статистика, Тренування). Користувач має доступ до персональних даних у кабінеті. Перехід до вкладок "Раціони", "Статистика" чи "Тренування", де користувач може переглядати, додавати, редагувати чи видаляти дані. Користувач взаємодіє з необхідними даними, а всі зміни зберігаються у базі. Користувач успішно виконав необхідні дії та дані збережено.

Взаємодія з елементами головної сторінки: Користувач на головній сторінці має доступ до базових функцій додатка. Користувач переглядає доступні функції, такі як Довідник продуктів чи вправ та калькулятори.

Користувач може взаємодіяти з елементами, переглядати повну інформацію та взаємодіяти з ними.

Користувач має доступ до необхідного функціоналу і може працювати з елементами головної сторінки.

Висновки до третього розділу

У даному розділі було детально розглянуто інструкцію щодо розгортання додатка на локальній машині. Описано всі необхідні технології та дії, які потрібно виконати для налаштування середовища. Зокрема, були обговорені етапи встановлення PHP, MySQL, Node.js, Composer, а також правильна конфігурація файлу .env для забезпечення стабільної роботи системи. Також приділено увагу створенню та налаштуванню бази даних, яка є ключовим елементом додатка.

У розділі проаналізовано принцип роботи додатка, його структуру та основні компоненти. Було описано функціонал для різних типів користувачів, включаючи гостей, зареєстрованих користувачів. Для гостей додаток дозволяє ознайомитися із загальною інформацією, переглядати функції системи та здійснювати реєстрацію. Зареєстровані користувачі отримують доступ до розширеного функціонала, включаючи створення персоналізованих раціонів, аналіз прогресу в тренуваннях та управління власними даними.

Після вивчення роботи інтерфейсу додатка було проведено тестування. Метою тестування було виявлення можливих помилок у роботі системи та їх усунення. Тестування охопило ключові елементи додатка, включаючи процес реєстрації, авторизацію, взаємодію з елементами персонального кабінету, генерацію планів харчування та управління тренуваннями. За підсумками тестування вдалося забезпечити стабільну роботу системи та відповідність функціонала визначеним вимогам.

Таким чином, у даному розділі описано всі аспекти створення, налаштування, роботи та тестування додатка, що демонструє його готовність до використання кінцевими користувачами.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи магістра було проведено комплексне наукове дослідження, спрямоване на вивчення впливу харчування та тренувань на стан здоров'я і фізичну підготовку людини. Особливу увагу було приділено збору статистичних даних та їх подальшому аналізу для розробки ефективних підходів до планування харчування і тренувальних програм. У ході роботи здійснювався збір даних про стан здоров'я, фізичні показники, звички харчування та рівень фізичної активності досліджуваних осіб. На основі зібраної інформації виконувалися розрахунки індивідуальних норм калорійності, потреби в макро- і мікронутрієнтах за допомогою науково обґрунтованих формул, а також аналіз змін у фізичних показниках протягом визначеного періоду часу.

Отримані результати дозволили встановити, що контроль за споживанням калорій і нутрієнтів, у поєднанні з регулярною фізичною активністю, позитивно впливає на зменшення ризиків розвитку хронічних захворювань і сприяє досягненню індивідуальних цілей, таких як зниження ваги, набір м'язової маси чи підтримання фізичної форми. Використані методи забезпечили високий рівень персоналізації рекомендацій, що підвищило їхню ефективність для кожного учасника дослідження.

