НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Програмне забезпечення для розрахунку та трекінгу здорового харчування

Виконав студент IV курсу, групи ІП-11
(шифр групи)
Гіжицький Даниїл Олександрович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.т.н., доц., Ліщук К. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант доцент, к.т.н., доц., Ліщук К. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент кафедри ІСТ, к.т.н., доц., Писаренко А. В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2025

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Гіжицькому Даниїлу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Програмне забезпечення для розрахунку та трекінгу здорового харчування

керівник проєкту Ліщук Катерина Ігорівна, к.т.н доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «23» травня 2025 р. №1705-с

2. Термін подання студентом проєкту «16» червня 2025 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.
3) Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання
4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.
5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема бази даних

3) Схема структурна компонентів програмного забезпечення

4) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «15» березня 2025 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	15.03.2025	
2	Аналіз існуючих методів розв’язання задачі	25.03.2025	
3	Постановка та формалізація задачі	05.04.2025	
4	Розробка інформаційного забезпечення	15.04.2025	
5	Алгоритмізація задачі	20.04.2025	
6	Обґрунтування вибору використаних технічних засобів	25.04.2025	
7	Розробка програмного забезпечення	15.05.2025	
8	Налагодження програми	20.05.2025	
9	Виконання графічних документів	26.05.2025	
10	Оформлення пояснювальної записки	28.05.2025	
11	Подання ДП на попередній захист	04.06.2025	
12	Подання ДП рецензенту	10.06.2025	
13	Подання ДП на основний захист	16.06.2025	

Студент _____ Даниїл ГІЖИЦЬКИЙ
(підпис) (ініціали, прізвище)

Керівник _____ Катерина ЛІЩУК
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п’яти розділів, містить 36 таблиць, 44 рисунків та 37 джерел – загалом 151 сторінка.

Дипломний проєкт присвячений розробці програмного забезпечення для розрахунку та трекінгу здорового харчування

Мета : покращення існуючих трекерів харчування для користувачів з врахуванням їх способу життя шляхом реалізації програмного забезпечення яке буде підлаштовуватися під параметри користувача.

У першому розділі наведено аналіз предметної області трекерів харчування, порівняльний аналіз між реалізованим програмним застосунком та аналогами, виділено їх переваги та недоліки.

У другому розділі наведено основні варіанти використання програмного забезпечення, функціональні та нефункціональні вимоги до програмного забезпечення та сформульовано системні вимоги до нього, наведено результати аналізу економічних показників програмного забезпечення та наведено приблизну його собівартість.

У третьому розділі наведено архітектуру програмного забезпечення, наведено обґрунтування прийнятих основних архітектурних рішень та застосування використаних засобів розробки.

У четвертому розділі наведено опис процесів тестування та контрольного прикладу.

У п’ятому розділі наведено опис обраного способу розгортання розробленого програмного забезпечення, розгортання програмного забезпечення, супровід програмного забезпечення.

Програмне забезпечення впроваджено на хостингу Azure.

КЛЮЧОВІ СЛОВА: : ВЕБ-ЗАСТОСУВАННЯ, КОНТРОЛЬ ХАРЧУВАННЯ, КОРИСТУВАЧ, N-TIER.

ABSTRACT

The explanatory note of the diploma project consists of five chapters, includes 36 tables, 44 figures, and 37 sources — totaling 151 pages.

The diploma project is dedicated to the development of software for calculating and tracking healthy nutrition.

Objective: To improve existing nutrition trackers for users by taking into account their lifestyle through the implementation of software that adapts to user parameters.

The first chapter provides an analysis of the subject area of nutrition trackers, a comparative analysis between the developed software application and its analogues, highlighting their advantages and disadvantages.

The second chapter presents the main use cases of the software, functional and non-functional requirements for the software, formulates the system requirements, provides an analysis of the economic indicators of the software, and estimates its approximate cost.

The third chapter describes the software architecture, justifies the main architectural decisions made, and the use of selected development tools.

The fourth chapter outlines the testing processes and a control example.

The fifth chapter describes the chosen method of deploying the developed software, software deployment, and maintenance.

The software has been deployed on Azure hosting.

KEYWORDS: WEB APPLICATION, NUTRITIONAL CONTROL, USER, N-TIER.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-1106.045440.01.91	Технічне завдання	12	
3	А4	КПІ.ІП-1106.045440.02.81	Пояснювальна записка	79	
4	А4	КПІ.ІП-1106.045440.03.12	Текст програми	27	
5	А4	КПІ.ІП-1106.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІП-1106.045440.05.34	Керівництво користувача	9	
7	А3	КПІ.ІП-1106.045440.06.99	Графічний матеріал	4	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІП-1106.045440.00.90		
Розроб.		Гіжицький Д.О.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів 1
Перевір.		Ліщук К.І.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-11	
Н.контр.		Ліщук К.І.					
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРАХУНКУ ТА ТРЕКІНГУ ЗДОРОВОГО ХАРЧУВАННЯ

Технічне завдання

КПІ.ІП-1106.045440.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Катерина ЛІЩУК

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Даниїл ГІЖИЦЬКИЙ

Київ – 2025

Зміст

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для розрахунку та трекінгу здорового харчування.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення для розрахунку та трекінгу здорового харчування HealthyTracker, котре використовується для контролю здорового харчування та призначена для контролю здорового харчування.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки HealthyTracker є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для розрахунку та трекінгу здорового харчування.

Метою розробки є покращення існуючих трекерів харчування для користувачів з врахуванням їх способу життя шляхом реалізації програмного забезпечення яке буде підлаштовуватися під параметри користувача.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Розрахунок калорійності

– користувач повинен мати змогу розрахувати норму харчування;
– користувач повинен мати змогу зберегти харчові цілі.

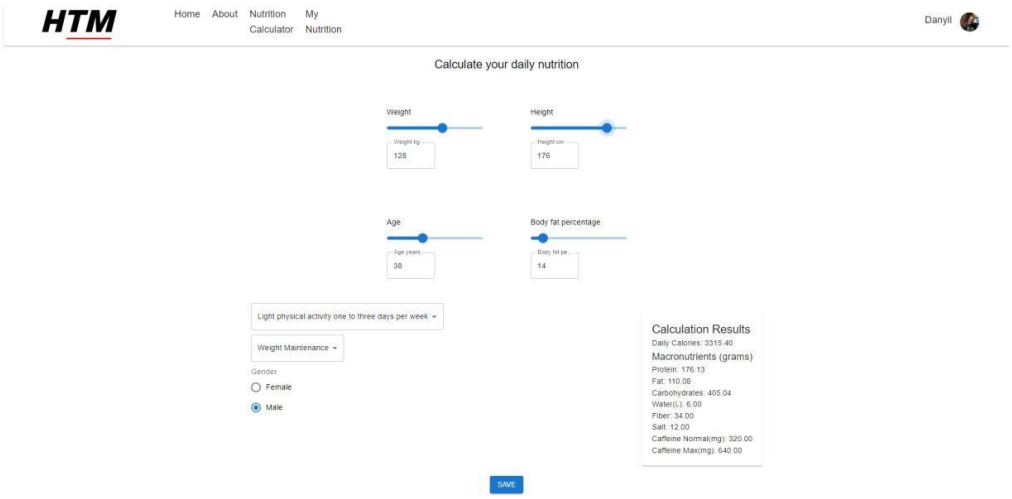


Рисунок 4.1 – Екранна форма головної сторінки

4.1.2 Перегляд щоденника харчування

– користувач повинен мати змогу додати продукти до щоденника;
– користувач повинен мати змогу перевірити виконання цілі.

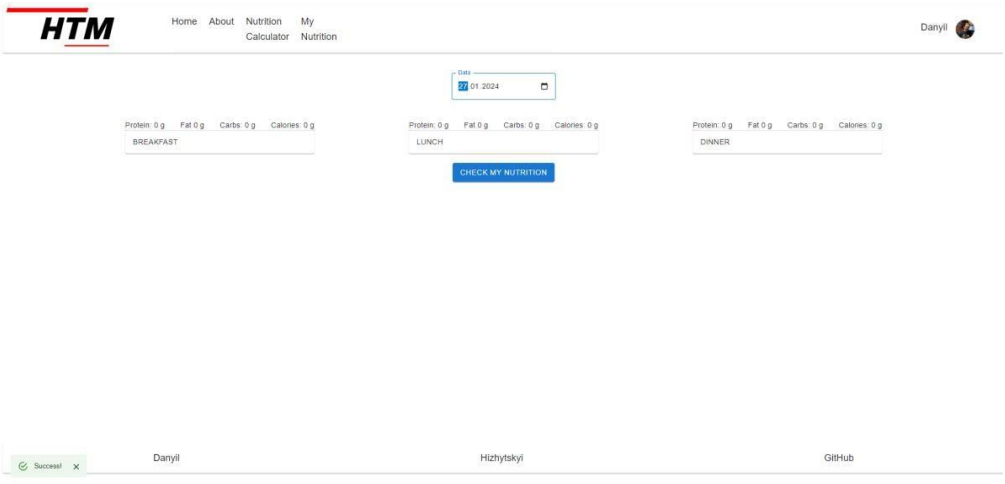


Рисунок 4.2 – Екранна форма ведення щоденника харчування

4.1.3 Перегляд інформації корисувача

– користувач повинен мати змогу змінювати інформацію про себе;
– користувач повинен мати змогу змінювати фото.

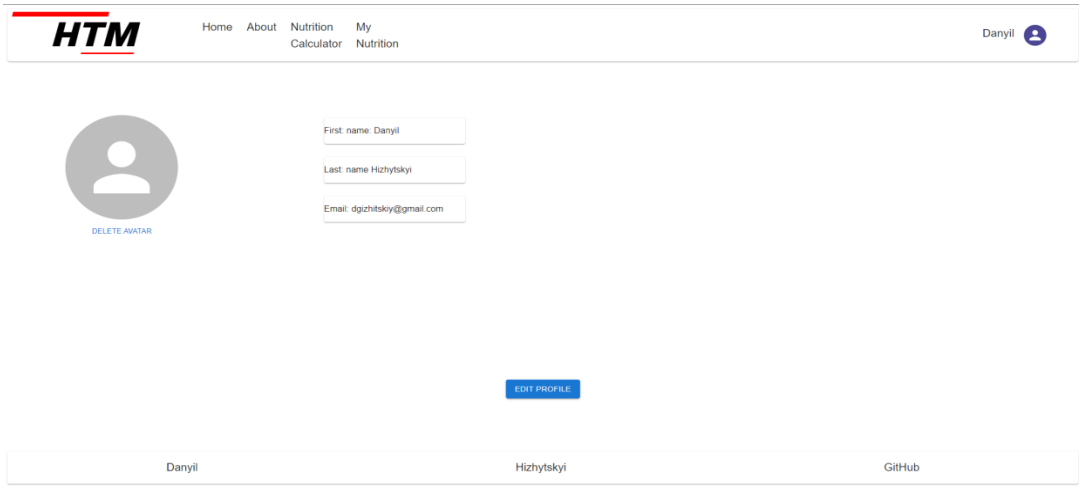


Рисунок 4.3 – Екранна форма акаунту користувача

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних. Забезпечити коректну роботу вебсервісу у випадку збоїв серверної частини, а також відловлювати та ховати помилки на клієнтській частині забезпечивши роботу решти частини сервісу.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Для обслуговування вебсервісу потрібні спеціалісти зі знаннями мови програмування C# та досвідом роботи з бібліотекою React, фреймворком

NextJS, HTML, scss, zustand для клієнтської частини та TypeScript і базою даних MS SQL для серверної частини. Також необхідний спеціаліст з навичками розгортання вебзастосунку і бази даних які використовуються у проекті.

4.4 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i5;
– об‘єм ОЗП: 8 Гб;
– підключення до мережі Інтернет зі швидкістю від 5 Мбіт/с.

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i7;
– об‘єм ОЗП: 16 Гб;
– підключення до мережі Інтернет зі швидкістю від 1000 мегабіт.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням усіх операційних систем, які забезпечують з'єднання з Інтернетом і можуть надати доступ до веб-застосунку через браузер.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в форматі JSON файлу.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в форматі JSON файлу.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування C#

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформах Rider, Visual Studio Code, Docker

4.5.5 Вимоги по представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді посилання на GitHub репозиторій з обмеженим доступом до перегляду та запису

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– текст програми;
– програма та методика тестування;
– керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;
– схема структурна компонентів програмного забезпечення;
– схема бази даних;
– архітектура програмного забезпечення;
– креслення вигляду екранних форм;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення рекомендованої літератури	15.03.2025	
2.	Аналіз існуючих методів розв’язання задачі	25.03.2025	Технічне завдання
3.	Постановка та формалізація задачі	05.04.2025	Специфікація програмного забезпечення
4.	Розробка інформаційного забезпечення	15.04.2025	
5.	Алгоритмізація задачі	20.04.2025	Формалізовані алгоритми програмного забезпечення
6.	Обґрунтування вибору використаних технічних засобів	25.04.2025	
7.	Розробка програмного забезпечення	15.05.2025	Тексти програмного забезпечення
8.	Налагодження програми	20.05.2025	Тести, результати тестування
9.	Виконання графічних документів	26.05.2025	Графічний матеріал проекту
10.	Оформлення пояснювальної записки	28.05.2025	Пояснювальна записка
11.	Подання ДП на попередній захист	04.06.2025	
12.	Подання ДП рецензенту	10.06.2025	
13.	Подання ДП на основний захист	16.06.2025	

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: Програмне забезпечення для розрахунку та трекінгу здорового харчування

КПІ.ІП-1106.045440.02.81

Київ = 2025

Зміст

ВСТУП

У сучасному світі, де щоденний темп життя стрімко зростає, більшість людей дедалі частіше веде малорухливий спосіб життя та рідко приділяє належну увагу харчуванню. Це, водночас, є критично важливим фактором для підтримання здоров’я та досягнення високих результатів у професійній, навчальній чи особистій діяльності. Наприклад, за умов інтенсивного розумового навантаження організм потребує раціону, багатого на антиоксиданти, повноцінні білки та складні вуглеводи. Проте, через постійний дефіцит часу й перевантаженість обов’язками, контроль за якістю щоденного харчування часто стає другорядним завданням.

Ще однією складністю є правильне формування індивідуального раціону. Дієтологія — це комплексна наукова галузь, що вимагає ґрунтовних знань і часу на опанування. Саме тому послуги фахівців у цій сфері залишаються дорогими та не завжди доступними для широкого кола людей. У зв’язку з цим виникає актуальна потреба в інструменті, який спростить процес створення персоналізованого плану харчування.

З огляду на те, що час набуває статусу найціннішого ресурсу, а інтернет став невід’ємною частиною повсякденності, оптимальним способом розв’язання подібних завдань є створення веб-застосунку. Такий інструмент, реалізований у вигляді мобільного або браузерного клієнта, дозволяє зручно отримувати доступ до функціоналу з будь-якого місця у зручний час, а також легко інтегруватися з іншими цифровими сервісами для комплексного підходу.

Хоч на ринку вже існують певні програмні рішення у сфері контролю харчування, більшість із них оперує усередненими

показниками без врахування індивідуальних потреб користувача, його стилю життя та цілей. Це створює потребу у розробці нової системи, здатної формувати точні рекомендації, адаптовані до конкретних особливостей кожної людини.

У межах даного проєкту планується детальний аналіз чинних веб-застосунків, які забезпечують підрахунок калорій і моніторинг харчування. На основі зібраних даних буде створена концепція нового, більш ефективного сервісу. Розроблений застосунок відповідатиме вимогам сучасного користувача, пропонуючи інструменти для зручного визначення калорійності продуктів, ведення щоденника харчування та формування персональних планів збалансованого раціону. Особливістю платформи стане використання технологій штучного інтелекту для підвищення точності рекомендацій та автоматизації процесу.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання дипломного проєктування

У сучасному світі, де технологічний прогрес постійно впливає на темп і характер суспільного розвитку, веб-застосунки перетворюються на невід'ємний інструмент ефективної діяльності підприємств і організацій. Це особливо важливо у сфері здорового способу життя, де критично необхідно надати користувачам доступний, зручний і функціональний сервіс для контролю харчування та підтримання загального добробуту. У цьому контексті завданням даного дипломного проєкту є створення інтерактивного веб-застосунку для моніторингу харчування, що поєднує персоналізовані поради з сучасними цифровими технологіями з метою покращення раціону користувачів. Розроблений сервіс має забезпечити можливість фіксації щоденного харчування, отримання рекомендацій щодо збалансованого раціону, а також контроль за динамікою досягнення індивідуальних цілей — усе це з урахуванням таких персональних характеристик, як професійна діяльність, стиль життя та фізіологічні особливості. Під час формулювання технічного завдання враховано ключові виклики, притаманні цій предметній галузі, зокрема потребу в точності, адаптивності та зручності використання, що є основоположними вимогами до програмних рішень у сфері здоров'я та харчування. При постановці завдання враховуються основні виклики цієї предметної області:

– персоналізація рекомендацій на основі сфери діяльності, способу життя фізіологічних даних, цілей та харчових уподобань;

– інтеграція з різними джерелами даних;
– побудова рекомендацій на основі персоналізованих даних з використання штучного інтелекту;
– технічні виклики: продуктивність та масштабованість.

Детальніше розглянемо проблеми та виклики.

Головна складність полягає у створенні раціону харчування, максимально адаптованого до конкретного користувача з урахуванням його індивідуальних характеристик і провідного виду діяльності в певний період часу. Більшість існуючих застосунків для моніторингу харчування використовують загальні схеми, які не враховують цих нюансів, що призводить до зниження ефективності досягнення поставлених цілей.

Контент, який міститься в системі, постійно змінюється: з’являються нові продукти, розширюються бази даних, інтегруються новітні дієтичні підходи. У зв’язку з цим платформа повинна бути здатна швидко адаптувати свої алгоритми без необхідності капітального перегляду всієї системи. Для реалізації цього необхідно впровадити два основні механізми: перший — покрокове оновлення інформаційних ресурсів щодо продуктів та нутрієнтів; другий — гнучку модульну структуру, що дозволяє оперативно додавати нові функції. Така архітектура гарантує актуальність рекомендацій і високий рівень персоналізації, одночасно забезпечуючи стабільність роботи.

Отже, основною метою даного проєкту є створення інтерактивного веб-застосунку, призначеного для контролю харчування, який поєднує інтелектуальні алгоритми аналізу харчової поведінки з інтуїтивно зрозумілим інтерфейсом. Програмне рішення повинно пропонувати персоналізовані поради на основі фізіологічних характеристик, цілей користувача та його вподобань. Одним із ключових компонентів стане механізм безперервного оновлення інформації, що дозволить системі

своєчасно реагувати на зміни в дієтології та харчових практиках. Успішна реалізація проєкту підвищить ефективність дотримання принципів здорового харчування, мінімізує потребу в регулярних консультаціях з фахівцями та зробить доступ до індивідуальних рекомендацій максимально зручним. Крім того, система сприятиме формуванню корисних харчових звичок, допомагаючи кожному користувачу досягати особистих результатів за допомогою технологічно розвинених інструментів аналізу.

1.2 Аналіз предметної області

У процесі дослідження предметної галузі було проаналізовано базові принципи та поточні методи формування оптимального раціону харчування відповідно до індивідуальних потреб користувачів.

У загальному випадку вирішення даної задачі здійснюється одним із двох основних способів, які застосовуються персоналізовано для кожної окремої особи.

Перший варіант передбачає регулярні консультації з кваліфікованими дієтологами або самостійне вивчення дієтології як наукової дисципліни. Такий підхід демонструє високу ефективність у контексті досягнення особистих цілей та глибокого налаштування раціону під конкретні фізіологічні потреби. Водночас, він є малопридатним у масовому використанні через значні витрати часу та коштів.

Другий підхід — це використання вже доступних цифрових рішень. Цей напрямок було проаналізовано детальніше.

Переважна більшість існуючих рішень функціонують як “тонкі” клієнти веб-застосунків, доступні у формі мобільних додатків або веб-інтерфейсів, що забезпечує зручний доступ до системи з різних пристроїв.

Окрему увагу було приділено аспекту зберігання та обробки персональної інформації, що є надзвичайно важливим для будь-якої

платформи такого типу. У межах розробки необхідною умовою є впровадження стандартів захисту персональних даних, зокрема відповідність вимогам GDPR[31], що гарантує безпечну та законну обробку конфіденційної інформації користувачів.

Серед характерних недоліків наявних програм можна виокремити обмежений рівень персоналізації: більшість із них базуються на усереднених нормах — калорійності, типовому наборі продуктів, рекомендованих годинах прийому їжі тощо. Такий підхід не враховує індивідуальні відмінності у способі життя чи фізіологічному стані.

Ще одним викликом є застарівання даних: у сучасних умовах швидкого розвитку харчової індустрії та змін у способі життя користувачів, оперативне оновлення інформації є критично необхідним. Натомість у багатьох існуючих рішеннях оновлення бази знань відбувається лише разом з релізами нових версій програмного забезпечення, що не забезпечує належної актуальності.

Паралельно із цим, все більше значення набуває впровадження технологій штучного інтелекту (ШІ), який уже активно застосовується в різних сферах повсякденного життя. Завдяки ШІ з’являється можливість забезпечення глибокої персоналізації та адаптації програмного рішення до змінних запитів користувача в режимі реального часу.