Зібрані дані та розроблені алгоритми можуть слугувати основою для подальших наукових досліджень у цій галузі, дозволяючи проводити масштабний аналіз ефективності методик серед різних демографічних груп. Перспективним напрямком є впровадження цифрових інструментів, які забезпечать збір великих обсягів даних у режимі реального часу, покращення адаптації рекомендацій та створення індивідуалізованих стратегій для покращення якості життя. Результати цієї роботи підкреслюють важливість комплексного підходу до аналізу харчування та фізичної активності, що може бути використано для розробки науково обґрунтованих рекомендацій для підтримки здоров'я і фізичної активності населення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. “Nutrient Recommendations”, The Office of Dietary Supplements [Електронний ресурс], 1994-2024. Режим доступу до ресурсу: <https://ods.od.nih.gov/HealthInformation/nutrientrecommendations.aspx>.
2. “MVP патерн”, DEV Community [Електронний ресурс], 2016-2024. Режим доступу до ресурсу: <https://dev.to/binoy123/architectural-pattern-model-view-presenter-mvp-28hl>.
3. “MVVM патерн”, Microsoft, 2001-2024 [Електронний ресурс]. Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm>.
4. “Bootstrap Documentation”, @mdo and @fat, 2010-2024. [Електронний ресурс]. Режим доступу до ресурсу: <https://getbootstrap.com/docs/5.3/getting-started/parcel/>.
5. “CSS Documentation”, mozilla.org, 1998–2024. [Електронний ресурс]. Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>.
6. “JavaScript ”, mozilla.org, 1998–2024. [Електронний ресурс]. Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.
7. “PHP Documentation”, The PHP Group, 2001-2024. [Електронний ресурс]. Режим доступу до ресурсу: <https://www.php.net/docs.php>.
8. “IDE PHPStorm”, JetBrains s.r.o., 2000-2024 [Електронний ресурс]. Режим доступу до ресурсу: <https://www.jetbrains.com/phpstorm/>.
9. “Installing MySQL”, Oracle, 1995-2024. [Електронний ресурс]. Режим доступу до ресурсу: <https://dev.mysql.com/doc/refman/8.0/en/windows-installation.html>.
10. “npm install”, npmjs, 2024. [Електронний ресурс]. Режим доступу до ресурсу: <https://www.npmjs.com/package/npm>.
11. “nodejs install”, OpenJS Foundation, 2024. [Електронний ресурс]. Режим доступу до ресурсу: <https://nodejs.org/en/download>.

12. “composer install”, Nils Adermann, Jordi Boggiano, 2024. [Электронный ресурс]. Режим доступа до ресурсу: <https://getcomposer.org/doc/00-intro.md>.
13. “MySQL Documentation” , Oracle, 1995-2024. [Электронный ресурс]. Режим доступа до ресурсу: <https://dev.mysql.com/doc/>.
14. “Bootstrap Templates”, BootstrapDash, 2024. [Электронный ресурс]. Режим доступа до ресурсу: <https://www.bootstrapdash.com/>.
15. “Email Delivery Platform - Mailtrap”, Railware Products Studio LLC [Электронный ресурс], 2024. Режим доступа до ресурсу: <https://mailtrap.io/>.
16. “laravel”, Laravel LLC, 2011-2024. [Электронный ресурс]. Режим доступа до ресурсу: <https://laravel.com/>.
17. “Basal metabolic rate”, FAO, 2004 – 2024 [Электронный ресурс]. Режим доступа до ресурсу: <https://www.fao.org/4/y5686e/y5686e07.htm>.
18. “Clinical Guidelines on the Identification, Evaluation, and Treatment of Overweight and Obesity in Adults”, NHLBI, 1998 [Электронный ресурс]. Режим доступа до ресурсу: <https://www.ncbi.nlm.nih.gov/books/NBK2003/>.
19. “ Healthy Living with Diabetes”, NIDDK, 1950 – 2024 [Электронный ресурс]. Режим доступа до ресурсу: <https://www.niddk.nih.gov/health-information/diabetes/overview/healthy-living-with-diabetes> .
20. “ Micronutrients for Health”, Linus Pauling Institute, 2024 [Электронный ресурс]. Режим доступа до ресурсу: https://lpi.oregonstate.edu/sites/lpi.oregonstate.edu/files/pdf/mic/micronutrients_for_health.pdf .
21. “Calories”, NCBI, 1994-2024 [Электронный ресурс]. Режим доступа до ресурсу: <https://www.ncbi.nlm.nih.gov/books/NBK499909/> .
22. A.Schwarzenegger 1999 *The New Encyclopedia of Modern Bodybuilding*, p.800.
23. D Epstein 2014 *The Sports Gene: Inside the Science of Extraordinary Athletic Performance*, p.368.

ДОДАТКИ

Код маршрутів додатка

```

Route::get('/', function () {
    return redirect('/home');
});
Route::get('/home', [App\Http\Controllers\HomeController::class, 'index'])->name('home');

// загальні маршрути
Route::get('/categories', [CategoryController::class, 'show'])->name('product.categories');
Route::get('/products', [ProductController::class, 'show'])->name('product.products');
Route::get('/categories/{category}', [ProductController::class, 'productsByCategory'])->
>name('categories.products');
Route::get('/products/{id}', [ProductController::class, 'getProductDetails'])->
>name('products.details');
Route::post('/product', [ProductController::class, 'store']); //додавання продукта
Route::post('/category', [CategoryController::class, 'store']); //додавання категорії
Route::get('/fat-calculator', [FatPercentageCalculatorController::class, 'index'])->name('fat-
calculator.index');
Route::post('/fat-calculator/calculate', [FatPercentageCalculatorController::class, 'calculate'])->
>name('fat-calculator.calculate');
Route::get('/bmr-calculator', [BMRCalculatorController::class, 'index'])->name('bmr-
calculator.index');
Route::post('/bmr-calculator/calculate', [BMRCalculatorController::class, 'calculate'])->name('bmr-
calculator.calculate');
Route::get('/max-rep', [MaxRepCalculatorController::class, 'index'])->name('max-rep.index');
Route::post('/max-rep/calculate', [MaxRepCalculatorController::class, 'calculate'])->name('max-
rep.calculate');
Route::get('/grain-calculator', [GrainWeightCalculatorController::class, 'index'])->name('grain-
calculator.index');
Route::post('/grain-calculator/calculate', [GrainWeightCalculatorController::class, 'calculate'])->
>name('grain-calculator.calculate');
Route::get('/sorted', [CategoryController::class, 'sortedProductsPage'])->name('sorted');

Route::middleware('auth')->group(function () {
    Route::get('/food-diary', [FoodDiaryIndexController::class, 'index'])->name('food-diary.index');
    Route::delete('/food-diary/item/{itemId}', [DiaryAddProductController::class, 'deleteProduct'])->
>name('food-diary.delete-product');
    Route::post('/food-diary/{diaryId?}/add-product', [DiaryAddProductController::class,
'addProduct'])->name('food-diary.add-product');
    Route::post('/food-diary/edit-product/{itemId}', [DiaryAddProductController::class,
'editProduct'])->name('food-diary.edit-product');
    Route::get('/food-diary/calendar', [DiaryCalendarController::class, 'calendar'])->name('food-
diary.calendar');
    Route::get('/food-diary/{date}', [DiaryCalendarController::class, 'getDiaryByDate'])->
>name('food-diary.byDate');
    Route::post('/food-diary/generate-plan', [DiaryOpenAiController::class, 'generatePlan'])->
>name('food-diary.generate-plan');
    Route::get('/food-diary/progress', [FoodDiaryIndexController::class, 'efficiencyGraph'])->
>name('food-diary.efficiency-graph');

```

```

Route::get('/preferences', [UserPreferenceController::class, 'index'])->name('preferences.index');
Route::post('/preferences', [UserPreferenceController::class, 'store'])->name('preferences.store');
Route::delete('/preferences/{id}', [UserPreferenceController::class, 'destroy'])->name('preferences.destroy');

Route::get('/workout-diary', [WorkoutDiaryController::class, 'index'])->name('workout-diary.index');
Route::put('/workout-diary/{id}', [WorkoutDiaryController::class, 'update'])->name('workout-diary.update');
Route::get('/health-issues', [HealthIssueController::class, 'index'])->name('health-issues.index');
Route::post('/health-issues', [HealthIssueController::class, 'store'])->name('health-issues.store');
Route::delete('/health-issues/{id}', [HealthIssueController::class, 'destroy'])->name('health-issues.destroy');
Route::post('/workout-diary/add-set', [WorkoutDiaryController::class, 'addSet'])->name('workout-diary.add-set');

Route::get('/profile', [ProfileController::class, 'showProfile'])->name('profile.main')->middleware('auth');

});

//вправи
Route::get('/exercise-categories', [ExerciseCategoryController::class, 'index'])->name('exercise_categories.index');
Route::get('/exercise-categories/{id}', [ExerciseCategoryController::class, 'show'])->name('exercise_categories.show');

Route::get('/exercise', [ExerciseController::class, 'index'])->name('exercise.index');
Route::get('/exercise/{id}', [ExerciseController::class, 'show'])->name('exercise.show');
Route::post('/exercise-categories/add', [ExerciseCategoryController::class, 'store'])->name('exercise_categories.store');
Route::post('/exercises/add', [ExerciseController::class, 'store'])->name('exercises.store');

// зміна замірів
Route::middleware('auth')->group(function () {
    Route::post('/update-measurements', [MeasurementController::class, 'updateMeasurements'])->name('measurements-update');
    Route::get('/measurements', [MeasurementController::class, 'showMeasurements'])->name('profile.measurement');
    Route::get('/measurements/{id}/edit', [MeasurementController::class, 'edit'])->name('measurements.edit');
    Route::put('/measurements/{id}', [MeasurementController::class, 'update'])->name('measurements.update');
    Route::delete('/measurements/{id}', [MeasurementController::class, 'destroy'])->name('measurements.destroy');
    Route::get('/measurements/graph', [MeasurementController::class, 'showResults'])->name('measurements.results');
});

Route::prefix('progress')->group(function () {
    Route::get('/calculate/{userId}/{exerciseId}', [ExerciseProgressController::class, 'calculateProgress'])