Таким чином, актуальність проєкту визначається потребою у створенні системи, здатної адаптувати рекомендації з харчування на основі індивідуальних характеристик користувача та специфіки його діяльності. За результатами аналізу предметної галузі та огляду аналогів, можна визначити ключові вимоги до майбутнього програмного продукту: веб-застосунок, що використовує можливості штучного інтелекту для забезпечення динамічної персоналізації і конкурентоспроможності на ринку.

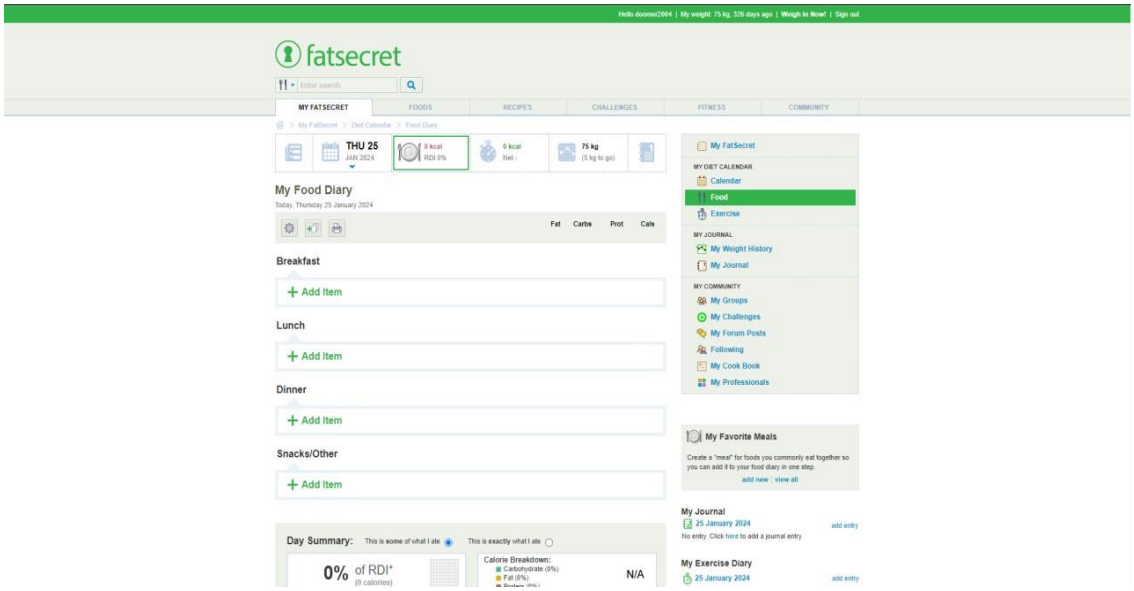
1.3 Аналіз існуючих рішень

Розглянемо сучасні алгоритмічні підходи та технічні рішення, що застосовуються у сфері здорового харчування, зокрема для реалізації програмного забезпечення, яке виконує функції розрахунку раціону та трекінгу харчових звичок. У подальшому буде проведено аналіз наявних програмних продуктів, допоміжних інструментів і технологій розробки, які можуть бути використані для створення ефективного веб-застосунку.

1.3.1 Аналіз відомих програмних продуктів

Оцінка ефективності веб-застосунків здійснюється на основі кількох ключових показників. Зокрема, продуктивність характеризується швидкістю обробки користувацьких запитів і здатністю системи функціонувати під високим навантаженням. Надійність відображає спроможність платформи працювати стабільно та відновлюватися після технічних збоїв. Масштабованість вказує на можливість системи адаптуватися до зростання кількості користувачів і обсягу інформації, пов’язаної з харчуванням. Захищеність системи охоплює запобігання несанкціонованому доступу та збереження конфіденційності особистих даних. Також до важливих характеристик належать сумісність з різними браузерами й пристроями, відкритість до впровадження нових технологій, а також зручність взаємодії з інтерфейсом. Не менш значущим є і показник повної вартості володіння, який враховує витрати на обслуговування, підтримку й розвиток застосунку.

1.3.1.1. FatSecret [1]



Риснок 1.1– Веб-застосунок FatSecret

Основні функціональні можливості: FatSecret дозволяє користувачам вести щоденник харчування, вказуючи спожиті продукти та кількість калорій, використовувати трекер фізичної активності, обмінюватися порадами та рецептами з іншими користувачами.

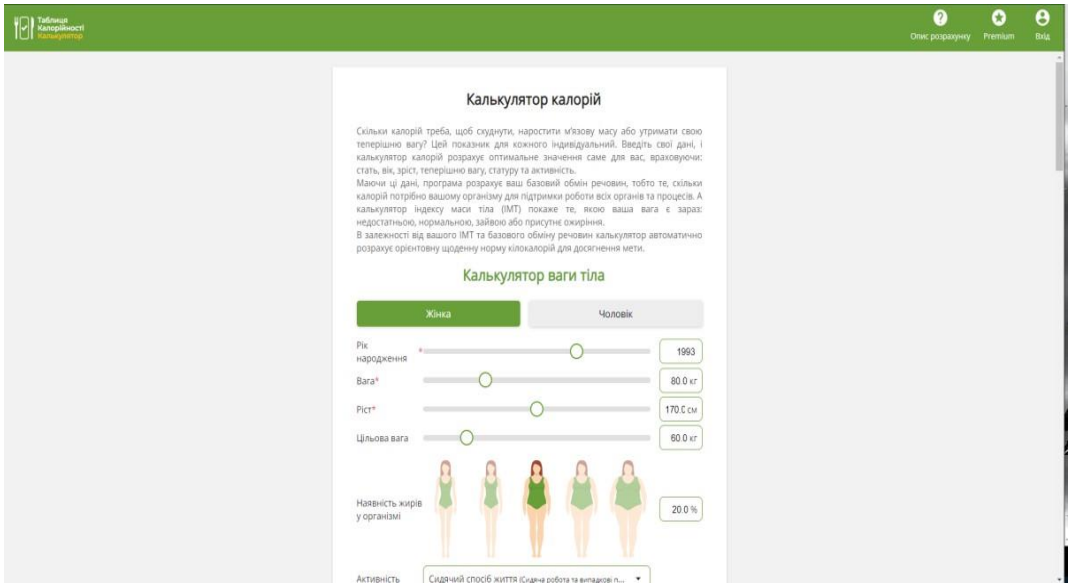
Переваги застосунку:

– застосунок є безкоштовним для завантаження та використання, що робить його доступним для широкого кола користувачів;
– графіки та статистика харчування;
– забезпечує інструменти для відстеження фізичної активності, ваги та інших факторів здоров'я.

Недоліки застосунку:

– інтерфейс може вимагати часу для освоєння;
– відсутність розрахунку калорійності в застосунку.

1.3.1.2. Калькулятор калорій [2]



Риснок 1.2 – Веб-застосунок Калькулятор калорій

Основні функціональні можливості: Калькулятор калорій дозволяє користувачам обчислювати кількість калорій у різних продуктах або стравах та визначати своє денне споживання калорій.

Переваги застосунку:

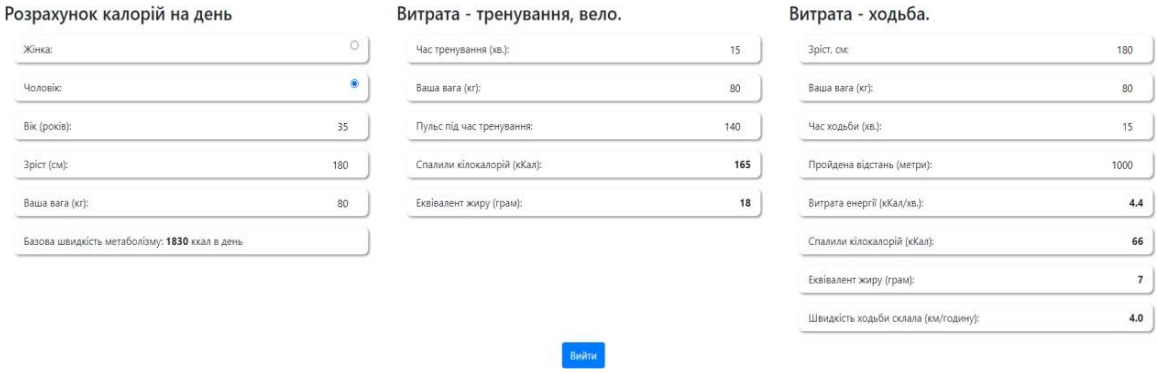
- простота використання;
- можливість швидко визначити кількість

калорій у продуктах.

Недоліки застосунку:

- застарілий інтерфейс;
- безкоштовна версія має досить обмежений функціонал.

1.3.1.3. kyiv-dpss.gov [3]



Риснок 1.3 – Веб-застосунок Калькулятор калорій

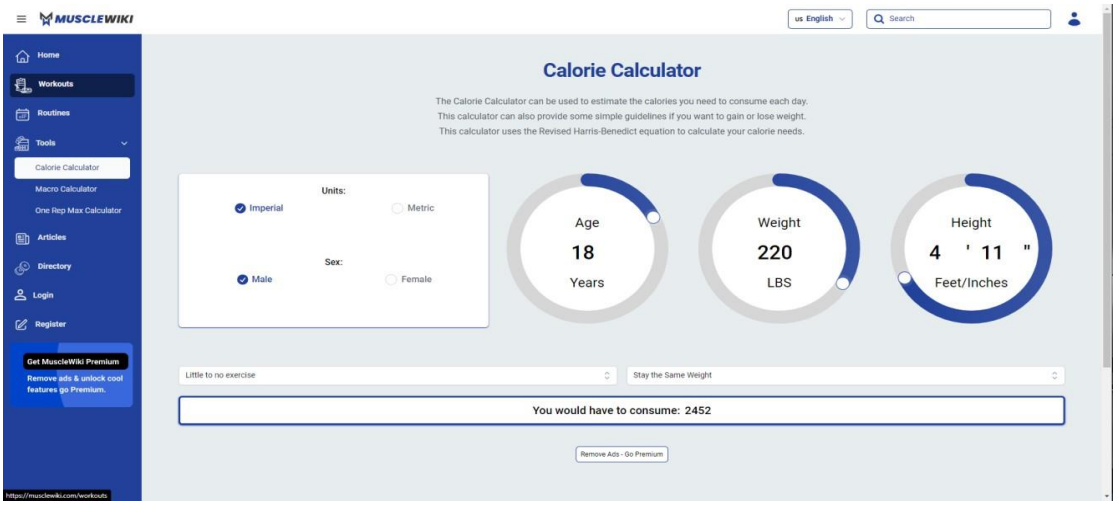
Основні функціональні можливості: Надає можливість визначити базовийметаболізм та витрати калорій відносно вело тренувань та пішої прогулянки Переваги застосунку:

– простота у використанні.

Недоліки застосунку:

– застарілий продукт.

1.3.1.4. MuscleWiki [4]



Риснок 1.4 – Веб-застосунок MuscleWiki

Основні функціональні можливості: Надає інформацію про різні вправи та тренування для розвитку м'язів, включаючи відео та інструкції.

Переваги застосунку:

– сучасний дизайн та зручний користувацький інтерфейс;

– детальна інформація для фітнесу та тренувань;

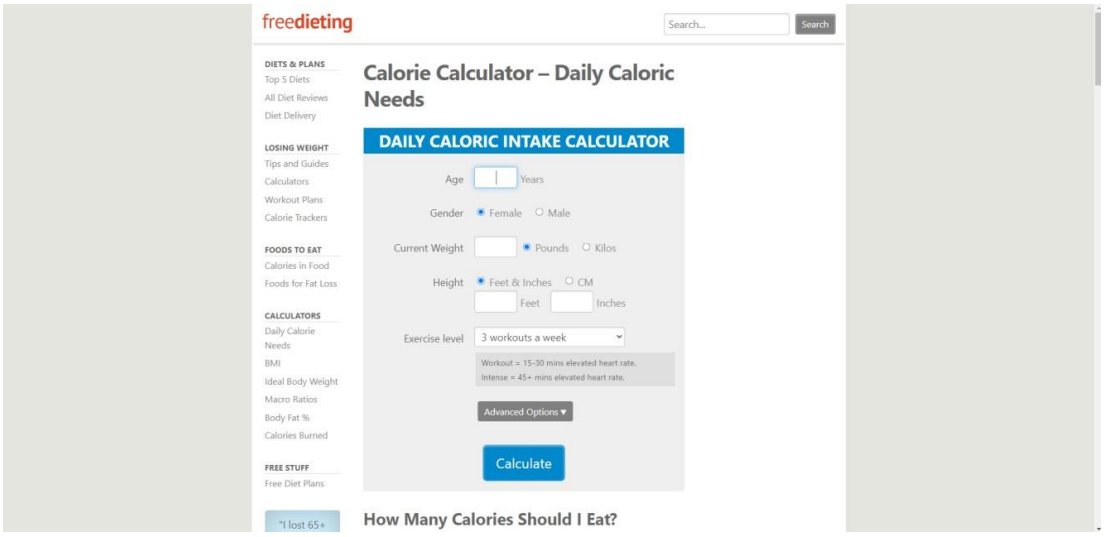
– вбудовані функції розрахунку калорійності та макроелементів.

Недоліки застосунку:

– відсутність щоденника харчування;

– основна частина функціоналу потребує підписки на продукт.

1.3.1.5. Freedieting [5]



Риснок 1.5 – Веб-застосунок Freedieting

Основні функціональні можливості: Надає інформацію та інструменти для планування дієти, включаючи калькулятор калорій, рецепти та статті про харчування.

Переваги застосунку:

- безкоштовний доступ;
- різноманітні інструменти для допомоги при плануванні дієти та контролікалорій.

Недоліки застосунку:

- застарілий інтерфейс;
- відсутність щоденника харчування.

Розглянувши наявні успішні IT-проекти, складемо таблицю порівняння.

Щоб здійснити якісний порівняльний аналіз веб-застосунків, призначених для моніторингу харчування, необхідно визначити показники, які найкраще відображають основні функціональні можливості та зручність використання для кінцевого користувача. Саме з цієї причини були сформульовані такі критерії оцінювання:

- інтуїтивно зрозумілий інтерфейс: Якість взаємодії користувача з веб-застосунком значною мірою залежить від зручності інтерфейсу. Простий і логічно побудований дизайн полегшує навігацію, сприяє кращому залученню користувачів і робить сервіс доступним навіть для тих, хто не має попереднього досвіду користування подібними платформами;
- функція розрахунку калорійності: Один із базових інструментів системи, який дозволяє точно визначати кількість спожитих калорій. Це відіграє ключову роль у створенні ефективного раціону, плануванні харчування та досягненні контрольованої втрати або набору ваги;
- розрахунок макроелементів: Ця можливість надає детальні відомості щодо співвідношення білків, жирів і

вуглеводів у щоденному меню користувача. Така інформація є необхідною для формування збалансованого раціону відповідно до фітнес-цілей або медичних рекомендацій;

– інформаційна таблиця харчових продуктів: Наявність повної бази даних із поживними властивостями продуктів допомагає користувачам усвідомлено підходити до вибору їжі та самостійно складати оптимальний раціон;
– можливість пошуку продуктів: Швидкий пошук за назвою чи категорією дає змогу миттєво знаходити необхідну інформацію про харчову цінність продуктів, що значно спрощує ведення харчового щоденника.

Обрані критерії відображають найважливіші характеристики, які користувачі шукають у веб-застосунках для контролю харчування. Вони охоплюють як основні функціональні можливості, так і користувацький досвід, і є вирішальними для визначення ефективності та популярності таких застосунків.

Таблиця 1.1 – Порівняння з аналогами

Функціонал	FatSecret	Калькулятор калорій	kyiv-dpss.gov	MuscleWiki	Freedieting	HealthyTracker
Зручний інтерфейс	-	-	-	+	-	+
Калькулятор калорій	-	+	+	+	+	+
Калькулятор макроелементів	-	+	-	+	+	+

Таблиця харчування	+	+	-	-	-	+
Пошук продуктів	+	+	-	-	-	+
Безкоштовний продукт	-	-	+	-	+	+

1.3.2 Аналіз відомих алгоритмічних та технічних рішень

У відповідності до загального опису та поставлених вимог до програмного забезпечення, що буде створене, система повинна мати користувацький інтерфейс, web-сервіс, що буде забезпечувати обробку запитів та сервіс, що відповідатиме за опрацювання і взаємодії з ШІ.

1.3.2.1. Архітектура

Одним із ключових аспектів програмного рішення є питання вибору архітектури. Оскільки виходячи з тенденцій використання програмних рішень в сучасному світі та як наслідок з переваг та недоліків існуючих програмних рішень був зроблений вибір у напрямку використання web-застосунку, як підходу до реалізації поставлених задач дипломної роботи, важливо розглянути доступні архітектурні підходи.

Існує декілька основних концепцій реалізації web-застосунків з використанням різних архітектурних рішень, мікросервісна, монолітна, SOA[6]. Їх популярність станом на поточний момент часу продемонстровано на рис. 1.6, дані для даної діаграми взято на основі аналізу джерел AWS microservices[8], Service Oriented Architecture[9], Microservices vs Monolithic Architecture: A Comparison to Guide Your 2025 Strategy[10], 90+ Cloud Computing Statistics: A 2025 Market Snapshot [11], The Future of Service-Oriented Architecture - Top Trends & Predictions for Software Architects[12].

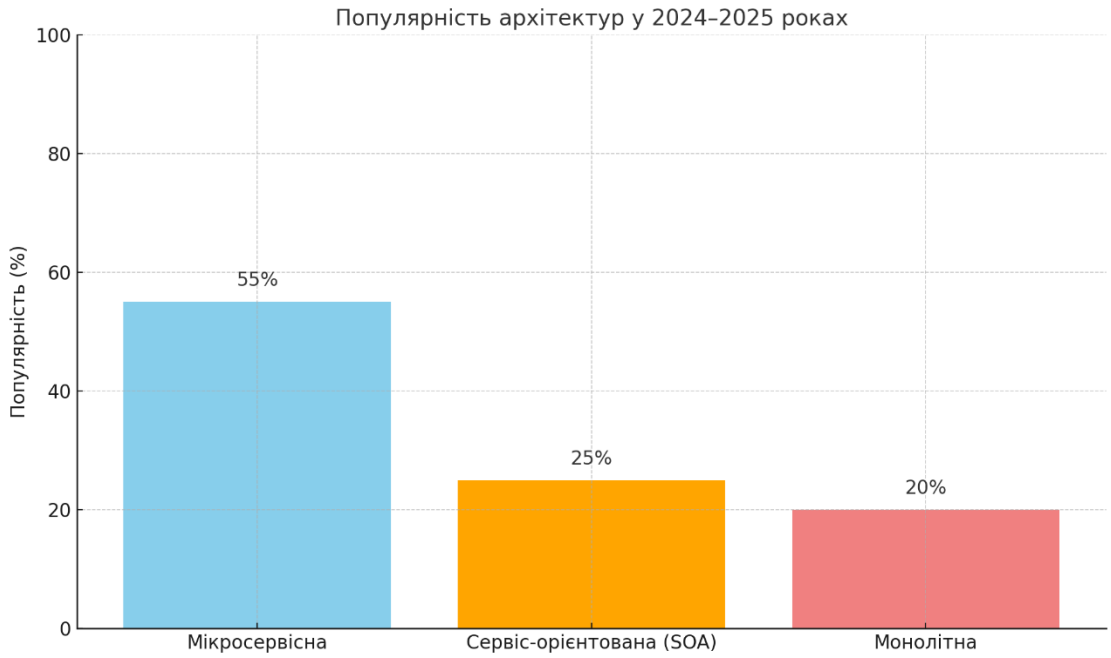


Рисунок 1.6 – Популярність архітектурних рішень

Перший і певно один із найпопулярніших Microservices Architecture Market Report by Component [13] – мікросервісна архітектура. Популярність даного підходу обумовлена наступними критеріями:

– масштабованість досягається можливістю незалежно від інших сервіс запустити інший інстанс поточного сервісу у випадку його навантаженості;
– незалежність розробки сервісів та їх розгортання в межах комерційних рішень, важлдиво мати можливість залучити багато людей та в паралельно вести розробку різних модулів, дана архітектура дозволяє це робити;
– покращена надійність, яка досягається масштабованістю і можливістю автономно запускати той чи інший сервіс гарантує, що відмова одного сервісу не обов'язково призводить до падіння всієї системи;
– гнучкість у виборі технологій розробки, оскільки сервіси взаємодіють за тим чи іншим контрактом, розробник має

можливість обрати найкращу для даної задачі технологію та реалізувати за допомогою неї обраний сервіс;

- спрощення процесу тестування та відлагодження - через використання SRP принципу Microservices and the Single Responsibility Principle (SRP) — Simplified [14] обсяг кодової бази кожного окремо взятого сервісу є значно меншим ніж при монолітних чи SOA[7] рішеннях, що в свою чергу спрощує обсяг коду що потребує відповідного покриття unit тестами [15] та спрощує використання об’єктів підставок;
- CI/CD підтримка значно краща, оскільки кожен сервіс може бути розміщений окремо, містити свою логіку розгортання та має слабкий зв'язок з іншими модулями/сервісами.