```

```

        ->name('progress.calculate');
Route::get('/get/{userId}/{exerciseId}', [ExerciseProgressController::class, 'getProgress'])
        ->name('progress.get');
Route::get('/chart/{userId}/{exerciseId}', [ExerciseProgressController::class, 'getProgressChart'])
        ->name('progress.chart');
});
Route::get('/progress/{exerciseId}/{range?}', [ExerciseProgressController::class, 'showProgress'])
        ->middleware('auth')
        ->name('progress.show');
Route::get('/progress', [WorkoutDiaryController::class, 'selectExercise'])->name('progress.select');

Route::post('/forecast', [FoodDiaryIndexController::class, 'calculateForecast'])->
>name('forecast.calculate');

Auth::routes();

```

ДОДАТОК Б

```

public function measurements()
{
    return $this->hasMany(UserMeasurement::class);
}
public function healthIssues()
{
    return $this->belongsToMany(
        HealthIssue::class,
        'user_health_issues',
        'user_id',
        'health_issue_id'
    )
        ->withPivot('id')
        ->withTimestamps();
}
public function userHealthIssues()
{
    return $this->hasMany(UserHealthIssue::class, 'user_id');
}
public function exerciseProgress()
{
    return $this->hasMany(ExerciseProgress::class);
}
public function workoutSets()
{
    return $this->hasMany(WorkoutSet::class);
}
public function productPreferences()
{
    return $this->hasMany(UserProductPreference::class);
}
public function calculateBodyFatPercentage()
{
    if ($this->gender === 'male') {

```

```

        return 86.01 * log10($this->waist - $this->neck) - 70.041 * log10($this->height) + 30.76;
    } else {
        return 163.205 * log10($this->waist + $this->hips - $this->neck) - 97.684 * log10($this->
height) - 106.387;
    }
}

public function calculateBMR()
{
    $weight = $this->weight;
    $height = $this->height;
    $age = $this->age;
    $gender = $this->gender;
    if ($this->weight <= 0 || $this->height <= 0 || $this->age <= 0) {
        return 0;
    }
    $bmrHarrisBenedict = $gender === 'male'
        ? 88.362 + (13.397 * $weight) + (4.799 * $height) - (5.677 * $age)
        : 447.593 + (9.247 * $weight) + (3.098 * $height) - (4.330 * $age);
    $bmrMifflinStJeor = $gender === 'male'
        ? (10 * $weight) + (6.25 * $height) - (5 * $age) + 5
        : (10 * $weight) + (6.25 * $height) - (5 * $age) - 161;
    $bodyFatPercentage = $this->calculateBodyFatPercentage();
    $leanMass = $weight * (1 - $bodyFatPercentage / 100);
    $bmrKatchMcArdle = 370 + (21.6 * $leanMass);
    $averageBMR = ($bmrHarrisBenedict + $bmrMifflinStJeor + $bmrKatchMcArdle) / 3;
    return $averageBMR;
}

public function calculateNutritionNorms()
{
    $activity = $this->activity_coefficient;
    $goal = $this->goal;

    $bmr = $this->calculateBMR();
    if ($bmr <= 0) {
        $bmr = 1;
    }

    $maintenanceCalories = $bmr * $activity;

    switch ($goal) {
        case 'weight_loss':
            $calories = $maintenanceCalories - $maintenanceCalories * 0.22;
            break;
        case 'weight_gain':
            $calories = $maintenanceCalories + $maintenanceCalories * 0.22;
            break;
        default:
            $calories = $maintenanceCalories;
            break;
    }
}