Проте в даного підходу є суттєві недоліки. Основними з яких є:

- складність інфраструктури, що обумовлена контейнирезацією та оркестрацією сервісів, а також вирішення задачі балансування навантаження;
- між сервісна взаємодія, оскільки кожен сервіс взаємодіє з іншими за контрактом в межах того чи іншого способу комунікації в межах обраних протоколів для її побудови, а отже зміна одного контракту потребує зміни у всіх сервісах, що використовують даний контракт;
- зростання затримки через мережеві виклики, оскільки система для реалізації обробки запиту потребує виклику ряду сервісів у відповідності до реалізації та використання SRP принципу, а кожен виклик іншого сервісу потребує мережевого виклику, час на виконання запиту збільшується;
- тестування, попри спрощене Unit тестування, задача реалізації інтеграційного тестування стає кратно більшою, що обумовлене необхідністю тестування контрактів взаємодії;

- збільшення витрат на підтримку, мікросервісна архітектура передбачає окреме розміщення наявних сервісів, що в свою чергу призводить до росту вартості підтримки інфраструктури.

Іншим архітектурним підходом є SOA[7]. Даний підхід є певним компромісним рішенням між мікросервісним та монолітним архітектурним підходом. Це досягається шляхом покладання більшої відповідальності на сервіс і зменшенням їх кількості. Даний підхід передбачає виділення в окремий сервіс великих модулів програмного рішення. Як наслідок він невелює частину недоліків мікросервісного підходу та монолітного, проте і агрегує частину з них.

Його перевагами є:

- модульність та розділення обов'язків, що досягається тим, що кожен сервіс відповідає за окрему функцію. Це підвищує керованість та спрощує підтримку;
- масштабованість окремих сервісів як і в мікросервісному архітектурному підході у випадку SOA, сервіси можна масштабувати незалежно, що корисно при змінному навантаженні;
- підтримка оркестрації;
- мовна й технологічна незалежність, кожен сервіс може бути реалізований на іншій технології.

Недоліки:

- витрати на інфраструктуру, вони менші ніж при використанні мікросервісного підходу, але більші ніж при використанні монолітного, що обумовлено кількістю окремих сервісів;
- складність тестування інтеграцій, аналогічно як і в попередньому пункті, є меншою ніж в мікросервісах, проте більшою ніж в монолітному рішенні, а складність обумовлена кількістю сервісів.

Монолітна архітектура, самий простий у реалізації підхід що передбачає виключно один сервіс, якій в свою чергу містить всю необхідну для системи поведінку.

Перевагами цього підходу є:

- простота розробки та розгортання;
- просте локальне середовище розробки;
- пряма інтеграція між модулями;
- менші витрати на інфраструктуру.

Недоліками даного підходу є:

- обмежена масштабованість;
- низька гнучкість при внесенні змін;
- обмежений вибір технологій.

1.3.2.2. Джерело даних (БД)

У процесі розробки програмного забезпечення постає необхідність вибору відповідної системи управління базами даних (СУБД). Існує два основні підходи: використання реляційних баз даних (SQL) або нереляційних (NoSQL) рішень.

У даному випадку було прийнято рішення на користь реляційної бази даних Microsoft SQL Server (MS SQL). Такий вибір зумовлений вимогами до чіткої структури даних, суворої логіки взаємозв’язків між сутностями та необхідністю підтримки транзакцій.

Переваги реляційної бази даних MS SQL:

- строга схема з фіксованою структурою – всі таблиці мають наперед визначені типи даних, первинні та зовнішні ключі, індекси й обмеження. Це дозволяє контролювати цілісність і валідність даних;

- підтримка ACID-властивостей – MS SQL гарантує надійність транзакцій, забезпечуючи узгодженість даних навіть у випадках відмови;
- інтеграція з інструментами Microsoft – MS SQL легко інтегрується з Visual Studio, Azure та іншими сервісами екосистеми Microsoft, що значно спрощує процес розробки та розгортання.

Основні недоліки реляційних баз даних:

- обмежена горизонтальна масштабованість – на відміну від деяких NoSQL рішень, MS SQL складніше масштабувати на декілька вузлів без використання додаткових механізмів;
- складність модифікації структури – будь-які суттєві зміни в схемі БД вимагають ретельного планування міграцій, що особливо важливо в продуктивному середовищі..

Хоча NoSQL рішення можуть бути привабливими через гнучку структуру зберігання даних і хорошу масштабованість, вони мають низку недоліків:

- відсутність повної транзакційної підтримки, заміненою моделлю eventual consistency;
- складність у роботі з сильно зв’язаними даними – що робить їх менш ефективними для задач, де важлива чітка модель зв’язків, як у випадку з харчовим трекером.

1.3.2.3. Між сервісна взаємодія

Для реалізації рівню комунікації, тобто між сервісної взаємодії існує ряд підходів. Серед них message broker Http комунікація та SOAP протокол взаємодії.

Message broker універсальне рішення для побудови івент орієнтованої архітектури в межах мікросервісного архітектурного підходу, з ключових

мінусів даного підходу є необхідність підтримки шини даних, реєстрація всіх тем та черг повідомлень.

Між сервісна комунікація на основі http протоколу, є значно простішою і базується на викликах API іншого сервісу шляхом http запиту. Основним недоліком даного підходу є втрата неуспішних або необроблених запитів.

SOAP протокол немає сенсу навіть розглядати оскільки він базується на серіалізації XML та є застарілим.

1.4 Аналіз та моделювання бізнес-процесів

Використаємо BPMN[15] діаграму для опису бізнес-процесу. BPMN — цестандарт для моделювання бізнес-процесів, який надає уніфікований мовний засіб для спільного розуміння, аналізу та оптимізації бізнес-процесів всередині організації. Використовуючи символи та правила, BPMN дозволяє створювати графічні представлення процесів, що полегшує співпрацю між бізнес-аналітиками та розробниками програмного забезпечення.

Використання Бізнес-процесної мережі (BPNM) дозволяє досягти стандартизації в моделюванні бізнес-процесів. Цей міжнародний стандарт надає єдиний набір термінів та концепцій, що сприяє узгодженій роботі різних команд та організацій.

Використання BPNM[26] спрощує керування проектами, надаючи можливість створювати, редагувати та вдосконалювати бізнес-процеси в єдиному середовищі. Цесприяє ефективній співпраці команд та управлінням змінами в організації.

Перевірка дотримання норми харчування є одним цільним бізнес процесом, який описується на рисунку 1.6 за допомогою BPMN.

Опис послідовності перевірки дотримання норми харчування:

- користувач входить в систему;

- користувач зберігає персональну норму харчування;
- клієнт застосунку надсилає запит на створення норми харчування;
- сервер застосунку отримує запит на збереження;
- сервер завершує створення норми харчування;
- користувач обирає дату та додає спожиті продукти в раціон;
- клієнт застосунку надсилає запит на оновлення раціону;
- сервер надсилає запит на API з продуктами;
- API з продуктами повертає продукт з його харчовою цінністю;
- сервер обролює цю інформацію та зберігає її;
- користувач перевіряє в застосунку чи його норма виконана;
- клієнт застосунку надсилає запит на перевірку норми харчування;
- сервер застосунку отримує запит на перевірку виконання норми;
- якщо норма не виконана, сервер надсилає відповідне повідомлення зінформацією які макроелементи або кількість калорій треба спожитти;
- якщо норма виконана, сервер надсилає відповідне повідомлення тапозначає в базі даних виконання норми що до цього дня;
- клієнт застосунку відпображає повідомлення про виконання норми.

Застосування нотації BPMN для моделювання віртуальної файлової системи дає змогу чітко структурувати її основні етапи та визначити взаємозв’язки між окремими компонентами. Такий підхід спрощує процес аналізу й подальшої оптимізації, а також покращує комунікацію між усіма учасниками проєкту. Побудована модель дозволяє заздалегідь виявити

потенційні уразливості й внести поліпшення в архітектуру, що у підсумку сприяє розробці стабільного та ефективного програмного рішення.

Для моделювання бізнес-процесу використовується BPMN модель (рисунок 1.7) (рисунок слугує ілюстрацією BPMN моделі).

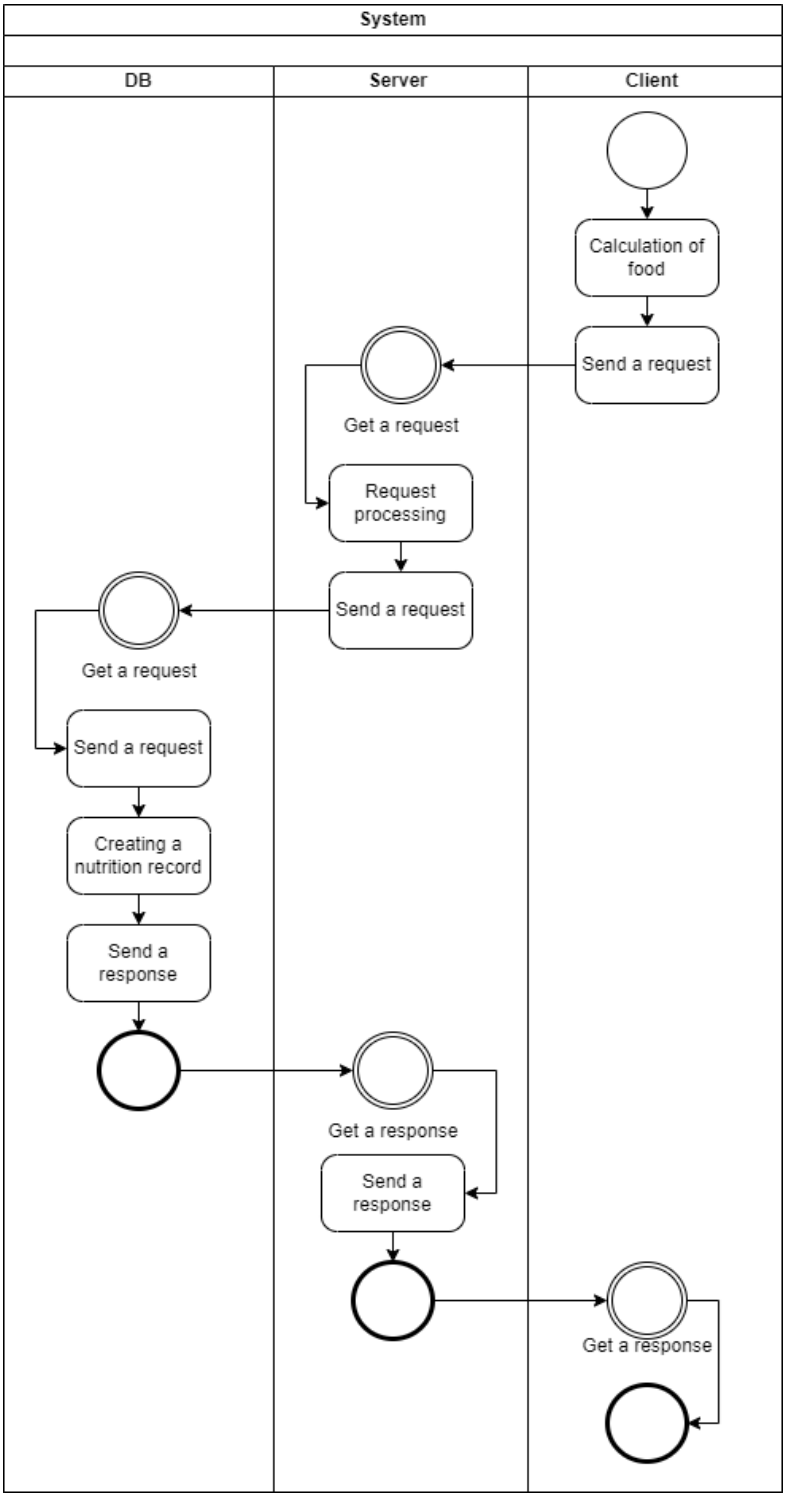


Рисунок 1.7 – Модель бізнес-процесу

Опис моделі бізнес-процесу створення облікового запису користувача:

– користувач переходить на сторінку реєстрації;

– користувач заповнює поля реєстрації;

– якщо введені поля не відповідають шаблону заповнення на клієнтській стороні, відповідні поля виділяються як заповнені некоректно.

Висновки до розділу

У рамках цього розділу було здійснено детальний аналіз предметної галузі, визначено основні вимоги до систем, що вирішують подібні задачі, розглянуто технічні підходи та проаналізовано бізнес-процеси, характерні для даного напряму.

У ході аналізу наявних програмних рішень було оцінено, наскільки функціональні можливості популярних систем відповідають сформульованим критеріям. Проведено порівняння, яке дозволило виокремити сильні та слабкі сторони існуючих платформ, а також виявити нестачу механізмів персоналізації дієти залежно від індивідуальних завдань користувача.

Також було проведено аналіз технічних реалізацій, у межах якого розглянуто ключові архітектурні концепції, переваги й обмеження різних типів архітектур для веб-застосунків, а також особливості джерел даних і протоколів обміну інформацією.

На завершення виконано моделювання основних бізнес-процесів, результати якого представлені у вигляді BPMN[26]-діаграми.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕРЕННЯ

Основуючись на проведеному аналізі існуючих підходів, сформулюємо вимоги до розроблюваного програмного забезпечення.

2.1 Варіанти використання програмного забезпечення

Програмне забезпечення, яке розробляється, орієнтоване на створення ефективного інструменту для розрахунку та контролю харчування, яке надає інтуїтивно зрозумілий та гнучкий інтерфейс для роботи із ним.

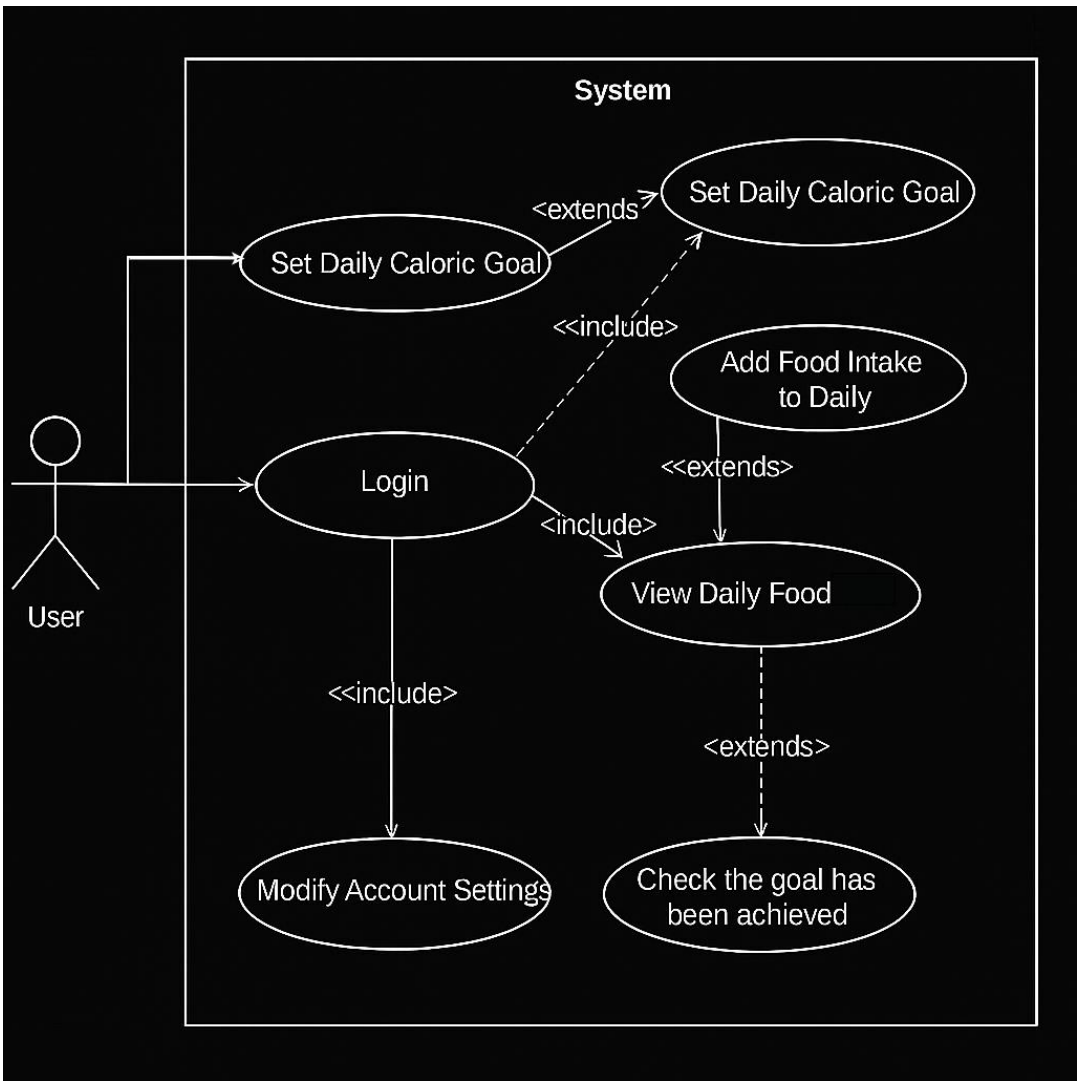


Рисунок 2.1 – Діаграма варіантів використання

В таблицях 2.1-2.10 наведено опис варіантів використання програмного забезпечення.

Таблиця 2.1 – Варіант використання UC-01

Use case name	Обчислення добової потреби в харчуванні
Use case ID	UC-01
Goals	Визначення щоденної норми споживання їжі
Actors	Користувач
Trigger	Користувач прагне дізнатись свою добову потребу в калоріях
Pre-conditions	-
Flow of Events	Користувач переходить на веб-застосунок, після чого заповнює необхідні поля з особистими параметрами для здійснення розрахунку калорійності.
Extension	-
Post-Condition	Система обчислює добову норму харчування на основі введеної користувачем інформації.

Таблиця 2.2 – Варіант використання UC-02

Use case name	Створення облікового запису користувачем
Use case ID	UC-02
Goals	Реєстрація нового користувача у системі
Actors	Незареєстрований користувач
Trigger	Користувач вирішує створити новий обліковий запис на платформі
Pre-conditions	-

Продовження таблиці 2.2

Flow of Events	Користувач заходить на сайт і натискає кнопку «Створити акаунт». Система відображає реєстраційну форму. Користувач заповнює необхідні поля, зокрема ім’я, електронну адресу, пароль тощо. Після цього система перевіряє коректність введених даних і перевіряє, чи є вказана пошта вже зареєстрованою.
Use case name	Реєстрація користувача
Extension	Якщо користувач вводить некоректні або неповні дані, система повідомляє про помилку та просить виправити введену інформацію.
Post-Condition	Після успішної реєстрації створюється обліковий запис користувача, система автоматично перенаправляє його на головну сторінку.

Таблиця 2.3 – Варіант використання UC-03

Use case name	Авторизація користувача в системі
Use case ID	UC-03
Goals	Надання зареєстрованому користувачу доступу до системи
Actors	Користувач із зареєстрованим обліковим записом
Trigger	Користувач хоче виконати вхід до свого акаунта
Pre-conditions	Користувач ще не авторизований у системі
Flow of Events	Користувач відкриває сторінку авторизації. У відповідні поля вводить електронну адресу та пароль. Після заповнення натискає кнопку входу. У разі успішної перевірки облікових даних система надає доступ і перенаправляє користувача далі.
Extension	-

Таблиця 2.4 – Варіант використання UC-04

Post-Condition	Користувач успішно входить до системи та отримує доступ до її функцій
Use case name	Оновлення профільної інформації користувача
Use case ID	UC-04
Goals	Надання можливості зареєстрованому користувачу змінювати дані свого профілю
Actors	Авторизований користувач
Trigger	Користувач хоче внести зміни у власні персональні дані
Pre-conditions	Користувач уже пройшов процедуру входу в систему
Flow of Events	Користувач відкриває розділ "Мій профіль". Для оновлення аватару він може завантажити нове зображення або видалити існуюче. Для зміни імені чи прізвища — редагує відповідні поля у формі. Після цього натискає кнопку збереження.
Extension	У разі некоректного введення даних система виділяє помилкове поле червоним маркуванням і виводить відповідне повідомлення
Post-Condition	Профільні дані користувача успішно оновлено та збережено в системі

Таблиця 2.5 – Варіант використання UC-05

Use case name	Збереження результатів розрахунку добової калорійності
Use case ID	UC-05
Goals	Збереження персоналізованої добової норми калорій для користувача
Actors	Користувач із активним обліковим записом
Trigger	Користувач бажає зафіксувати обчислену норму харчування у системі

Продовження таблиці 2.5

Pre-conditions	Користувач авторизувався в системі
Flow of Events	Користувач відкриває сторінку для обчислення добової калорійності, вводить необхідні параметри (зріст, вага, активність тощо) та натискає кнопку «Зберегти».
Extension	-
Post-Condition	Значення добової норми успішно збережене в акаунті користувача для подальшого використання

Таблиця 2.6 – Варіант використання UC-06

Use case name	Перегляд раціону користувача за вибраний день
Use case ID	UC-06
Goals	Отримати інформацію про список вжитих продуктів за конкретну дату та їхню харчову цінність
Actors	Авторизований користувач
Trigger	Користувач бажає ознайомитись зі своїм харчуванням за певний день
Pre-conditions	Користувач повинен бути авторизованим у системі
Flow of Events	Користувач переходить у розділ з власним харчуванням. Обирає дату, за яку хоче переглянути дані. Система формує список спожитих продуктів із зазначенням їх харчової цінності та показує, чи виконана добова норма, якщо така була попередньо збережена.
Extension	Якщо у вибраний день користувач не додав жодного продукту, система виводить порожній список із відповідним повідомленням.
Post-Condition	Користувач переглянув інформацію про вживані продукти, отримав відповідні харчові показники та статус виконання норми.

Таблиця 2.7 – Варіант використання UC-07

Use case name	Додавання продукту до особистого харчового щоденника
Use case ID	UC-07
Goals	Внести новий спожитий продукт до свого раціону та врахувати його харчову цінність
Actors	Користувач із зареєстрованим обліковим записом
Trigger	Користувач має намір додати продукт, який він спожив, до щоденного списку харчування
Pre-conditions	Користувач має бути авторизованим у системі
Flow of Events	Користувач заходить у розділ харчування, обирає потрібну дату. Далі він вказує прийом їжі (сніданок, обід тощо) і натискає кнопку додавання. У результаті відкривається меню, де користувач вводить назву продукту та його кількість (вагу), після чого підтверджує дію.
Extension	-
Post-Condition	Обраний продукт успішно додано до раціону, і його поживна цінність врахована в загальному денному підсумку.

Таблиця 2.8 – Варіант використання UC-08

Use case name	Перевірка відповідності добового раціону встановленій нормі
Use case ID	UC-08
Goals	Визначити, чи дотримався користувач своєї індивідуальної норми харчування за конкретний день
Actors	Зареєстрований користувач
Trigger	Користувач хоче перевірити, чи виконав денну норму калорій та поживних речовин, визначену раніше

Продовження таблиці 2.7

Pre-conditions	Користувач авторизований у системі та має збережену добову норму харчування
Flow of Events	Користувач відкриває розділ з власним раціоном і вибирає дату, за яку потрібно провести перевірку. Система обчислює загальну кількість спожитих калорій і порівнює її з раніше збереженою нормою.
Extension	Якщо користувач не дотримався норми, система повідомляє про це з відповідною індикацією
Post-Condition	Користувач бачить повідомлення про досягнення або недосягнення добової харчової мети

Таблиця 2.9 – Варіант використання UC-09

Use case name	Оновлення профільного зображення користувача
Use case ID	UC-09
Goals	Надати можливість користувачу змінити своє фото у профілі
Actors	Зареєстрований користувач
Trigger	Користувач хоче оновити своє зображення профілю
Pre-conditions	Користувач має активну сесію в системі
Flow of Events	Користувач переходить до особистого профілю. У відповідному розділі обирає нове зображення або видаляє наявне. Система відображає оновлене фото після збереження змін.
Extension	-
Post-Condition	Профільне зображення користувача успішно оновлено у системі

Таблиця 2.10 – Варіант використання UC-10

Use case name	Відновлення доступу до облікового запису (відновлення пароля)
Use case ID	UC-10

Продовження таблиці 2.10

Goals	Надати користувачу можливість встановити новий пароль у випадку втрати доступу
Actors	Незареєстрований користувач / користувач без активної сесії
Trigger	Користувач хоче відновити доступ до свого акаунта шляхом скидання пароля
Pre-conditions	Користувач має зареєстрований акаунт, але не ввійшов у систему
Flow of Events	На сторінці авторизації користувач натискає опцію «Забули пароль», після чого вводить електронну адресу. Система перевіряє наявність акаунта за вказаною поштою та надсилає лист для скидання пароля.
Extension	Якщо система не знаходить обліковий запис за введеною електронною адресою, виводиться відповідне повідомлення про помилку
Post-Condition	Користувач отримує можливість створити новий пароль і надалі авторизується в системі

2.2 Розроблення функціональних вимог

Функціональні вимоги [16] — це опис конкретних дій або можливостей, які має реалізовувати програмне забезпечення чи система для задоволення потреб користувачів та виконання поставлених завдань. Вони визначають, яку саме функціональність повинна підтримувати система, щоб забезпечити виконання необхідних операцій. Зазвичай такі вимоги формулюються як набір дій, які користувач повинен мати змогу здійснювати під час взаємодії з програмним продуктом.

Таблиця 2.11 – Загальна модель вимог

№	Назва	ID Вимоги	Пріоритети	Ризики
2	Система авторизації користувача	FR-2	Високий	Високий
2.1	Реєстрація користувача	FR-3	Високий	Високий
2.2	Авторизація користувача	FR-4	Високий	Високий
5	Розрахунок добової норми харчування	FR-8	Високий	Середній
3	Відновлення пароля	FR-6	Високий	Середній
2.3	Вихід із акаунту	FR-5	Середній	Низький
4	Редагування профілю	FR-7	Середній	Низький
6	Збереження норми харчування	FR-9	Середній	Низький
7	Перегляд харчування за датою	FR-10	Середній	Низький
8	Додавання продуктів до раціону	FR-11	Середній	Низький
9	Редагування/видалення продуктів раціону	FR-12	Середній	Низький
10	Перевірка виконання норми харчування	FR-13	Середній	Низький
1	Перегляд сторінки привітання	FR-1	Низький	Низький

Таблиця 2.12 – Перелік функціональних вимог

Назва	Опис
FR-1	Реєстрація користувача з унікальним імейлом. Користувач має підтвердити імейлдля реєстрації
FR-2	Вхід в систему за імейлом та паролем
FR-3	Користувач може відновити пароль,запросивши лист на пошту, по посиланню з якого можна встановити новий пароль та увійти в систему
FR-4	Користувач може редагувати ім’я та прізвище, аватар.
FR-5	Користувач може розрахувати свою добову норму харчування за такими параметрами яквік, зріст, вага, відсоток жиру, спосіб життя та персональна ціль.
FR-6	Користувач може зберегти свою добову норму харчування.
FR-7	Користувач може переглядати своє харчування відносно введеної ним дати.
FR-8	Користувач може додавати до свого харчуваня вжиті продукти.
FR-9	Якщо користувач має продукт у раціоні, то він може редагувати його вагу або ж видалитиз раціону.
FR-10	Якщо у користувача визначена норма харчування, то він може перевірити чи вонавиконана

	UC-01	UC-02	UC-03	UC-04	UC-05	UC-06	UC-07	UC-08	UC-09	UC-10
FR-1		+								
FR-2			+							
FR-3										+
FR-4				+					+	
FR-5	+									
FR-6					+					
FR-7						+				
FR-8							+			
FR-9							+			
FR-10								+		

Рисунок 2.2 – Матриця трасування вимог

2.3 Розроблення нефункціональних вимог

Нефункціональні вимоги описують загальні властивості програмного забезпечення або системи, які не пов’язані безпосередньо з виконанням конкретних функцій, але характеризують її якість та поведінку. Вони охоплюють такі параметри, як продуктивність, стабільність, рівень захищеності та інші показники, що визначають ефективність і надійність роботи системи в цілому.

Таблиця 2.12 – Таблиця не функціональних вимог

Номер	Назва	Опис
NRF-1	Надійність	Система повинна функціонувати стабільно та коректно, забезпечуючи стійкість до помилок. У разі виникнення збоїв, користувач має отримувати чіткі й інформативні повідомлення для розуміння ситуації.

Продовження таблиці 2.12

NRF-2	Безпека	Платформа зобов’язана гарантувати збереження особистої інформації та захист від несанкціонованого доступу під час будь-яких операцій. Це включає шифрування даних і дотримання стандартів безпеки.
NRF-3	Зручність інтерфейсу	Інтерфейс користувача має бути інтуїтивно зрозумілим, логічно структурованим та простим у взаємодії навіть для нових користувачів.

Продовження таблиці 2.12

NRF-4	Адаптація	Застосунок повинен коректно функціонувати у середовищі браузера Google Chrome версії 120.0.6099.130, забезпечуючи повну підтримку його можливостей.

Наведені в таблиці вимоги допомагають враховувати ключові аспекти, які забезпечують не тільки функціональну повноту, але й задовольняють якість програмного продукту.

2.4 Аналіз системних вимог

Мінімальними системними вимогами для роботи Web-сервісу є:

- програмне забезпечення повинно функціонувати на IB M-сумісних персональних комп‘ютерах;
- тип процесору Intel Pentium 4 / AMD Athlon 64 або пізнішої версії з підтримкою SSE2;
- об’єм ОПЗ 4 ГБ;
- підключення до мережі Інтернет зі швидкістю від 2 мегабіт.

Дані системні вимоги ґрунтуються на мінімальних системних вимогах до браузерів на яких повинен працювати Web-сервіс.

2.5 Аналіз економічних показників програмного забезпечення

Для аналізу економічної доцільності створення програмного забезпечення доцільно застосувати модель оцінювання COCOMO (Constructive Cost Model), яка дозволяє кількісно оцінити ключові характеристики проєкту: трудомісткість, тривалість розробки, необхідну кількість фахівців та приблизну вартість реалізації.

У межах базової моделі COCOMO I розрахунок трудомісткості здійснюється за формулою:

$$Effort = a \cdot KLOC^b$$

Де, Effort – обсяг трудовитрат у людино-місяцях, KLOC – обсяг коду в тисячах рядків, a і b – емпіричні коефіцієнти, які залежать від типу проєкту.

Оскільки розроблюване програмне забезпечення належить до категорії проєктів середньої складності, відповідні коефіцієнти дорівнюють: a = 3.0, b = 1.12. Припустимо, що обсяг коду становить 10 KLOC (10 000 рядків коду).

Підставляємо значення у формулу, та отримуємо результат 39.5. Це означає, що загальний обсяг трудових витрат на розробку становить приблизно 39,5 людино-місяців.

Далі розрахуємо орієнтовну тривалість розробки, використовуючи формулу:

$$Time = c \cdot Effort^d$$

Де, Time – загальна тривалість проєкту в місяцях, c і d – емпіричні коефіцієнти, що визначаються типом проєкту.

Для проєктів середньої складності значення коефіцієнтів: c = 2.5, d = 0.35. Підставляючи попередньо розраховану трудомісткість.

Отже, орієнтовна тривалість розробки програмного забезпечення становить трохи більше 9 місяців.

Для визначення кількості розробників, необхідних для виконання проєкту у зазначені строки, скористаємося формулою:

$$People = Efort/Time$$

Таким чином, для реалізації цього програмного рішення протягом 9 місяців потрібно залучити команду з 5 спеціалістів.

Зазначені обчислення ґрунтуються на класичній моделі COCOMO та мають теоретичний характер. Вони не враховують сучасні практики повторного використання коду, застосування готових бібліотек, фреймворків або засобів автоматизації, які можуть значно знизити фактичні трудозатрати та вартість розробки.

2.6 Постановка завдання на розробку програмного забезпечення

У межах розробки веб-застосунку для моніторингу та обчислення параметрів харчування визначено низку функціональних задач, необхідних для реалізації основного функціоналу системи. Зокрема, потрібно забезпечити можливість реєстрації нових користувачів із підтвердженням електронної пошти та реалізувати безпечну авторизацію за допомогою email і пароля. Також необхідно передбачити механізм відновлення пароля у разі його втрати.

Користувач повинен мати змогу розраховувати свою добову норму харчування на основі введених фізіологічних параметрів (вік, зріст, вага тощо) і обраного стилю життя, з подальшим збереженням цієї інформації в особистому профілі.

Серед інших важливих функцій — можливість редагування профільних даних, зокрема імені, прізвища та аватару, а також перегляд і аналіз власного харчування за конкретну дату. Користувачі мають змогу додавати до свого раціону продукти, редагувати або видаляти їх, а також перевіряти відповідність фактичного харчування

встановленій добовій нормі.

Усі зазначені задачі повинні бути реалізовані у відповідності до функціональних вимог, що гарантує ефективність та практичну цінність веб-застосунку.

Висновки до розділу

У другому розділі даної дипломної роботи було визначено та сформовано основні вимоги до програмного забезпечення, призначеного для обчислення та моніторингу показників здорового харчування.

Зокрема, було виконано наступні кроки: побудовано діаграму варіантів використання, яка демонструє сценарії взаємодії користувача із системою для розрахунку та трекінгу харчування; сформульовано перелік функціональних вимог до розроблюваного програмного продукту; визначено нефункціональні вимоги, що охоплюють якісні характеристики системи.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Архітектурною основою веб-застосунку є багаторівнева модель n-tier[19], яка є розвитком традиційної клієнт-серверної архітектури. Її ключова особливість полягає в чіткому розмежуванні функціональних шарів: інтерфейс користувача, бізнес-логіка, доступ до даних і їх зберігання. Така структура забезпечує високу гнучкість та спрощує масштабування системи.

Завдяки n-tier архітектурі, розробники можуть модифікувати або оновлювати окремі компоненти застосунку незалежно один від одного, не впливаючи на загальну структуру. Це рішення є особливо ефективним у випадках, коли система обробляє велику кількість користувачів або працює з великими масивами даних.

Серверний рівень веб-застосунку може включати такі модулі:

- API шлюз, який забезпечує точку входу для клієнтських запитів;
- бізнес-логіка, що обробляє бізнес-правила та операції;
- модуль доступу до даних, який відповідає за взаємодію з базою даних таіншими зовнішніми джерелами.

Клієнтська частина реалізована у вигляді застосунку, створеного з використанням бібліотеки React. Архітектура фронтенду побудована за компонентним підходом, що передбачає поділ інтерфейсу на незалежні модулі — кожен компонент може бути повторно використаний у різних частинах системи.

Серверна логіка розроблена відповідно до принципів REST-архітектури, що забезпечує зрозумілу структуру запитів та спрощує інтеграцію з клієнтською частиною. Для збереження сесії та реалізації

механізму автентифікації було застосовано JWT (JSON Web Token) [18], що дає змогу безпечно передавати дані про авторизованого користувача

Розглянемо схему архітектури:

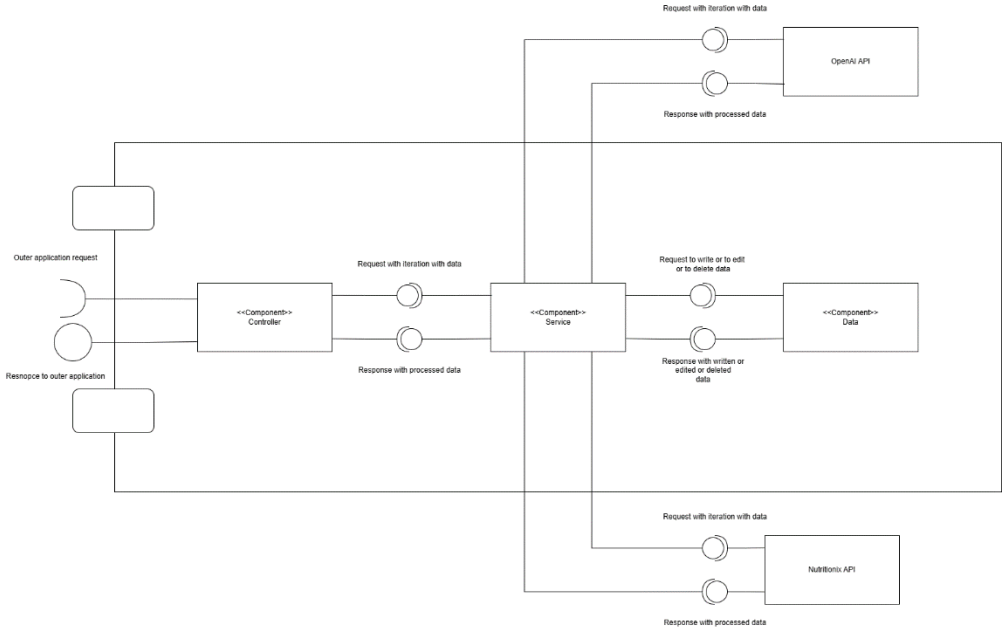


Рисунок 3.1 – Діаграма компонентів веб-застосунку

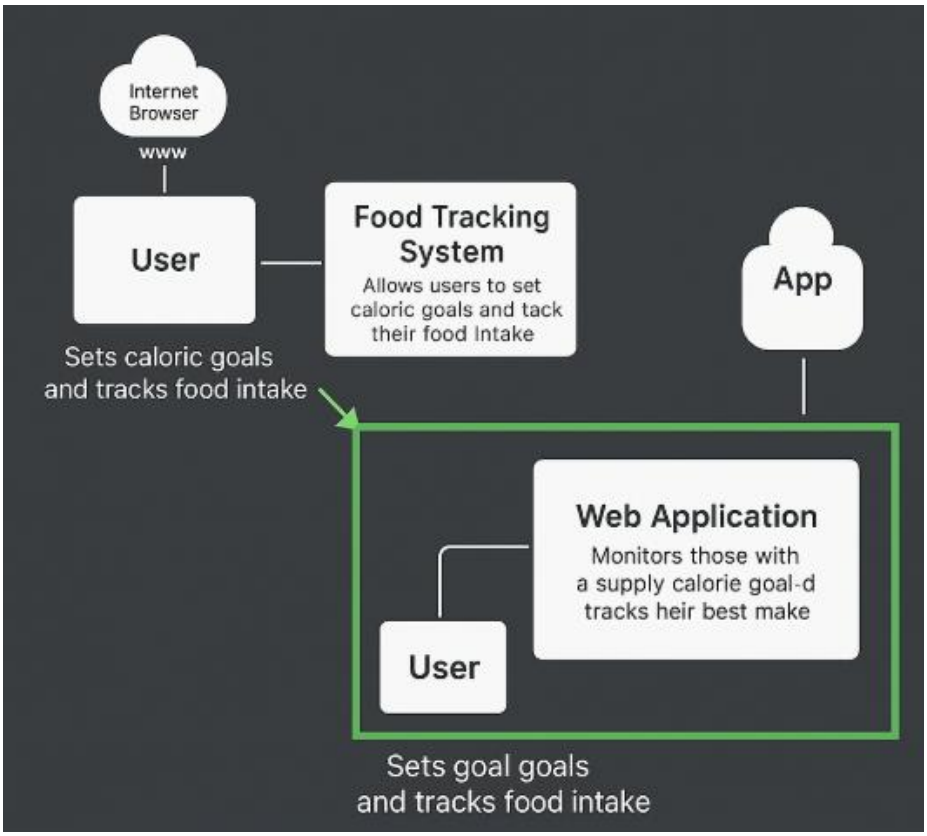


Рисунок 3.2 – Діаграма C4 верхніх рівнів

3.2 Архітектурні рішення та обґрунтування вибору засобів розробки

У рамках побудови архітектури програмного забезпечення було обрано монолітну багаторівневу структуру, виходячи з критерію швидкості впровадження рішення, а також враховуючи нефункціональні вимоги щодо навантаження на систему на початкових етапах розробки. Механізм авторизації реалізовано у самому програмному рішенні із застосуванням стандарту JWT (JSON Web Token)[18], що забезпечує надійну автентифікацію користувачів.

Після аналізу типів баз даних та з урахуванням структури інформації, яка буде зберігатися, було прийнято рішення використовувати реляційну базу даних, як найбільш відповідну до вимог проєкт.

Взаємодія із зовнішнім сервісом для формування рекомендацій реалізована за допомогою HTTPS-протоколу, що дозволяє зменшити складність підтримки інфраструктури. Зазначений сервіс також містить виклики до нейронної мережі, які так само здійснюються через захищене з’єднання.

Для розробки серверної частини застосунку було обрано технологію .NET у поєднанні з мовою програмування C#, тоді як фронтенд реалізовано за допомогою фреймворку React.js.

3.3 Конструювання програмного забезпечення

Основою для розробки програмного забезпечення є алгоритм розрахунку та аналізу харчової цінності продуктів, який реалізований за допомогою .NET та MS SQL Server. Дані про споживання їжі надходять через веб-інтерфейс, створений на React, обробляються серверною частиною на .NET та зберігаються в реляційній базі даних MS SQL. Візуалізація та управління даними здійснюються через інтерактивний інтерфейс на React.

Для розрахунку харчової цінності використовується підхід, заснований на попередньо сформованій базі продуктів із вказаними поживними характеристиками (калорії, білки, жири, вуглеводи тощо). Модель розрахунку працює в реальному часі, автоматично оновлюючи статистику споживання.

Алгоритм обробки даних включає наступні етапи:

- отримання вхідних даних через веб-інтерфейс (React[23]);
- передача даних на серверну частину (.NET[21]) для валідації та обробки;
- розрахунок поживної цінності на основі даних з MS SQL;
- агрегація та аналіз отриманої інформації (добові, тижневі, місячні норми);
- збереження результатів у базі даних та подальша візуалізація в інтерфейсі.

Розроблюване програмне забезпечення працює з структурованими даними, які зберігаються в реляційній базі MS SQL[20], а також передаються між клієнтською та серверною частинами у форматі JSON.