```

```

    if ($calories < 0) {
        $calories = 0;
    }

    $proteins = 0;
    $fats = 0;

    if ($this->gender === 'female') {
        if ($this->calculateBodyFatPercentage() <= 20) {
            $proteins = $goal === 'weight_gain' ? $this->weight * 1.8 : $this->weight * 1.6;
            $fats = $goal === 'weight_gain' ? $this->weight * 1.1 : $this->weight * 1;
        } elseif ($this->calculateBodyFatPercentage() > 20 && $this->calculateBodyFatPercentage() <= 30) {
            $proteins = $goal === 'weight_gain' ? $this->weight * 2 : $this->weight * 1.8;
            $fats = $goal === 'weight_gain' ? $this->weight * 1 : $this->weight * 0.9;
        } elseif ($this->calculateBodyFatPercentage() > 30) {
            $proteins = $goal === 'weight_gain' ? $this->weight * 1.8 : $this->weight * 1.6;
            $fats = $goal === 'weight_gain' ? $this->weight * 0.9 : $this->weight * 0.8;
        }
        if ($calories < 1200) {
            $calories = 1200;
        }
    } else {
        if ($this->calculateBodyFatPercentage() <= 15) {
            $proteins = $goal === 'weight_gain' ? $this->weight * 2.1 : $this->weight * 2;
            $fats = $goal === 'weight_gain' ? $this->weight * 1 : $this->weight * 0.9;
        } elseif ($this->calculateBodyFatPercentage() > 15 && $this->calculateBodyFatPercentage() <= 25) {
            $proteins = $goal === 'weight_gain' ? $this->weight * 2 : $this->weight * 1.85;
            $fats = $goal === 'weight_gain' ? $this->weight * 1 : $this->weight * 0.8;
        } elseif ($this->calculateBodyFatPercentage() > 25 && $this->calculateBodyFatPercentage() <= 50) {
            $proteins = $goal === 'weight_gain' ? $this->weight * 1.9 : $this->weight * 1.95;
            $fats = $goal === 'weight_gain' ? $this->weight * 0.9 : $this->weight * 0.7;
        }
    }
    $carbs = ($calories - ($proteins * 4) - ($fats * 9)) / 4;
    return [
        'calories' => round($calories, 2),
        'proteins' => round($proteins, 2),
        'fats' => round($fats, 2),
        'carbs' => round($carbs, 2),
    ];
}

public function calculateVitaminMineralNorms()
{
    $norms = [
        'calcium' => 1000, // mg
        'iron' => $this->gender === self::GENDER_FEMALE ? 18 : 8, // mg
        'magnesium' => $this->gender === self::GENDER_FEMALE ? 310 : 400, // mg
        'phosphorus' => 700, // mg
    ];
}

```

```

        'potassium' => 3500, // МГ
        'sodium' => 2300, // МГ
        'zinc' => $this->gender === self::GENDER_FEMALE ? 8 : 11, // МГ
        'copper' => 0.9, // МГ
        'selenium' => 55, // МКТ
        'vitamin_c' => 75, // МГ
        'vitamin_d' => 600, // МЕ
        'vitamin_e' => 15, // МГ
        'vitamin_b6' => 1.3, // МГ
        'folic_acid' => 400, // МКТ
    ];

    switch ($this->goal) {
        case self::GOAL_WEIGHT_LOSS:
            $norms['magnesium'] += 10;
            $norms['vitamin_c'] += 15;
            break;
        case self::GOAL_WEIGHT_GAIN:
            $norms['calcium'] += 200;
            $norms['phosphorus'] += 100;
            break;
    }

    return $norms;
}

```

ДОДАТОК В

```

public function calculateNutritionNorms()
{
    $activity = $this->activity_coefficient;
    $goal = $this->goal;

    $bmr = $this->calculateBMR();
    if ($bmr <= 0) {
        $bmr = 1;
    }

    $maintenanceCalories = $bmr * $activity;

    switch ($goal) {
        case 'weight_loss':
            $calories = $maintenanceCalories - $maintenanceCalories * 0.22;
            break;
        case 'weight_gain':
            $calories = $maintenanceCalories + $maintenanceCalories * 0.22;
            break;
        default:
            $calories = $maintenanceCalories;
            break;
    }
}