3.3.1 Опис структури бази даних

В якості системи управління базами даних використовується SQL Server. База даних серверу призначеняя для системи контролю норми харчування, що включає інформацію про користувачів, їхні норми харчування, прийоми їжі, продукти та інші пов'язані дані. Таблиця AspNetUsers є базовою для користувача, вона є частиною класу IdentityUser та зберігає дані користувачів, такі як ім'я, електронна пошта, парольта іншу інформацію. Таблиця AspNetUserClaims зберігає клейми для користувачів. Таблиця AspNetUserLogins зберігає дані для входу користувачів. Таблиця AspNetUserRoles зв'язок між користувачами і ролями. Таблиця AspNetUserTokens зберігає токени для користувачів. Таблиця CalorieGoal зберігає норми калорій, білків,жирів та вуглеводів

для користувачів, адже у кожного користувача індивідуальний набір параметрів харчування. Таблиця CaloriesConsumed зберігає інформацію про прийоми їжі, вказуючи їхній тип та пов'язаний день. Таблиця Food берігає дані про їжу, включаючи кількість калорій, білків, жирів та інших харчових компонентів, це потрібно для того щоб користувач міг відслідкувати які саме продукти він спожив у конкретний день. Таблиця UserRegistrations Зберігає дані про реєстрації користувачів,такі як URL та час створення.Опис таблиць бази даних наведено у таблицях 3.1 - 3.12. Модель бази даних наведена на рисунку 3.3

Таблиця 3.1 – Опис таблиці AspNetRoles

Назва поля	Тип даних	Опис поля
Id	uniqueidentifier	Унікальний ідентифікатор ролі (первинний ключ)
Name	nvarchar	Назва ролі у системі
NormalizedName	nvarchar	Приведене до стандартного формату ім’я ролі (для пошуку та порівняння)
ConcurrencyStamp	nvarchar	Токен для забезпечення цілісності при паралельному оновленні записів

Таблиця 3.2 – Опис таблиці AspNetRoleClaims

Назва поля	Тип даних	Опис поля
Id	int identity	Унікальний ідентифікатор запису (первинний ключ)
RoleId	uniqueidentifier	Посилання на роль — зовнішній ключ до таблиці ролей
ClaimType	nvarchar	Тип клейма, що визначає категорію права або атрибута
ClaimValue	nvarchar	Конкретне значення, яке відповідає певному типу клейма
Назва поля	Тип даних	Опис поля

Таблиця 3.3 – Опис таблиці AspNetUsers

Назва поля	Тип даних	Опис поля
Id	uniqueidentifier	Унікальний ідентифікатор користувача (первинний ключ)
DisplayName	nvarchar	Відображуване ім’я користувача
RefreshToken	nvarchar	Токен, що використовується для оновлення сесії
RefreshTokenExpiryTime	datetimeoffset	Дата та час закінчення дії токена оновлення
Avatar	nvarchar	URL або шлях до аватарки користувача
UserName	nvarchar	Ім’я користувача (логін)
NormalizedUserName	nvarchar	Приведене до стандартного вигляду ім’я користувача (логін)
EmailConfirmed	bit	Статус підтвердження електронної адреси
PasswordHash	nvarchar	Хеш значення пароля користувача
SecurityStamp	nvarchar	Унікальна мітка безпеки, що змінюється при зміні критичних даних
ConcurrencyStamp	nvarchar	Мітка для контролю паралельних змін у записі

Продовження таблиці 3.3

PhoneNumber	nvarchar	Номер телефону користувача
PhoneNumberConfirmed	bit	Ознака підтвердження номера телефону
TwoFactorEnabled	bit	Вказує, чи увімкнено двофакторну автентифікацію
LockoutEnd	datetimeoffset	Дата та час завершення блокування акаунта
LockoutEnabled	bit	Ознака, чи дозволено блокування облікового запису
AccessFailedCount	int	Кількість невдалих спроб входу
FirstName	nvarchar	Ім’я користувача
LastName	nvarchar	Прізвище користувача

Таблиця 3.4 – Опис таблиці AspNetUserClaims

Назва поля	Тип даних	Опис поля
Id	int	Унікальний ідентифікатор запису (первинний ключ)
UserId	uniqueidentifier	Унікальний ідентифікатор користувача, що пов’язує право з обліковим записом
ClaimType	nvarchar	Категорія права (тип претензії), яка визначає область дії
ClaimValue	nvarchar	Значення відповідного права або атрибуту, що належить користувачеві

Таблиця 3.5 – Опис таблиці AspNetUserLogins

Назва поля	Тип даних	Опис поля
LoginProvider	nvarchar	Назва зовнішнього постачальника автентифікації (наприклад, Google, GitHub)
ProviderKey	nvarchar	Унікальний ідентифікатор користувача в системі зовнішнього постачальника
ProviderDisplayName	nvarchar	Відображуване ім’я провайдера, яке видно в інтерфейсі

Таблиця 3.6 – Опис таблиці AspNetUserRoles

Назва поля	Тип даних	Опис поля
UserId	uniqueidentifier	Унікальний ідентифікатор користувача, який пов'язаний з роллю
RoleId	uniqueidentifier	Ідентифікатор ролі, призначеної відповідному користувачеві

Таблиця 3.7 – Опис таблиці AspNetUserTokens

Назва поля	Тип даних	Опис поля
UserId	uniqueidentifier	Унікальний ідентифікатор користувача, до якого прив'язано токен

Продовження таблиці 3.7

LoginProvider	uniqueidentifier	Унікальний ідентифікатор постачальника автентифікації
Name	nvarchar	Назва токена (тип або категорія, наприклад, "access_token")
Value	nvarchar	Значення токена, яке використовується для підтвердження доступу

Таблиця 3.8 – Опис таблиці CalorieGoal

Назва поля	Тип даних	Опис поля
Id	uniqueidentifier	Унікальний ідентифікатор запису з цільовими показниками (первинний ключ)
UserId	uniqueidentifier	Ідентифікатор користувача, якому належать ці значення
Calories	real	Бажана кількість калорій для споживання протягом доби
Carbs	real	Цільовий обсяг споживання вуглеводів (у грамах)
Fat	real	Запланована кількість жирів у раціоні (у грамах)
Protein	real	Необхідна добова кількість білків (у грамах)

Таблиця 3.9 – Опис таблиці CaloriesConsumed

Назва поля	Тип даних	Опис поля
Id	uniqueidentifier	Унікальний ідентифікатор запису про факт виконання норми
Date	datetime2	Дата, за яку здійснюється перевірка споживання калорій
UserId	uniqueidentifier	Ідентифікатор користувача, до якого належить запис
NormIsFulfilled	bit	Логічне значення: true — якщо добову норму виконано, false — якщо ні

Таблиця 3.10 – Опис таблиці Meal

Назва поля	Тип даних	Опис поля
Id	uniqueidentifier	Унікальний ідентифікатор запису про прийом їжі (первинний ключ)
DailyId	uniqueidentifier	Ідентифікатор добового запису калорій, до якого належить цей прийом їжі
Type	nvarchar	Тип прийому їжі (наприклад: сніданок, обід, вечеря, перекус тощо)

Таблиця 3.11 – Опис таблиці Food

Назва поля	Тип даних	Опис поля
Id	uniqueidentifier	Унікальний ідентифікатор продукту (первинний ключ)
ProductName	nvarchar	Назва продукту
Volume	int	Об’єм або вага продукту в грамах (або мілілітрах — залежно від типу)
Caffeine	real	Вміст кофеїну в продукті (мг)
Calories	real	Кількість калорій
Carbs	real	Вміст вуглеводів (г)
Cellulose	real	Кількість клітковини (г)
Fat	real	Вміст жирів (г)
Protein	real	Вміст білків (г)
Salt	real	Вміст солі (г)
Water	real	Кількість води в складі продукту (мл або г)
MealId	uniqueidentifier	Ідентифікатор прийому їжі, до якого належить даний продукт (зовнішній ключ)

Таблиця 3.12 – Опис таблиці UserRegistrations

Назва поля	Тип даних	Опис поля
Id	uniqueidentifier	Унікальний ідентифікатор запису реєстрації
Url	nvarchar	Посилання, за яким користувач виконує реєстрацію або підтвердження
CreatedAt	datetimeoffset	Дата та час створення реєстраційного посилання
ExpiresAt	datetimeoffset	Дата та час завершення дії посилання
IsUrlRegenerated	bit	Позначка, яка вказує, чи було повторно згенеровано посилання
UserId	uniqueidentifier	Ідентифікатор користувача, якому належить дане реєстраційне посилання

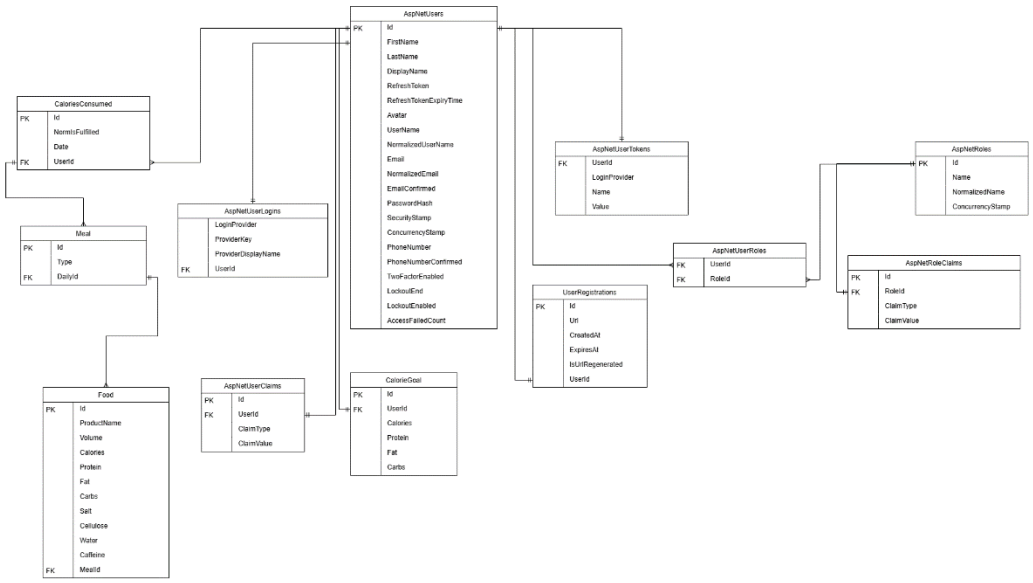


Рисунок 3.3 – ER діаграма сутностей AspNetRoles, AspNetRoleClaims, AspNetUsers, AspNetUserClaims, AspNetUserLogins, AspNetUserRoles, AspNetUserTokens, CalorieGoal, CaloriesConsumed, Meal, Food, UserRegistrations

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.13.

Таблиця 3.13 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Jetbrains Rider	Rider використовувався як основне інтегроване середовище для програмування, призначене для розробки серверної частини веб-застосунку. Цей інструмент має повну сумісність із .NET-екосистемою, підтримує роботу з системами контролю версій і дозволяє виконувати модульне тестування безпосередньо в середовищі розробки.

Продовження таблиці 3.13

2	Postman[37]	Застосунок, що використовувався для симулювання та перевірки RESTful API-запитів. Дозволив проводити тестування обробки HTTP-запитів серверною частиною, а також налагоджувати зв’язок між клієнтським інтерфейсом і сервером.
3	SQL server management studio	Засіб з графічним інтерфейсом, призначений для роботи з базами даних Microsoft SQL Server. Використовувався для керування базами, перегляду вмісту таблиць, написання й запуску SQL-команд, а також для вивчення структури даних, збережених у системі.
4	Visual studio code[28]	Продуктивне та легке середовище розробки, яке застосовувалося для створення фронтенд-частини веб-застосунку. Завдяки численним плагінам VS Code забезпечував зручну роботу з такими технологіями, як HTML, CSS, JavaScript і бібліотека React.

Тексти програмного коду наведені в окремому документі «Текст програми».

3.4 Аналіз безпеки даних

Захист інформації є одним із ключових аспектів у процесі створення програмного забезпечення, особливо коли йдеться про роботу з чутливими даними, такими як особисті відомості, харчові переваги або медичні показники користувачів. У цьому розділі розглядаються основні загрози інформаційній безпеці та аналізуються методи зниження ймовірності виникнення таких ризиків.

Серед найбільш розповсюджених уразливостей у програмних системах можна виділити SQL-ін’єкції, міжсайтове скриптування (XSS), міжсайтову підробку запитів (CSRF) та ненадійні механізми автентифікації.

SQL-ін’єкції виникають у випадках, коли зловмисник отримує можливість виконувати небезпечні SQL-запити через необроблені або ненадійно оброблені вхідні дані. Це може призвести до доступу до критичних даних, їх модифікації або видалення. Щоб уникнути подібних атак, необхідно використовувати параметризовані запити та підготовлені інструкції. У межах даного проєкту застосовується ORM Entity Framework (EF), який за замовчуванням підтримує параметризацію, що значно знижує ризик подібних атак.

XSS-атаки передбачають вставлення шкідливого коду в сторінку, яку потім переглядають інші користувачі. Однак у випадку цього застосунку такі атаки, як і CSRF, не є актуальними, оскільки взаємодія відбувається виключно через захищене API, без безпосереднього рендерингу контенту на стороні браузера.

Ще однією серйозною загрозою є слабка автентифікація, що може дозволити зловмисникам проникнути в систему без дозволу. Неналежна авторизація, у свою чергу, створює ризик ескалації прав доступу, коли користувач отримує доступ до ресурсів, які йому не призначені. Для запобігання таким сценаріям слід застосовувати надійні механізми аутентифікації, наприклад, OAuth 2.0 або JWT (JSON Web Tokens), а також реалізовувати рольову модель доступу (RBAC). Збереження паролів повинно здійснюватися у хешованому вигляді з використанням алгоритмів, таких як bcrypt, що гарантує безпечне зберігання облікових даних.

Таким чином, своєчасне виявлення потенційних вразливостей і впровадження відповідних засобів захисту є критично важливими для забезпечення надійної роботи системи. Це дозволяє створити безпечне програмне середовище, у якому користувачі можуть з упевненістю керувати своїми харчовими даними, не побоюючись за їхню цілісність і конфіденційність.

Висновки до розділу

В розділі "Конструювання програмного забезпечення" для програмного забезпечення, призначеного для розрахунку та моніторингу харчування, детально описано процес побудови та реалізації основних модулів системи з використанням архітектурної моделі N-tier. Такий підхід було обрано через його здатність забезпечити масштабованість, гнучкість та чітке розділення логіки представлення й обробки даних.

Основною технічною особливістю є поділ системи на окремі рівні: інтерфейс користувача (frontend), шар бізнес-логіки, доступ до даних (data access), а також інтеграція з зовнішніми сервісами. Для реалізації інтерфейсу використано сучасну бібліотеку React [6], що дозволила створити зрозумілий та зручний для користувача дизайн. Бізнес-логіка була впроваджена у вигляді сервісів, які забезпечують високу продуктивність та адаптивність до змін у даних користувачів.

У рамках реалізації застосунку було розроблено набір RESTful API, що значно спростив обмін інформацією між клієнтською та серверною частинами. Така структура сприяє легкому оновленню, масштабуванню та підтримці окремих компонентів системи. Розмежування шарів обробки та зберігання даних оптимізувало як процес розробки, так і технічне обслуговування.

Застосування сучасних підходів до розробки забезпечило високу ефективність, стабільність і захищеність системи. У результаті було створено функціональну веб-платформу, яка об’єднує швидку обробку запитів, безпечну роботу з персональними даними та зручний інтерфейс для кінцевого користувача.

Загалом, цей етап роботи завершився створенням масштабованого та надійного рішення, яке не лише відповідає актуальним технічним стандартам, але й забезпечує якісний користувацький досвід та ефективну підтримку процесів планування й контролю харчування.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Метою тестування є наступне:

- перевірка коректності роботи розробленого програмного забезпечення відповідно до функціональних вимог;
- перевірка правильності збереження даних;
- перевірка зручності користувацького інтерфейсу.

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.1-4.7.

Таблиця 4.1 – Тест 1

Тест	Реєстрація нового користувача з підтвердженням електронної пошти
Модуль	Реєстрація користувача
Номер тесту	1.1
Початковий стан системи	Користувач перебуває на сторінці реєстрації
Вхідні данні	Ім’я, прізвище, email-адреса, пароль і його підтвердження

Продовження таблиці 4.1

Опис проведення тесту	У відповідні поля вводяться такі дані: правильні ім’я та прізвище, дійсна електронна пошта, яка раніше не використовувалася в системі, а також надійний пароль, що містить принаймні одну велику літеру, спеціальний символ і цифру. Після надсилання форми користувач отримує на свою пошту лист для підтвердження реєстрації. Перейшовши за наданим посиланням, користувач автоматично перенаправляється на головну сторінку вебзастосунку
Очікуваний результат	Реєстрація завершується успішно — новий користувач додається до бази даних, і система перенаправляє його на головну сторінку.
Фактичний результат	Процес реєстрації виконується без помилок: користувач успішно додається до системи та автоматично переходить на головну сторінку після підтвердження електронної пошти.

Таблиця 4.2 – Тест 2

Тест	Вхід до системи
Модуль	Реєстрація користувача
Номер тесту	1.2

Продовження таблиці 4.2

Початковий стан системи	Користувач перебуває на сторінці авторизації, обліковий запис попередньо створений і підтверджений через електронну пошту
Вхідні данні	Email-адреса, пароль
Опис проведення тесту	У відповідні поля на сторінці входу вводяться правильна електронна адреса, що була використана під час реєстрації, та відповідний пароль. Після підтвердження даних користувач успішно проходить автентифікацію та автоматично перенаправляється до головної сторінки застосунку.
Очікуваний результат	Користувач успішно входить до системи та переходить на головну сторінку без помилок.
Фактичний результат	Процес входу проходить успішно — після введення правильних даних користувач потрапляє на головну сторінку застосунку.

Таблиця 4.3 – Тест 3

Тест	Вхід у систему без підтвердження електронної пошти
Модуль	Реєстрація користувача
Номер тесту	1.3
Початковий стан системи	Користувач перебуває на сторінці авторизації. Обліковий запис створено, але підтвердження електронної пошти ще не здійснено.

Продовження таблиці 4.3

Вхідні данні	Електронна адреса, пароль
Опис проведення тесту	Користувач уводить у відповідні поля електронну пошту, яка була вказана при реєстрації, та правильний пароль. Обліковий запис ще не підтверджено за допомогою листа на email. Після спроби авторизації система блокує вхід та повідомляє про необхідність підтвердження електронної адреси.
Очікуваний результат	Користувач не отримує доступ до системи. Відображається повідомлення з інструкцією щодо активації облікового запису через підтвердження email.
Фактичний результат	Авторизація не відбувається. З’являється повідомлення про потребу підтвердити електронну пошту перед входом.

Таблиця 4.4 – Тест 4

Тест	Відновлення забутого пароля
Модуль	Реєстрація користувача
Номер тесту	1.4
Початковий стан системи	Користувач перебуває на сторінці авторизації. Обліковий запис уже створено, але користувач не пам’ятає свій пароль.
Вхідні данні	Електронна адреса, новий пароль

Продовження таблиці 4.4

Опис проведення тесту	Користувач натискає на посилання для відновлення пароля. Далі переходить на відповідну сторінку, вводить свою зареєстровану електронну адресу та ініціює надсилання листа з посиланням для зміни пароля. Отримавши лист, користувач переходить за посиланням, вводить новий пароль двічі для підтвердження й підтверджує зміну.
Очікуваний результат	Старий пароль замінено новим. Облікові дані оновлено успішно.
Фактичний результат	Пароль змінено — процес відновлення виконано без помилок.

Таблиця 4.5 – Тест 5

Тест	Розрахунок індивідуальної добової потреби в харчуванні
Модуль	Харчування
Номер тесту	1.5
Початковий стан системи	Користувач перебуває на сторінці, призначеній для розрахунку добової норми споживання поживних речовин.

Продовження таблиці 4.5

Вхідні данні	Параметри користувача: зріст, вага, вік, відсоток жирової маси, рівень фізичної активності, індивідуальна мета (наприклад, схуднення, підтримка форми або набір ваги)
Опис проведення тесту	Користувач заповнює всі відповідні поля форми: вводить фізіологічні дані та ціль. Після відправки форми система обробляє введені значення та відображає на сторінці персоналізовану картку з розрахованим харчовим планом.
Очікуваний результат	Система коректно виконує розрахунок і показує користувачу його добову норму харчування згідно з введеними даними.
Фактичний результат	Розрахунок виконано успішно, користувач бачить сформований план харчування.

Таблиця 4.6 – Тест 6

Тест	Збереження розрахованої добової норми харчування
Модуль	Харчування
Номер тесту	1.6
Початковий стан системи	Користувач авторизований у системі та перебуває на сторінці розрахунку харчування.
Вхідні данні	Індивідуальні показники: зріст, маса тіла, вік, відсоток жирової тканини, рівень фізичної активності, ціль (наприклад, підтримання ваги або схуднення)

Продовження таблиці 4.6

Опис проведення тесту	Користувач вводить усі необхідні дані у відповідні поля форми. Після обробки інформації система відображає персоналізований план харчування. Далі користувач натискає кнопку збереження, щоб додати отримані дані до свого профілю.
Очікуваний результат	Результати розрахунку добової норми харчування успішно збережено для поточного користувача.
Фактичний результат	Дані харчового плану збережені в обліковому записі користувача без помилок.

Таблиця 4.7 – Тест 7

Тест	Додавання спожитих продуктів до щоденного раціону
Модуль	Харчування
Номер тесту	1.7
Початковий стан системи	Користувач авторизований та знаходиться на сторінці обліку харчування.
Вхідні данні	Назва продукту, кількість (вага), дата або прийом їжі (сніданок, обід тощо)

Продовження таблиці 4.7

Опис проведення тесту	Користувач натискає кнопку для додавання нового продукту. У відкритій формі вводить назву продукту, його вагу та день, до якого потрібно віднести споживання. Після цього натискає кнопку для підтвердження додавання до раціону.
Очікуваний результат	Обраний продукт успішно додається до списку спожитих у зазначений день та відображається на сторінці користувача.
Фактичний результат	Продукт доданий до раціону та відображений на сайті

Таблиця 4.8 – Тест 8

Тест	Перевірка досягнення добової норми харчування
Модуль	Харчування
Номер тесту	1.8
Початковий стан системи	Користувач авторизований у системі, перебуває на сторінці харчування та вже додав продукти, які були спожиті протягом дня.
Вхідні данні	
Опис проведення тесту	Користувач натискає кнопку для перевірки, чи відповідає фактичне споживання їжі встановленій індивідуальній добовій нормі. Система обчислює відхилення між рекомендованими та фактичними значеннями.

Продовження таблиці 4.8

Очікуваний результат	Якщо відхилення становить не більше 5% у будь-який бік, виводиться повідомлення про те, що денна норма харчування виконана.
Фактичний результат	Система визначила, що норма дотримана, та відобразила відповідне підтвердження на сторінці користувача.

Таблиця 4.9 – Тест 9

Тест	Оновлення аватарки користувача
Модуль	Акаунт користувача
Номер тесту	1.9
Початковий стан системи	Користувач авторизований та знаходиться на своїй персональній сторінці профілю.
Вхідні данні	Графічний файл (зображення)
Опис проведення тесту	Користувач клацає на поточну аватарку. Система відкриває інтерфейс вибору файлів, де користувач обирає нове зображення. Після підтвердження вибраного файлу система завантажує нову аватарку.
Очікуваний результат	Обране зображення встановлюється як нова аватарка, збережене у профілі користувача й відображається на сайті.
Фактичний результат	Зображення оновлено успішно — нова аватарка збереглася та відображається на персональній сторінці користувача.

4.3 Опис контрольного прикладу

Після запуску веб-застосунку користувач автоматично потрапляє на головну сторінку інтерфейсу.

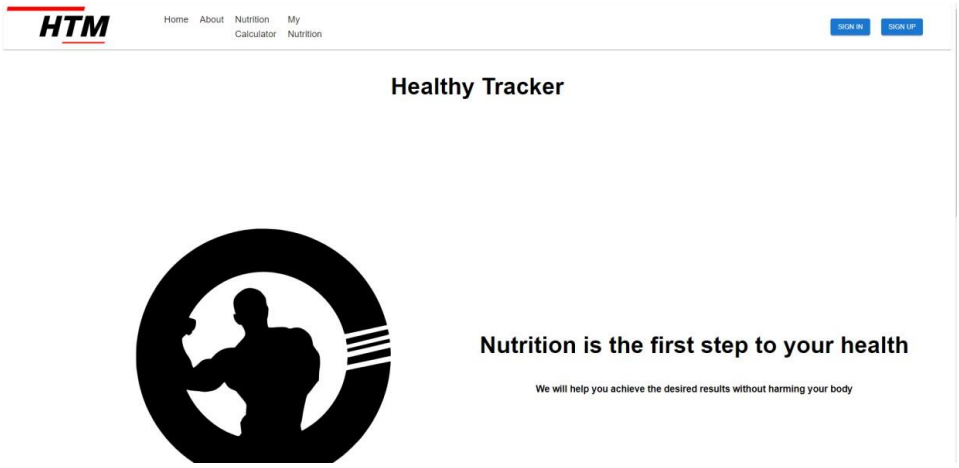


Рисунок 4.1 – Головна сторінка

Далі здійснюється перехід до форми авторизації.

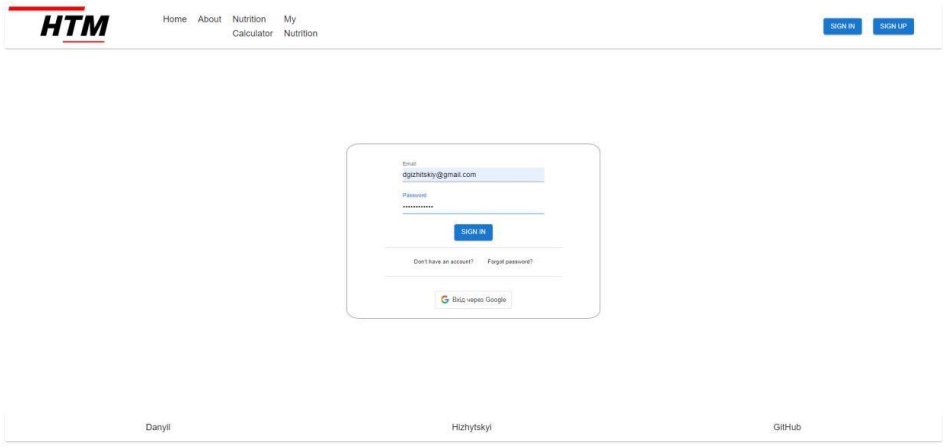


Рисунок 4.2 – Сторінка авторизації користувача

Після успішного введення облікових даних система перенаправляє користувача на його особисту головну сторінку. Інтерфейс завантажується без помилок, і з цього моменту стає доступним основний функціонал веб-застосунку.

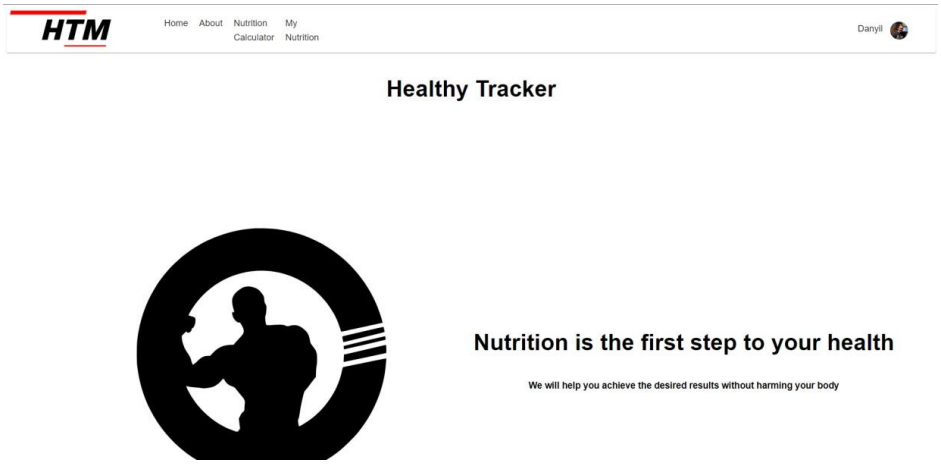


Рисунок 4.3 – Головна сторінка користувача

Наступним кроком користувач переходить на сторінку, призначену для розрахунку добової норми харчування, і вводить відповідні фізіологічні параметри.

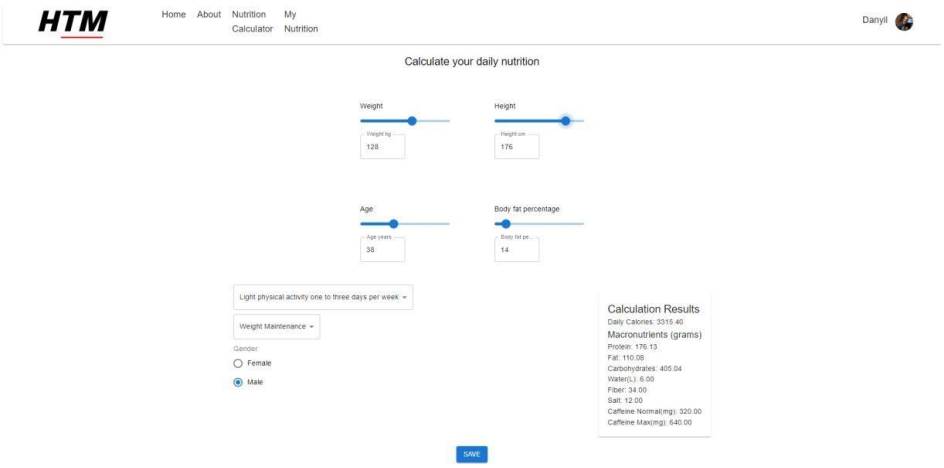


Рисунок 4.4 – Сторінка розрахунку харчування

Після завершення розрахунку перевіряється можливість збереження результату. Користувач натискає кнопку «Зберегти», після чого з’являється повідомлення про успішне збереження даних.

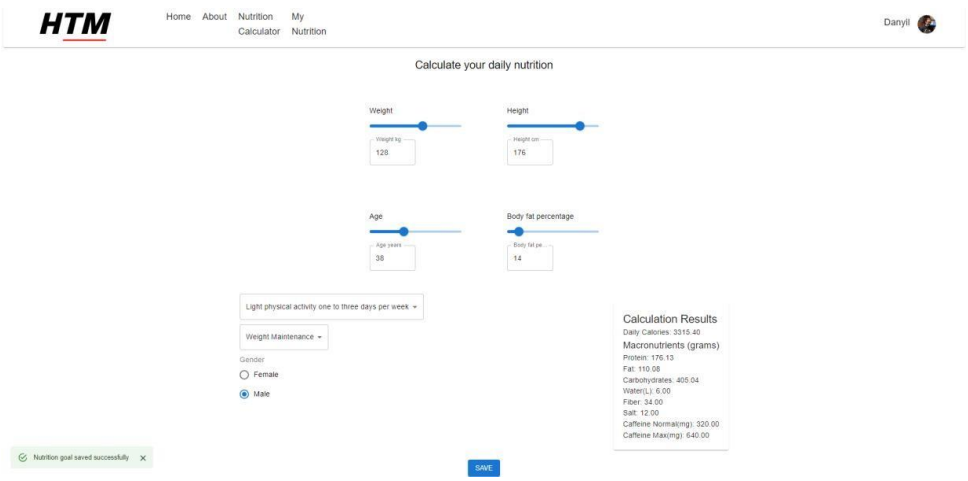


Рисунок 4.5 – Розраховане харчування збережене для користувача

Щоб додати продукти до щоденного раціону, користувач переходить на вкладку «Моє харчування».

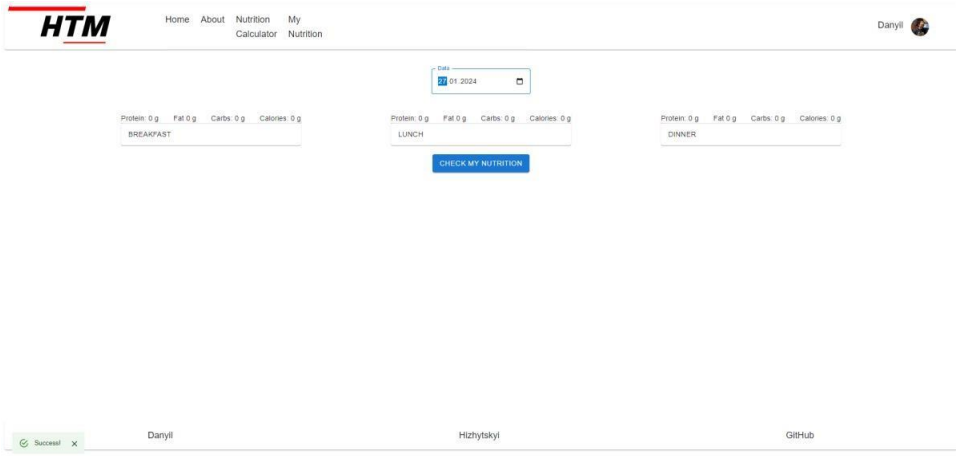


Рисунок 4.6 – Сторінка харчування користувача

Далі користувач розгортає відповідний прийом їжі та обирає опцію «Додати продукт». У полях, що з’являються, він зазначає назву продукту та його вагу.

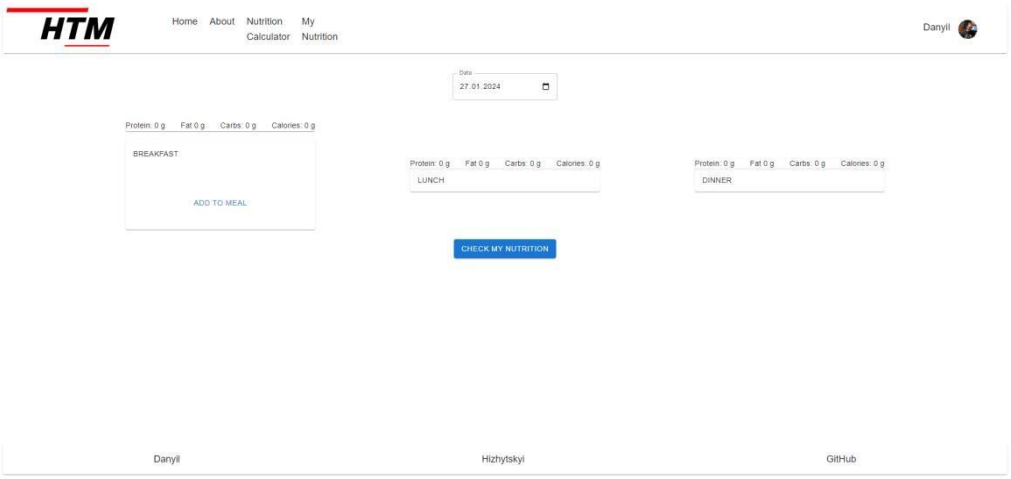


Рисунок 4.7 – Розгорнутий прийом їжі

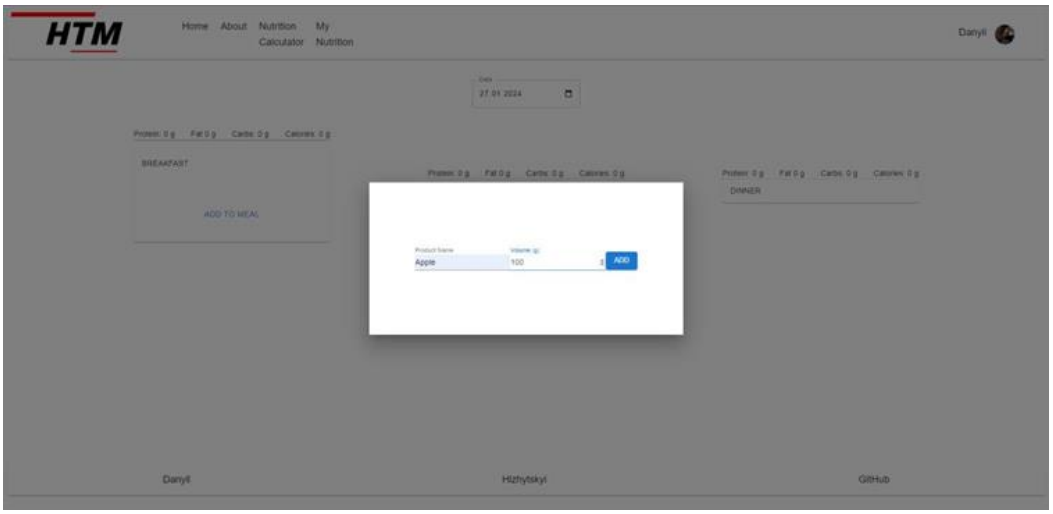


Рисунок 4.8 – Форма вводу продукту

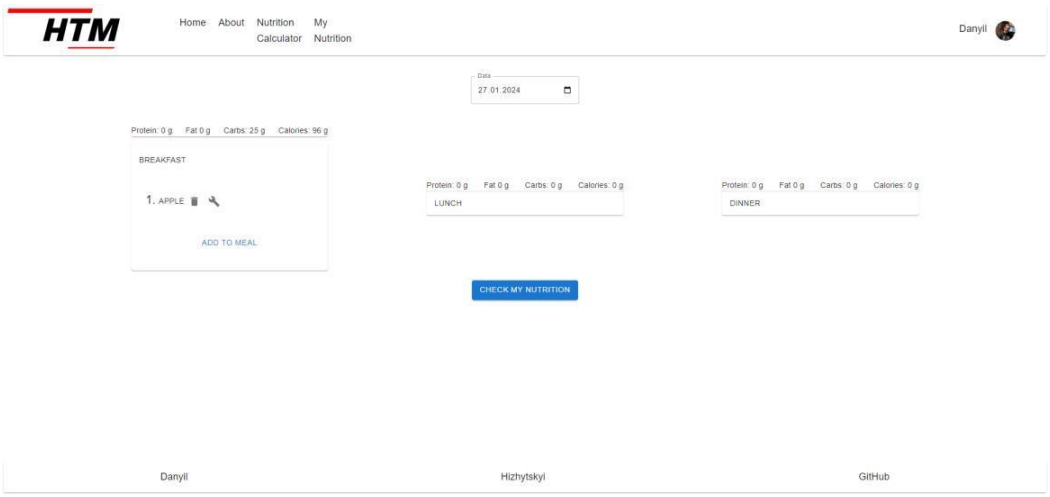


Рисунок 4.9 – Продукт доданий до харчування

Наступним етапом є перевірка виконання встановленої добової норми на основі доданих продуктів. Для цього виконується відповідна перевірка.

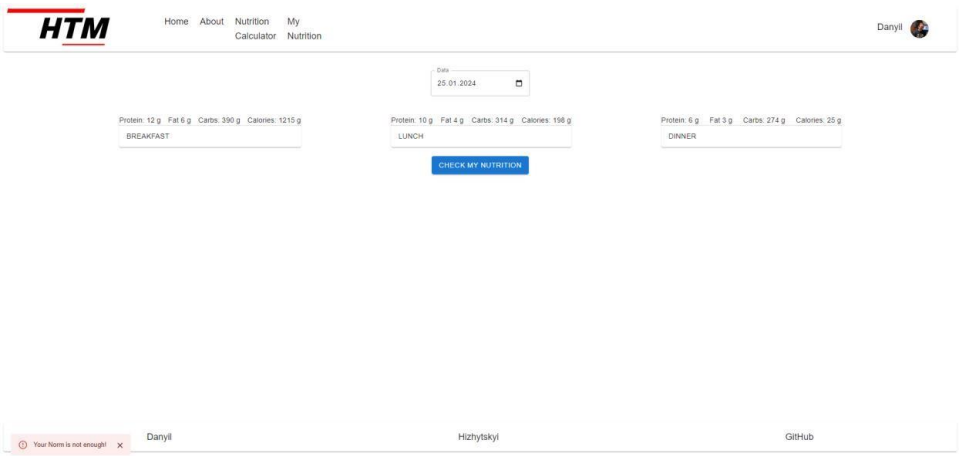


Рисунок 4.10 – Повідомлення що норма не виконана

У результаті з’являється повідомлення, що добова норма не виконана, оскільки загальна кількість спожитих продуктів не відповідає необхідним показникам.

Висновки до розділу

У цьому розділі була проведена оцінка якості функціонування створеного веб-застосунку шляхом його тестування. Основна мета полягала в перевірці відповідності функціоналу вимогам, пошуку й виправленні можливих помилок, а також у перевірці правильного збереження та обробки даних.

Було реалізовано ручне тестування всіх ключових функцій програмного продукту. Результати тестування зафіксовано в таблицях 4.1–4.9. Кожен тест містить опис початкових умов, вхідних даних, кроків виконання, а також очікуваних і фактичних результатів. Усі перевірки пройдені успішно, що підтверджує коректну роботу системи.

Крім того, представлено демонстраційний приклад, який ілюструє використання основного функціоналу з урахуванням типових сценаріїв та можливих відгалужень у логіці застосунку.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Процес розгортання програмного забезпечення надає кінцевому користувачеві можливість використання програмного засобу. На поточний момент існує велика кількість способів вирішення задачі розгортання веб застосувань, проте дані способи умовно можна розділити на розгортання на виділеному сервері та cloud рішення.

У випадку першого способу, для .net застосунків використовується веб сервер IIS[32]. При такому підході всі застосунки знаходяться на одному фізичному сервері, а управління і розмежування доступу здійснюється шляхом видачі прав доступу до певної аплікації чи пулу аплікацій.

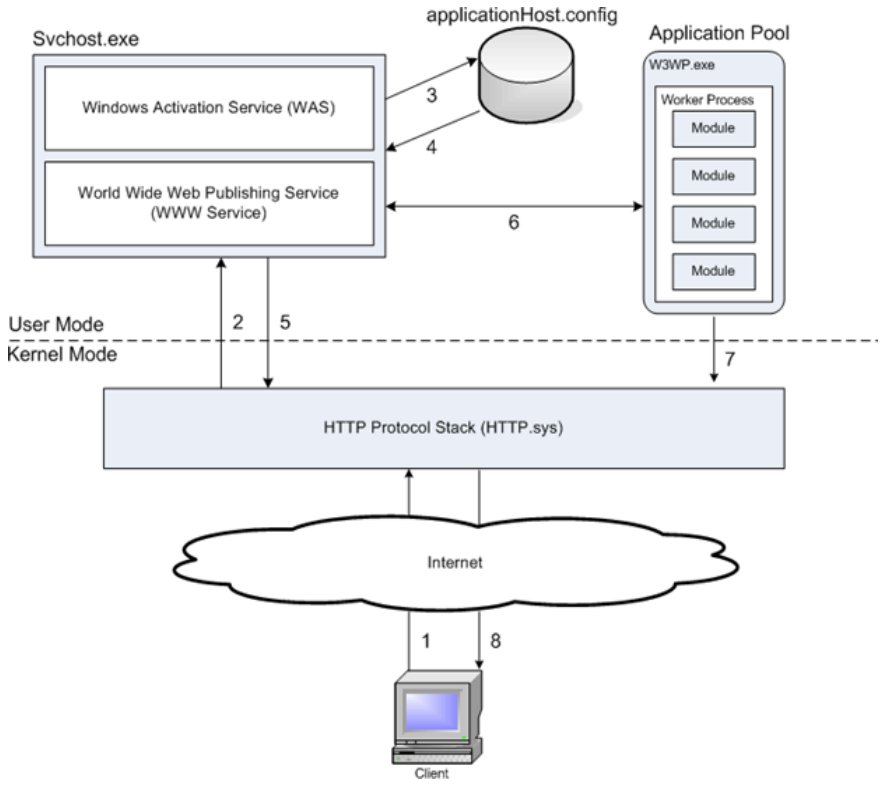


Рисунок 5.1 – Модель взаємодії компонентів IIS

Як видно з рис 5.1 Користувач взаємодіє з сервером, шдяхом відправки http/https запиту, після чого модуль http.sys виконує низько рівневу обробку запиту та маршрутиризує його до svhost.exe, що в свою чергу звертається до конфігурації і передає запит на виконання до пулу аплікацій, якій в залежності

від домейну здійснює маршрутизацію до конкретного пулу аплікацій, що представлений дочірнім процесом служби IIS W3WP.exe і як наслідок до конкретної аплікації, що здійснює подальшу обробку запиту. Очевидно, що при такому підході глобальним мінусом є відсутність можливості рівномірного розприділення ресурсів між аплікаціями, оскільки виділенням пам’яті та процесорним часом керує операційна система. Іншим мінусом даного підходу є підтримка працездатності аплікації та питання оновлення версії програмного коду. Саме через ці аспекти даний підхід є застарілим та майже не використовується.