```

```

if ($calories < 0) {
    $calories = 0;
}

$proteins = 0;
$fats = 0;

if ($this->gender === 'female') {
    if ($this->calculateBodyFatPercentage() <= 20) {
        $proteins = $goal === 'weight_gain' ? $this->weight * 1.8 : $this->weight * 1.6;
        $fats = $goal === 'weight_gain' ? $this->weight * 1.1 : $this->weight * 1;
    } elseif ($this->calculateBodyFatPercentage() > 20 && $this->calculateBodyFatPercentage()
<= 30) {
        $proteins = $goal === 'weight_gain' ? $this->weight * 2 : $this->weight * 1.8;
        $fats = $goal === 'weight_gain' ? $this->weight * 1 : $this->weight * 0.9;
    } elseif ($this->calculateBodyFatPercentage() > 30) {
        $proteins = $goal === 'weight_gain' ? $this->weight * 1.8 : $this->weight * 1.6;
        $fats = $goal === 'weight_gain' ? $this->weight * 0.9 : $this->weight * 0.8;
    }
    if ($calories < 1200) {
        $calories = 1200;
    }
} else {
    if ($this->calculateBodyFatPercentage() <= 15) {
        $proteins = $goal === 'weight_gain' ? $this->weight * 2.1 : $this->weight * 2;
        $fats = $goal === 'weight_gain' ? $this->weight * 1 : $this->weight * 0.9;
    } elseif ($this->calculateBodyFatPercentage() > 15 && $this->calculateBodyFatPercentage()
<= 25) {
        $proteins = $goal === 'weight_gain' ? $this->weight * 2 : $this->weight * 1.85;
        $fats = $goal === 'weight_gain' ? $this->weight * 1 : $this->weight * 0.8;
    } elseif ($this->calculateBodyFatPercentage() > 25 && $this->calculateBodyFatPercentage()
<= 50) {
        $proteins = $goal === 'weight_gain' ? $this->weight * 1.9 : $this->weight * 1.95;
        $fats = $goal === 'weight_gain' ? $this->weight * 0.9 : $this->weight * 0.7;
    }
}

$carbs = ($calories - ($proteins * 4) - ($fats * 9)) / 4;
return [
    'calories' => round($calories, 2),
    'proteins' => round($proteins, 2),
    'fats' => round($fats, 2),
    'carbs' => round($carbs, 2),
];
}

```

ДОДАТОК Г

```

public function generatePlan(Request $request)
{
    $products = UserProductPreference::where('user_id', auth()->id())
        ->with('product')
        ->get()
        ->pluck('product.name')
}

```

```

->toArray();

if (empty($products)) {
    return response()->json(['error' => 'Список бажаних продуктів пустий.'], 400);
}

// Рассчитываем нормы КБЖУ для пользователя
$user = Auth::user();
$nutritionNorms = $user->calculateNutritionNorms();

$openai = OpenAI::client(config('openai.api_key'));

try {
    $response = $openai->chat()->create([
        'model' => 'gpt-3.5-turbo',
        'messages' => [
            [
                'role' => 'system',
                'content' => 'Ти професійний дієтолог. Твоє завдання – скласти план харчування
для користувача з урахуванням його норм КБЖУ.
Використовуй лише перелічені продукти. Відповідай у форматі JSON. Вкажи назви
продуктів та їх кількість у грамах, при цьому намагайся дотриматися норм::
калорії - ' . $nutritionNorms['calories'] . ' ккал, білки - ' . $nutritionNorms['proteins']
. ' г, жири - ' . $nutritionNorms['fats'] . ' г,
вуглеводи - ' . $nutritionNorms['carbs'] . ' г. Приклад структури: {"сніданок":
[{"назва": "Яйця", "вага": 100}],
"обід": [{"назва": "Гречана ", "кількість": 200}]}'. Відхилення за білками,
жирами та вуглеводами не повинно перевищувати 10% від норми.'
            ],
            [
                'role' => 'user',
                'content' => 'Склади план харчування з продуктів: ' . implode(', ', $products) . '. На
1 день.',
            ],
        ],
        'temperature' => 0.2,
        'max_tokens' => 1000,
    ]);
} catch (\Exception $e) {
    return response()->json(['error' => 'Помилка зв'язку з OpenAI: ' . $e->getMessage()], 500);
}

$cleanedContent = trim($response['choices'][0]['message']['content']);
$cleanedContent = preg_replace('/\s+/', ' ', $cleanedContent);
$cleanedContent = str_replace(["\n", "\r"], "", $cleanedContent);
$cleanedContent = preg_replace('/\s*\/', '/', $cleanedContent);
$cleanedContent = preg_replace('/\s*]/', ']', $cleanedContent);

$plan = json_decode($cleanedContent, true);

if (json_last_error() !== JSON_ERROR_NONE) {
    return response()->json(['error' => 'Помилка декодування JSON: ' . json_last_error_msg()],