Сучасною альтернативою є cloud рішення. Існує цілий ряд способів побудови та розміщення інфраструктури вебзастосунків на cloud платформах.

Першим варіантом є використання віртуальної машини, що надається хмарою. Даний підхід надає ізольованість поточного застосунку від решти, гарантує виділення і використання коректної кількості ресурсів, надає повний доступ до операційної системи і можливість встановлення з’єднання за допомогою RDP[33], проте є економічно не вигідним у порівнянні з іншими альтернативами.

Альтернативним підходом до розгортання є використання контейнерних технологій. Цей метод дозволяє досягти ізоляції виконання та ефективного розподілу ресурсів між компонентами системи. Хоча контейнери не забезпечують такого рівня контролю, як віртуальні машини, вони є оптимальним компромісом між ізольованістю, продуктивністю та витратами.

Для хостингу було обрано платформу Azure, оскільки вона надає широкі можливості для підтримки .NET-застосунків, а також інтегрується з інструментами розробки завдяки наявності вбудованих розширень і плагінів.

Для розгортання запропонованого рішення було обрано використання docker compose[34] в якості оркестратора для контейнерів, Azure App Service для розгортання в хмарі та Azure Database for PostgreSQL[35] для розгортання бази даних.

Для забезпечення розгортання даних сервісів в першу чергу потрібно створити групу ресурсів, після чого створити базу даних, після чого створити аплікації на основі докер компоуз файлу та відповідних докер файлів.

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\system32> az group create --name FoodManagerResourceGroup --location westeurope
```

Рисунок 5.2 – Команда для створення ресурс групи

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\system32> az postgres flexible-server create \
>>    --resource-group FoodManagerResourceGroup \
>>    --name FoodManagerPostgresServer \
>>    --admin-user pgadmin \
>>    --admin-password  \
>>    --location westeurope \
>>    --sku-name Standard_B1ms \
>>    --version 15
```

Рисунок 5.3 – Команда для створення бази даних

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\system32> az acr create --resource-group FoodManagerResourceGroup --name FoodManagerAcrRegistry --sku Basic
>> az acr login --name FoodManagerAcrRegistry
```

Рисунок 5.4 – Команда для налаштування Azure Container Registry

Результат розгортання зображено на рис 5.4 Діаграма розгортання.

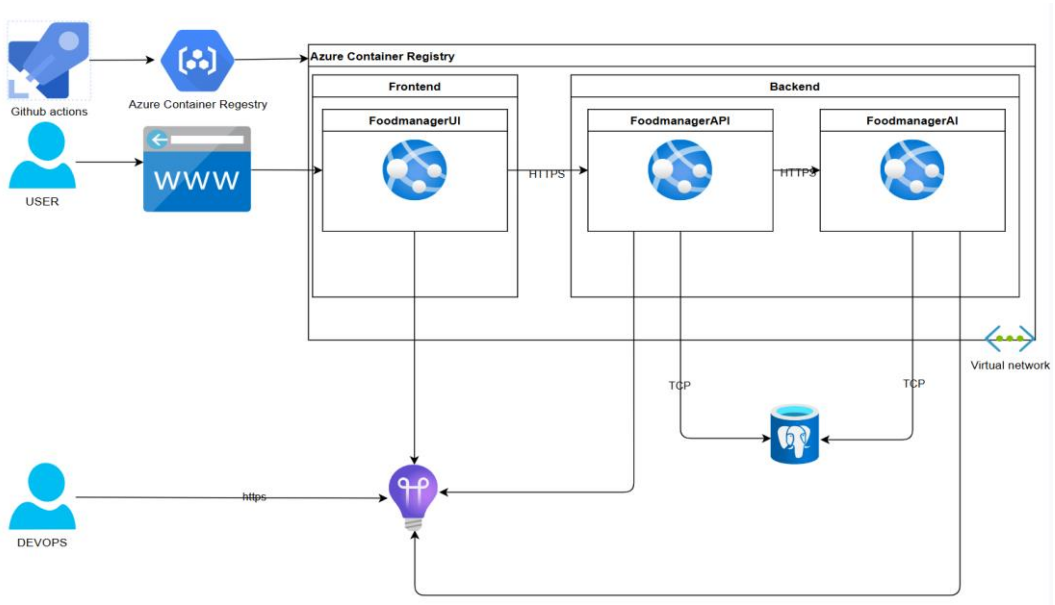


Рисунок 5.5 – Діаграма розгортання

5.2 Супровід програмного забезпечення

Користувачам необхідно забезпечити доступ до актуальної версії консольного застосунку з кожним новим релізом. Крім того, кожна нова версія повинна публікуватися в репозиторії npm. Для автоматизації цього процесу було застосовано механізм GitHub Actions. Зокрема, був створений workflow-файл .github/workflows/deploy.yml, який автоматизує послідовність дій при випуску нової версії застосунку [36].

Процес публікації запускається автоматично після додавання нової версії у гілку release. Далі система виконує збірку Docker-контейнерів, після чого вони завантажуються до Azure Container Registry (ACR), а пов’язані сервіси App Service оновлюються відповідно до нової версії застосунку.

```yaml
name: Deploy Docker Compose to Azure

on:
  push:
    branches:
      - release

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v3

    - name: Login to Azure
      uses:
      with:
        creds: ${{ secrets.AZURE_CREDENTIALS }}

    - name: Build and Push images to ACR
      uses:
      with:
        login-server: ${{ secrets.AZURE_ACR_NAME }}.azurecr.io
        username: ${{ secrets.AZURE_ACR_NAME }}
        password: ${{ secrets.ACR_PASSWORD }}

    - name: Build and Push frontend
      run: |
        docker build -t ${{ secrets.AZURE_ACR_NAME }}.azurecr.io/frontend:latest ./frontend
        docker push ${{ secrets.AZURE_ACR_NAME }}.azurecr.io/frontend:latest

    - name: Build and Push api
      run: |
        docker build -t ${{ secrets.AZURE_ACR_NAME }}.azurecr.io/backend:latest ./api
        docker push ${{ secrets.AZURE_ACR_NAME }}.azurecr.io/api:latest

    - name: Build and Push api
      run: |
        docker build -t ${{ secrets.AZURE_ACR_NAME }}.azurecr.io/backend:latest ./ai
        docker push ${{ secrets.AZURE_ACR_NAME }}.azurecr.io/ai:latest

    - name: Deploy to App Service
      uses: azure/webapps-deploy@v2
      with:
        app-name: ${{ secrets.AZURE_WEBAPP_NAME }}
        slot-name: production
        images: |
          ${{ secrets.AZURE_ACR_NAME }}.azurecr.io/frontend:latest
          ${{ secrets.AZURE_ACR_NAME }}.azurecr.io/api:latest
          ${{ secrets.AZURE_ACR_NAME }}.azurecr.io/ai:latest
```

Рисонок 5.6 – Листинг github actions скрипта.

```yaml
version: '3.4'

services:
  Foodmanager:
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
      - ASPNETCORE_URLS=https://+:443;http://+:80
    ports:
      - "5000:80"
      - "5001:443"
    volumes:
      - ${APPDATA}/Microsoft/UserSecrets:/root/.microsoft/usersecrets:ro
      - ${APPDATA}/ASP.NET/Https:/root/.aspnet/https:ro

  Foodmanager.api:
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
      - ASPNETCORE_URLS=https://+:443;http://+:80
    ports:
      - "5002:80"
      - "5003:443"
    volumes:
      - ${APPDATA}/Microsoft/UserSecrets:/root/.microsoft/usersecrets:ro
      - ${APPDATA}/ASP.NET/Https:/root/.aspnet/https:ro
  Foodmanager.ai:
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
      - ASPNETCORE_URLS=https://+:443;http://+:80
    ports:
      - "5004:80"
      - "5005:443"
    volumes:
      - ${APPDATA}/Microsoft/UserSecrets:/root/.microsoft/usersecrets:ro
      - ${APPDATA}/ASP.NET/Https:/root/.aspnet/https:ro

  Foodmanager.frontend:
    image: youracr.azurecr.io/frontend:latest
    build: ./frontend
    ports:
      - "4200:80"
    environment:
      - NODE_ENV=production

volumes:
    blobdata:
        external: false
```

Рисонок 5.7 – Листинг Docker compose файлу.

```dockerfile
FROM mcr.microsoft.com/dotnet/aspnet:3.1 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:3.1 AS build
WORKDIR /src
COPY ["FoodManager/FoodManager.csproj", "FoodManager/"]
RUN dotnet restore "FoodManager/FoodManager.csproj"
COPY . .
WORKDIR "/src/FoodManager"
RUN dotnet build "FoodManager.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "FoodManager.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "FoodManager.dll"]
```

Рисонок 5.8 – Листинг Docker файлу для API.

```
FROM mcr.microsoft.com/dotnet/aspnet:3.1 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:3.1 AS build
WORKDIR /src
COPY ["FoodManager/FoodManagerAI.csproj", "FoodManagerAI/"]
RUN dotnet restore "FoodManager/FoodManagerAI.csproj"
COPY . .
WORKDIR "/src/FoodManagerAI"
RUN dotnet build "FoodManagerAI.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "FoodManagerAI.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "FoodManagerAI.dll"]
```

Рисонок 5.9 – Листинг Docker файлу для AI.

```
FROM node:18 AS build
WORKDIR /clientapp/src
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

FROM nginx:alpine
COPY --from=build /clientapp/src/build /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Рисонок 5.10 – Листинг Docker файлу для UI.

Висновки до розділу

У цьому розділі було проаналізовано переваги використання Docker Compose для розгортання створеного програмного забезпечення.

Наведено поетапний опис процесу налаштування сервісів, необхідних для запуску застосунку в контейнерному середовищі. Також було розглянуто підхід до забезпечення користувачів актуальною версією застосунку, включаючи рішення щодо автоматичного оновлення та публікації нових релізів.

ВИСНОВКИ

У результаті виконаної розробки було створено програмне рішення, призначене для розрахунку індивідуальних харчових потреб і ведення обліку здорового харчування.

У процесі роботи було досліджено існуючі способи реалізації подібних систем і проведено їх аналіз. На основі цього аналізу сформульовано функціональні вимоги до системи, яка дозволяє здійснювати ефективний трекінг харчування користувача.

Розроблене програмне забезпечення реалізує наступний набір можливостей:

- розрахунок добової калорійності раціону користувача;
- визначення необхідної кількості макроелементів;
- перевірка дотримання рекомендованої норми харчування;
- ведення персонального харчового щоденника;
- додавання спожитих продуктів до денного раціону.

Під час проведення тестування було підтверджено, що застосунок повністю відповідає як функціональним, так і нефункціональним вимогам.

Фінальна версія програмного продукту була успішно розгорнута за допомогою інструменту Docker Compose, що дозволило спростити процес деплойменту. Крім того, було реалізовано механізм отримання оновлень, що забезпечує користувачам доступ до актуальних версій застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Застосунок FatSecret [Електронний ресурс] – Режим доступу: https://www.fatsecret.com/

2) Застосунок Калькулятор Калорій [Електронний ресурс] – Режим доступу: https://calc.tablycjakalorijnosti.com.ua/

3) Застосунок kyiv-dpss.gov [Електронний ресурс] – Режим доступу: http://kyiv-dpss.gov.ua/calk.html

4) Застосунок Musclewiki [Електронний ресурс] – Режим доступу: https://musclewiki.com/

5) Застосунок Freedieting [Електронний ресурс] – Режим доступу: http://www.freedieting.com/

6) What Is SOA? – Режим доступу: https://www.oracle.com/ua/service-oriented-architecture-soa/

7) What Is SOA? – Режим доступу: https://www.oracle.com/ua/service-oriented-architecture-soa/

8) AWS microservices – Режим доступу: https://aws.amazon.com/microservices/monolith/

9) Service Oriented Architecture – Режим доступу: https://www.marketresearch.com/Global-Industry-Analysts-v1039/Service-Oriented-Architecture-40817829/

10) Microservices vs Monolithic Architecture: A Comparison to Guide Your 2025 Strategy – Режим доступу: https://kitrum.com/blog/microservices-vs-monolithic-architecture/

11) 90+ Cloud Computing Statistics: A 2025 Market Snapshot – Режим доступу: https://www.cloudzero.com/blog/cloud-computing-statistics/?utm_source=chatgpt.com

12) The Future of Service-Oriented Architecture - Top Trends & Predictions for Software Architects – Режим доступу: https://moldstud.com/articles/p-the-future-of-service-oriented-architecture-top-trends-predictions-for-software-architects?utm_source=chatgpt.com

13) Microservices Architecture Market Report by Component – Режим доступу: https://www.imarcgroup.com/microservices-architecture-market

14) Microservices and the Single Responsibility Principle (SRP) — Simplified – Режим доступу: https://medium.com/@premsaiarroju/microservices-and-the-single-responsibility-principle-srp-explained-in-depth-a0c3ec4a73f8
15) Unit Testing - Software Testing Режим доступу: https://www.geeksforgeeks.org/unit-testing-software-testing/
16) Бізнес-аналітики і нефункціональні вимоги в agile розробці – Режим доступу: hhttps://www.ba.in.ua/2023/08/18/biznes-analityky-i-nefunkczionalni-vymogy-v-agile-rozrobczi/
17) Фреймворк React [Електронний ресурс] – Режим доступу: https://react.dev/
18) Токен JWT [Електронний ресурс] – Режим доступу: https://jwt.io/
19) N-tier architecture [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Multitier_architecture
20) База даних SQL Server [Електронний ресурс] – Режим доступу: https://www.microsoft.com/en-us/sql-server
21) Мова програмування C# [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/
22) Мова програмування TypeScript [Електронний ресурс] – Режим доступу: https://www.typescriptlang.org/
23) Бібліотека React Router [Електронний ресурс] – Режим доступу: https://reactrouter.com/en/main
24) 7 Best C# IDEs in 2022 [Електронний ресурс] – Режим доступу: https://www.codeguru.com/tools/7-best-c-sharp-ide/
25) JetBrains DataGrip [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/datagrip/
26) BPMN [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/BPMN

27) JetBrains Rider [Електронний ресурс] – Режим доступу:https://www.jetbrains.com/rider/
28) Visual Studio Code [Електронний ресурс] – Режим доступу:https://code.visualstudio.com/
29) Функціональні вимоги [Електронний ресурс] – Режим доступу: https://visuresolutions.com/uk/blog/functional-requirements/
30) Таблиця трасування вимог [Електронний ресурс] – Режим доступу: https://qalight.ua/baza-znaniy/matriczya-vidpovidnosti-vimog-requirements-traceability- matrix/
31) GDPR [Електронний ресурс] – Режим доступу: https://ajax.systems/ua/gdpr-white-paper/
32) IIS[Електронний ресурс] – Режим доступу: https://www.techtarget.com/searchwindowsserver/definition/IIS
33) RDP[Електронний ресурс] – Режим доступу: https://delinea.com/what-is/rdp-remote-desktop-protocol
34) Docker Compose[Електронний ресурс] – Режим доступу: https://spacelift.io/blog/docker-compose
35) Azure Database for PostgreSQL[Електронний ресурс] – Режим доступу: https://azure.microsoft.com/en-us/products/postgresql
36) GitHub Actions workflow [Електронний ресурс] – Режим доступу: https://codefresh.io/learn/github-actions/github-actions-workflows-basics-examples-and-a-quick-tutorial/
37) Postman[Електронний ресурс] – Режим доступу:https://www.geeksforgeeks.org/introduction-postman-api-development/

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

Звіт подібності

метадані

Назва організації

National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute

Заголовок

ІП-11_Гіжицький_ПЗ

Автор Науковий керівник / Експерт

ІП-11_ГіжицькийЛіщук К.І.

підрозділ

ФІОТ, К-а інформатики та програмної інженерії

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

КП 1

10	10013	82987
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРАХУНКУ ТА ТРЕКІНГУ ЗДОРОВОГО ХАРЧУВАННЯ

Текст програми

КПІ.ІП-1106.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Катерина ЛІЩУК

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Даниїл Гіжицький

Київ – 2025

Посилання на репозиторій з повним текстом програмного коду

https://github.com/doomer2004/HealthyTracker

Файл AuthService.cs

Реалізація функціональної задачі реєстрації користувача:

```
using AutoMapper;
using HealthyTracker.BLL.Extensions;
using HealthyTracker.BLL.Services.Auth.Interfaces;
using HealthyTracker.Client.Nutrition;
using HealthyTracker.Common.Models;
using HealthyTracker.Common.Models.Configs;
using HealthyTracker.Common.Models.DTOs.Auth;
using HealthyTracker.Common.Models.DTOs.Error;
using HealthyTracker.DAL.Entities;
using HealthyTracker.Email.Models;
using HealthyTracker.Email.Services.Interfaces;
using LanguageExt;
using LanguageExt.Common;
using Microsoft.AspNetCore.Identity;
using Microsoft.Extensions.Logging;

namespace HealthyTracker.BLL.Services.Auth.Auth;

public class AuthService : AuthServiceBase, IAuthService
{
    private readonly IMapper _mapper;
    private readonly IEmailSender _emailSender;
    private readonly IUserRegistrationService _userRegistrationService;

    public AuthService(
        UserManager<User> userManager,
        JwtConfig jwtConfig,
        ILogger<AuthServiceBase> logger,
        IMapper mapper,
        IEmailSender emailSender,
        IUserRegistrationService userRegistrationService)
        : base(userManager, jwtConfig, logger)
    {
        _mapper = mapper;
```

```
    _emailSender = emailSender;
    _userRegistrationService = userRegistrationService;
}

public async Task<Either<ErrorDto, SignUpResponseDto>> SignUpAsync(SignUpDTO dto)
{

    var user = await _userManager.FindByEmailAsync(dto.Email);
    if (user is not null)
        return new AlreadyExistsErrorDTO("User with this email already exists");



    user = _mapper.Map<User>(dto);
    var createdUser = await _userManager.CreateAsync(user, dto.Password);
    if (!createdUser.Succeeded)
    {
        _logger.LogIdentityError(user, createdUser);
        return new IdentityErrorDTO(
            "Unable to create a user. Please try again later or use another email address");
    }

    var generateUrl = await _userRegistrationService.GenerateEmailConfirmationUrlAsync(user.Id);
    return await generateUrl.Match<Task<Either<ErrorDto, SignUpResponseDto>>>(
        Right: async url =>
        {
            var emailSent = await _emailSender.SendEmailAsync(user.Email!,
                new EmailConfirmationMessage() {Url = url});

            return emailSent.Match<Either<ErrorDto, SignUpResponseDto>>(
                None: () =>
                {
                    _logger.LogInformation("Email confirmation url sent to user {0}", user.Id);
                    return new SignUpResponseDto {UserId = user.Id};
                },
                Some: error =>
                {
                    _logger.LogError("Unable to send email confirmation url to user {0}", user.Id);
                    return new ExternalErrorDTO(error.Message);
                });
        },
        Left: error =>
        {
```

```
            _logger.LogError("Unable to generate email confirmation url for user {0}, error: {1}", user.Id,
                error.Message);

            return Task.FromResult<Either<ErrorDto, SignUpResponseDto>>(
                new IncorrectParametersErrorDTO(error.Message));
        });
    }

    public async Task<Either<ErrorDto, AuthSuccessDTO>> SignInAsync(SignInDTO dto)
    {
        var user = await _userManager.FindByEmailAsync(dto.Email);
        if (user is null)
            return new NotFoundErrorDTO("User with this email does not exist");

        if (!user.EmailConfirmed)
            return new IncorrectParametersErrorDTO("Confirm your email before signing in");

        var validPassword = await _userManager.CheckPasswordAsync(user, dto.Password);
        if (!validPassword)
            return new IncorrectParametersErrorDTO("Email or password is incorrect");

        return await GenerateAuthResultAsync(user);
    }
}
```