```

```

500);
}

$userId = auth()->id();
$date = $request->input('date') ?? now()->format('Y-m-d');

FoodDiary::where('user_id', $userId)->where('date', $date)->delete();

$mealTimes = [
    'сніданок' => ['meal' => 'breakfast', 'time' => '09:00:00'],
    'обід' => ['meal' => 'lunch', 'time' => '13:00:00'],
    'вечеря' => ['meal' => 'dinner', 'time' => '18:00:00'],
    'перекус' => ['meal' => 'snack', 'time' => '16:00:00'],
];

foreach ($mealTimes as $mealTimeRu => $mealTimeData) {
    if (isset($plan[$mealTimeRu])) {
        $foodDiary = FoodDiary::where([
            'user_id' => $userId,
            'date' => $date,
            'meal_time' => $mealTimeData['meal'],
        ])->first();

        if (!$foodDiary) {
            $foodDiary = new FoodDiary();
            $foodDiary->user_id = $userId;
            $foodDiary->date = $date;
            $foodDiary->meal_time = $mealTimeData['meal'];
            $foodDiary->total_calories = 0;
            $foodDiary->total_proteins = 0;
            $foodDiary->total_fats = 0;
            $foodDiary->total_carbs = 0;
            $foodDiary->time = $mealTimeData['time'];
            $foodDiary->save();
        }

        foreach ($plan[$mealTimeRu] as $productData) {
            $productName = $productData['назва'];
            $productAmount = $productData['вага'];

            $product = Product::where('name', $productName)->first();

            if ($product) {
                $calories = ($product->energy_kcal / 100) * $productAmount;
                $proteins = ($product->protein / 100) * $productAmount;
                $fats = ($product->fat / 100) * $productAmount;
                $carbs = ($product->carbs / 100) * $productAmount;

                FoodDiaryItem::create([
                    'food_diary_id' => $foodDiary->id,
                    'product_id' => $product->id,
                    'amount' => $productAmount,
                ]);
            }
        }
    }
}

```

```

        'calories' => $calories,
        'proteins' => $proteins,
        'fats' => $fats,
        'carbs' => $carbs,
    );
    $foodDiary->increment('total_calories', $calories);
    $foodDiary->increment('total_proteins', $proteins);
    $foodDiary->increment('total_fats', $fats);
    $foodDiary->increment('total_carbs', $carbs);
    }
    }
}

return redirect()->route('food-diary.index', ['date' => $date])->with('success', 'План харчування
успішно згенерований та доданий до щоденника!');
}

```

ДОДАТОК Д

```

public function calculateScenario($dailySummaries, $currentWeight, $startDate, $weeks,
$scenarioType)
{
    $totalChange = 0;
    $dailyData = [];
    $user = Auth::user();

    $norms = $user->calculateNutritionNorms();
    $calorieNorm = $norms['calories'];

    $start = Carbon::parse($startDate);
    $end = $start->copy()->addDays($weeks * 7 - 1);

    $summaryIndex = 0;

    while ($start->lte($end)) {
        $date = $start->toDateString();
        $calories = null;

        if (
            $summaryIndex < count($dailySummaries) &&
            $dailySummaries[$summaryIndex]->date === $date
        ) {
            if ($dailySummaries[$summaryIndex]->total_calories > 0) {
                $calories = $dailySummaries[$summaryIndex]->total_calories;
            }
            $summaryIndex++;
        }
        $bmr = $user->calculateBMR();
        if ($calories === null) {
            $calories = $calorieNorm;
        }

        if ($scenarioType === 'optimistic') {

```

```

    $calories = $calorieNorm ;
    if ($user->goal === 'weight_loss') {
        $calories -= 500;
    } elseif ($user->goal === 'weight_gain') {
        $calories += 500;
    }
    } elseif ($scenarioType === 'pessimistic') {
        $calories += $calorieNorm * 0.2;
    } elseif ($scenarioType === 'current') {
        if ($calories === null) {
            $calories = $calorieNorm;
        }
    }
    }
    $calorieDeficit = $calories - $norms['calories'];
    $dailyChange = $calorieDeficit / 7200;
    $totalChange += $dailyChange;

    $dailyData[] = [
        'date' => $date,
        'weight' => round($currentWeight + $totalChange, 2),
    ];

    $start->addDay();
}

return $dailyData;
}

```