Файл UserRegistrationService.cs

Реалізація функціональної задачі підтвердження реєстрації:

```
using System.Web;
using HealthyTracker.BLL.Services.Auth.Auth;
using HealthyTracker.BLL.Services.Auth.Interfaces;
using HealthyTracker.Common.Models.Configs;
using HealthyTracker.Common.Models.DTOs.Error;
using HealthyTracker.Common.Models.Utility;
using HealthyTracker.DAL.Entities;
using HealthyTracker.DAL.Repositories;
using HealthyTracker.DAL.Repositories.Interfaces;
using LanguageExt;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;

namespace HealthyTracker.BLL.Services.UserServices.Services;
```

```
public class UserRegistrationService : IUserRegistrationService
{
    private readonly IUserRegistrationRepository _userRegistrationRepository;
    private readonly AuthConfig _authConfig;
    private readonly UserManager<User> _userManager;
    private readonly CallbackUrisConfig _urisConfig;

    public UserRegistrationService(IUserRegistrationRepository userRegistrationRepository, AuthConfig authConfig,
        UserManager<User> userManager, CallbackUrisConfig urisConfig)
    {
        _userRegistrationRepository = userRegistrationRepository;
        _authConfig = authConfig;
        _userManager = userManager;
        _urisConfig = urisConfig;
    }

    public async Task<Option<ErrorModel>> CanConfirmEmailAsync(Guid userId, string url)
    {
        var registration = await _userRegistrationRepository.Table
            .Include(x => x.User)
            .FirstOrDefaultAsync(x => x.UserId == userId);

        if (registration is null)
            return new ErrorModel("You have not requested email confirmation");

        if (registration.ExpiresAt < DateTimeOffset.UtcNow)
            return new ErrorModel("Confirmation code has expired. Please request a new one");

        await _userRegistrationRepository.Delete(registration);
        return Option<ErrorModel>.None;
    }

    public async Task<Either<ErrorModel, string>> RegenerateEmailConfirmationUrlAsync(Guid userId)
    {
        var registration = await _userRegistrationRepository.Table
            .FirstOrDefaultAsync(r => r.UserId == userId);

        if (registration is null)
            return new ErrorModel("You have not requested email confirmation");

        if (registration.IsUrlRegenerated)
```

```
            return new ErrorModel("You have already requested a new url");

        registration.Url = (string) await GenerateEmailConfirmationUrlAsync(userId);
        registration.ExpiresAt = DateTimeOffset.UtcNow.Add(_authConfig.ConfirmationUrlLifetime);
        registration.IsUrlRegenerated = true;
        await _userRegistrationRepository.Update(registration);

        return registration.Url;
    }

    public async Task<Either<ErrorModel, string>> GenerateEmailConfirmationUrlAsync(Guid userId)
    {
        var user = await _userManager.FindByIdAsync(userId.ToString());

        var token = await _userManager.GeneratePasswordResetTokenAsync(user);
        var encodedToken = HttpUtility.UrlEncode(token);
        var callbackUrl = string.Format(_urisConfig.ResetPasswordUriTemplate, user.Email, encodedToken);

        return callbackUrl;
    }
}
```

Файл ProductService.cs

Реалізація функціональної задачі додавання продуктів до харчування:

```
using AutoMapper;
using HealthyTracker.BLL.Services.ProductService.Interfaces;
using HealthyTracker.Client.Nutrition;
using HealthyTracker.Client.Nutrition.Models.Requests;
using HealthyTracker.Client.Nutrition.Models.Responses;
using HealthyTracker.Common.Enums;
using HealthyTracker.Common.Models.DTOs;
using HealthyTracker.Common.Models.DTOs.Error;
using HealthyTracker.Common.Models.DTOs.Product;
using HealthyTracker.Common.Models.Utility;
using HealthyTracker.DAL.Entities;
using HealthyTracker.DAL.Repositories;
using HealthyTracker.DAL.Repositories.Interfaces;
using LanguageExt;
using LanguageExt.ClassInstances;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;
```

```
namespace HealthyTracker.BLL.Services.ProductService.Services;

public class ProductService : IProductService
{
    private readonly IProductRepository _productRepository;
    private readonly IMapper _mapper;
    private readonly INutritionsClient _nutritionsClient;
    private readonly IMealRepository _mealRepository;
    private readonly ILogger _logger;

    public ProductService(IProductRepository productRepository, IMapper mapper,
        INutritionsClient nutritionsClient, IMealRepository mealRepository, ILogger<ProductService> logger)
    {
        _productRepository = productRepository;
        _mapper = mapper;
        _nutritionsClient = nutritionsClient;
        _mealRepository = mealRepository;
        _logger = logger;
    }
    public async Task<Option<ErrorModel>> AddProduct(string name, int volume, Guid mealId)
    {
        try
        {
            var request = await _nutritionsClient.GetNutritionsByNameAsync
                (new GetNutritionByNameRequest {Query = name});
            var products = ParseProducts(request);
            var product = await ProductByParams(products, volume);
            var productResponse = _mapper.Map<Product>(product);
            _mealRepository.Table.First(meal => meal.Id == mealId).Products.Add(productResponse);
            await _mealRepository.SaveChangesAsync();
            return Option<ErrorModel>.None;
        }
        catch (Exception e)
        {
            _logger.LogError(e.Message);
            return new ErrorModel(e.Message);
        }
    }

    public async Task<Option<ErrorModel>> UpdateProduct(Guid productId, int volume)
    {
```

```
    try
    {
        var product = _productRepository.Table.Where(p => p.Id == productId);
        ;
        var updatedProduct = await ProductByParams(product.ToList(), volume);
        var productResponse = _mapper.Map<Product>(product);
        await _productRepository.Update(productResponse);
        return Option<ErrorModel>.None;
    }
    catch (Exception e)
    {
        _logger.LogError(e.Message);
        return new ErrorModel(e.Message);
    }
}

public async Task<Option<ErrorModel>> DeleteProduct(Guid productId)
{
    try
    {
        var product = await _productRepository.Table.FirstAsync(p => p.Id == productId);

        await _productRepository.Delete(product);

        return Option<ErrorModel>.None;
    }
    catch (Exception e)
    {
        _logger.LogError(e.Message);
        return new ErrorModel(e.Message);
    }
}

private List<Product> ParseProducts(GetNutritionByNameResponse response)
{
    if (response?.Foods != null && response.Foods.Any())
    {
        return response.Foods.Select(food => new Product
        {
            ProductName = food.FoodName,
            Volume = (int) food.ServingWeightGrams,
            Calories = Convert.ToSingle(food.FullNutrients.First(nutrient => nutrient.AttrId == 208).Value),
```

```
            Protein = Convert.ToSingle(food.FullNutrients.First(nutrient => nutrient.AttrId == 203).Value),
            Fat = Convert.ToSingle(food.FullNutrients.First(nutrient => nutrient.AttrId == 204).Value),
            Carbs = Convert.ToSingle(food.FullNutrients.First(nutrient => nutrient.AttrId == 205).Value),
            Salt = Convert.ToSingle(food.FullNutrients.First(nutrient => nutrient.AttrId == 307).Value),
            Caffeine = Convert.ToSingle(food.FullNutrients.First(nutrient => nutrient.AttrId == 262).Value),
            Water = Convert.ToSingle(food.FullNutrients.First(nutrient => nutrient.AttrId == 255).Value),
        }).ToList();
    }

    return new List<Product>();
}

private Task<ProductDTO> ProductByParams(List<Product> products, int volume)
{
    var product = products.First();

    var productParams = new ProductDTO
    {
        Volume = 1,
        Calories = (product.Calories / product.Volume) + (product.Calories % product.Volume),
        Protein = (product.Protein / product.Volume) + (product.Protein % product.Volume),
        Fat = (product.Fat / product.Volume) + (product.Fat % product.Volume),
        Carbs = (product.Carbs / product.Volume) + (product.Carbs % product.Volume),
        Salt = (product.Salt / product.Volume) + (product.Salt % product.Volume),
        Caffeine = (product.Caffeine / product.Volume) + (product.Caffeine % product.Volume),
        Water = (product.Water / product.Volume) + (product.Water % product.Volume)
    };

    var realProduct = new ProductDTO
    {
        Volume = volume,
        Calories = productParams.Calories * volume,
        Protein = productParams.Protein * volume,
        Fat = productParams.Fat * volume,
        Carbs = productParams.Carbs * volume,
        Salt = productParams.Salt * volume,
        Caffeine = productParams.Caffeine * volume,
        Water = productParams.Water * volume
    };

    return Task.FromResult(realProduct);
}
```

```
}
```

Файл NutritionGoalService.cs

Реалізація функціональної задачі розрахунку виконання норми харчування:

```
using AutoMapper;
using HealthyTracker.BLL.Services.NutritionService.Interfaces;
using HealthyTracker.Common.Models.DTOs.Calories;
using HealthyTracker.Common.Models.DTOs.Error;
using HealthyTracker.DAL.Entities;
using HealthyTracker.DAL.Repositories;
using HealthyTracker.DAL.Repositories.Interfaces;
using LanguageExt;
using Microsoft.EntityFrameworkCore;

namespace HealthyTracker.BLL.Services.NutritionService.Services;

public class NutritionGoalService : INutritionGoalService
{
   private readonly INutritionGoalRepository _nutritionGoalRepository;
   private readonly IMapper _mapper;

   public NutritionGoalService(INutritionGoalRepository nutritionGoalRepository,
      IMapper mapper)
   {
      _nutritionGoalRepository = nutritionGoalRepository;
      _mapper = mapper;
   }

   public async Task<Either<ErrorDto, NutritionGoalDTO>> GetAsync(Guid userId)
   {
      var goal = await _nutritionGoalRepository.Table.Include(u => u.User)
         .FirstOrDefaultAsync(u => u.UserId == userId);

      if (goal is null)
         return new NotFoundErrorDTO("User with this id does not exist");

      return _mapper.Map<NutritionGoalDTO>(goal);
   }

   public async Task SaveAsync(Guid userId, NutritionGoalDTO dto)
   {
         var goal = await _nutritionGoalRepository.Table.FirstOrDefaultAsync(g => g.UserId == userId); ;
```

```
            if (goal is null)
            {
                await AddAsync(userId, dto);
                return;
            }

            await UpdateAsync(userId, dto);
        }

        private async Task<bool> AddAsync(Guid userId, NutritionGoalDTO dto)
        {
            var newGoal = _mapper.Map<NutritionGoal>(dto);
            newGoal.UserId = userId;
            await _nutritionGoalRepository.Insert(newGoal);
            return true;
        }

        private async Task<bool> UpdateAsync(Guid userId, NutritionGoalDTO dto)
        {
            var nutrition = _mapper.Map<NutritionGoal>(dto);
            await _nutritionGoalRepository.Insert(nutrition);

            return true;
        }
    }
```

Файл MealService.cs

Реалізація функціональної задачі додавання прийомів їжі:

```
using AutoMapper;
using HealthyTracker.BLL.Services.MealService.Interfaces;
using HealthyTracker.Common.Models.DTOs.Meal;
using HealthyTracker.Common.Models.DTOs.Nutrition;
using HealthyTracker.Common.Models.DTOs.Product;
using HealthyTracker.DAL.Entities;
using HealthyTracker.DAL.Repositories;
using HealthyTracker.DAL.Repositories.Interfaces;
using LanguageExt;
using Microsoft.EntityFrameworkCore;

namespace HealthyTracker.BLL.Services.MealService.Services;

public class MealService : IMealService
```

```
{
    private readonly IMapper _mapper;
    private readonly IMealRepository _mealRepository;
    private readonly IProductRepository _productRepository;

    public MealService(IMapper mapper, IMealRepository mealRepository,
        IProductRepository productRepository)
    {
        _mapper = mapper;
        _mealRepository = mealRepository;
        _productRepository = productRepository;
    }

    public async Task AddMealAsync(Guid userId, Guid dailyId)
    {
        var meal = new Meal
        {
            DailyId = dailyId,
            Products = new List<Product>(3)
        };

        await _mealRepository.Insert(meal);
        return;
    }

    public async Task<List<ProductDTO>> GetAllProductsAsync(Guid userId,
        DateTime date, Guid mealId)
    {
        var products = await _productRepository.GetAll();

        return _mapper.Map<List<ProductDTO>>(products);
    }

    public async Task<GetMealNutritionDTO> GetNutritionAsync(Guid userId, Guid mealId)
    {
        var meals = await _mealRepository.Table.Include(u => u.Daily)
            .ThenInclude(d => d.UserId).Where(u => u.Daily.UserId == userId).ToListAsync();

        var nutrition = new GetMealNutritionDTO();
        foreach (var meal in meals)
        {
            nutrition.Calories += meal.Products.Sum(p => p.Calories);
```

```
            nutrition.Protein += meal.Products.Sum(p => p.Protein);
            nutrition.Fat += meal.Products.Sum(p => p.Fat);
            nutrition.Carbs += meal.Products.Sum(p => p.Carbs);
        }

        return nutrition;
    }

    public async Task<List<Meal>> GetUserMealAsync(Guid userId)
    {
        var nutrition = await _mealRepository.Table.Include(u => u.Daily)
            .ThenInclude(d => d.UserId).Where(u => u.Daily.UserId == userId).ToListAsync();

        return nutrition;
    }
}
```

Файл DailyService.cs

Реалізація функціональної задачі перевірки норми харчування:

```
using AutoMapper;
using HealthyTracker.BLL.Services.DailyService.Interfaces;
using HealthyTracker.BLL.Services.MealService.Interfaces;
using HealthyTracker.Common.Enums;
using HealthyTracker.Common.Models.DTOs.Nutrition;
using HealthyTracker.DAL.Entities;
using HealthyTracker.DAL.Repositories;
using HealthyTracker.DAL.Repositories.Interfaces;
using LanguageExt;
using Microsoft.EntityFrameworkCore;

namespace HealthyTracker.BLL.Services.DailyService.Services;

public class DailyService : IDailyService
{
    private readonly IMapper _mapper;
    private readonly IDailyRepository _dailyRepository;
    private readonly IMealService _mealService;
    private readonly INutritionGoalRepository _nutritionGoalRepository;

    public DailyService(IMapper mapper, IDailyRepository dailyRepository,
        IMealService mealService, INutritionGoalRepository nutritionGoalRepository)
    {
```

```
    _mapper = mapper;
    _dailyRepository = dailyRepository;
    _mealService = mealService;
    _nutritionGoalRepository = nutritionGoalRepository;
}


public async Task<bool> CheckDailyAsync(Guid userId)
{
    var daily = await _dailyRepository.Table.FirstOrDefaultAsync(d => d.UserId == userId);
    if (daily is null)
        return false;

    var meals = await _mealService.GetUserMealAsync(userId);
    var nutrition = new GetNutritionDTO();
    foreach (var temp in meals.Select(meal => _mapper.Map<GetNutritionDTO>(meal)))
    {
        nutrition.Calories += temp.Calories;
        nutrition.Protein += temp.Protein;
        nutrition.Fat += temp.Fat;
        nutrition.Carbs += temp.Carbs;
    }

    var goal = await _nutritionGoalRepository.Table.FirstOrDefaultAsync(g => g.UserId == userId);

    if (goal is null)
    {
        daily.NormIsFulfilled = false;
        return false;
    }
        //return "The norm is not defined";
    if (goal.Calories > nutrition.Calories)
    {
        daily.NormIsFulfilled = false;
        return false;
    }
        //return $"You have not eaten enough calories(need {goal.Calories} got {nutrition.Calories})";
    if (goal.Protein > nutrition.Protein)
    {
        daily.NormIsFulfilled = false;
        return false;
    }
```

```
            //return $"You have not eaten enough protein(need {goal.Protein} got {nutrition.Protein})";
        if (goal.Fat > nutrition.Fat)
        {
            daily.NormIsFulfilled = false;
            return false;
        }
            //return $"You have not eaten enough fat(need {goal.Fat} got {nutrition.Fat})";
        if (goal.Carbs > nutrition.Carbs)
        {
            daily.NormIsFulfilled = false;
            return false;
        }
            //return $"You have not eaten enough carbs(need {goal.Carbs} got {nutrition.Carbs})";

        daily.NormIsFulfilled = true;
        await _dailyRepository.Update(daily);

        return true;
        //return "Daily is fulfilled";
    }
}
```

Файл NutritionsClient.cs

Впровадження стороннього API з продуктами харчування:

```
using HealthyTracker.Client.Nutrition.Extensions;
using HealthyTracker.Client.Nutrition.Models.Requests;
using HealthyTracker.Client.Nutrition.Models.Responses;
using HealthyTracker.Common.Models.Configs;
using Microsoft.Extensions.Logging;
using RestSharp;

namespace HealthyTracker.Client.Nutrition;
public class NutritionsClient : INutritionsClient
{
    private readonly IHttpClientFactory _clientFactory;
    private readonly ILogger<NutritionsClient> _logger;
    private readonly NutritionsConfig _nutritionsConfig;
    public NutritionsClient(
        IHttpClientFactory clientFactory,
        ILogger<NutritionsClient> logger, NutritionsConfig nutritionsConfig)
    {
```

```
        _clientFactory = clientFactory;
        _logger = logger;
        _nutritionsConfig = nutritionsConfig;
    }



    public async Task<GetNutritionByNameResponse?> GetNutritionsByNameAsync(GetNutritionByNameRequest
request)
    {
        try
        {
            var client = _clientFactory.CreateRestClient("NutritionsClient");

            var restRequest = new RestRequest("https://trackapi.nutritionix.com/v2/natural/nutrients")
                .AddHeader("x-app-id", _nutritionsConfig.AppId)
                .AddHeader("x-app-key", _nutritionsConfig.AppKey)
                .AddBody(request);

            var result = await client.PostAsync<GetNutritionByNameResponse>(restRequest);

            return result;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error in Nutrition client");
            return null;
        }
    }
}
```

Файл Program.cs

Збір та запуск сервісної частини:

```
using System.Net;
using System.Net.Mail;
using System.Text;
using FluentEmail.MailKitSmtp;
using HealthyTracker.BLL.Extensions;
using HealthyTracker.BLL.Services.Auth.Auth;
using HealthyTracker.BLL.Services.Auth.Interfaces;
using HealthyTracker.BLL.Services.DailyService.Interfaces;
using HealthyTracker.BLL.Services.DailyService.Services;
```

```
using HealthyTracker.BLL.Services.MealService.Interfaces;
using HealthyTracker.BLL.Services.MealService.Services;
using HealthyTracker.BLL.Services.NutritionService.Interfaces;
using HealthyTracker.BLL.Services.NutritionService.Services;
using HealthyTracker.BLL.Services.ProductService.Interfaces;
using HealthyTracker.BLL.Services.ProductService.Services;
using HealthyTracker.BLL.Services.UserServices.Services;
using HealthyTracker.Client.Nutrition;
using HealthyTracker.Common.Models.Configs;
using HealthyTracker.DAL.Contexts;
using HealthyTracker.DAL.Entities;
using HealthyTracker.DAL.Repositories;
using HealthyTracker.DAL.Repositories.Interfaces;
using HealthyTracker.Email.Services;
using HealthyTracker.Email.Services.Interfaces;
using HealthyTracker.Extensions;
using HealthyTracker.Mapping.Profiles;
using HealthyTracker.Validation.Auth;
using HealthyTracker.Validation.Extensions;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using Microsoft.OpenApi.Models;
using Serilog;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddHttpClient("NutritionsClient");
builder.Services.AddScoped<INutritionsClient, NutritionsClient>();

var appDataConfig = new AppDataConfig();
var jwtConfig = new JwtConfig();
var emailConfig = new EmailConfig();
var googleConfig = new GoogleConfig();
var nutritionsConfig = new NutritionsConfig();
//var emailSettings = builder.Configuration.GetSection("EmailConfig");

builder.Services.AddConfigs(builder.Configuration, opt =>
    opt.AddConfig<AppDataConfig>(out appDataConfig, configureOptions: config =>
        {
            config.AppDataPath = config.AppDataPath.ToAbsolutePath();
```

```
            config.WebRootPath = builder.Environment.WebRootPath;
        })
        .AddConfig<JwtConfig>(out jwtConfig)
        .AddConfig<EmailConfig>(out emailConfig,
            configureOptions: config => config.TemplatesPath = config.TemplatesPath.ToAbsolutePath())
        .AddConfig<AuthConfig>()
        .AddConfig<GoogleConfig>(out googleConfig)
        .AddConfig<NutritionsConfig>(out nutritionsConfig)
        .AddConfig<CallbackUrisConfig>());

//DbContext
var dbContextLoggerFactory = LoggerFactory.Create(cfg => cfg.AddConsole());

builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection"))
        .UseLoggerFactory(dbContextLoggerFactory));

//Repositories
builder.Services.AddScoped<IDailyRepository, DailyRepository>();
builder.Services.AddScoped<IUserRegistrationRepository, UserRegistrationRepository>();
builder.Services.AddScoped<IProductRepository, ProductRepository>();
builder.Services.AddScoped<IMealRepository, MealRepository>();
builder.Services.AddScoped<INutritionGoalRepository, NutritionGoalRepository>();


//Services
builder.Services.AddScoped<IAuthService, AuthService>();
builder.Services.AddScoped<IEmailConfirmationService, EmailConfirmationService>();
builder.Services.AddScoped<IGoogleAuthService, GoogleAuthService>();
builder.Services.AddScoped<IPasswordService, PasswordService>();
builder.Services.AddScoped<IRefreshTokenService, RefreshTokenService>();
builder.Services.AddScoped<IUserRegistrationService, UserRegistrationService>();
builder.Services.AddScoped<IUserService, UserService>();
builder.Services.AddScoped<IProductService, ProductService>();
builder.Services.AddScoped<INutritionGoalService, NutritionGoalService>();
builder.Services.AddScoped<IDailyService, DailyService>();
builder.Services.AddScoped<IMealService, MealService>();



//Email
var client = new SmtpClient();
```

```
client.Credentials = new NetworkCredential(emailConfig.DefaultEmail, emailConfig.Password);
client.Host = emailConfig.SmtpServer;
client.Port = 587;
client.DeliveryMethod = SmtpDeliveryMethod.Network;
client.EnableSsl = true;
client.UseDefaultCredentials = false;

builder.Services.AddFluentEmail(emailConfig.DefaultEmail)
    .AddSmtpSender(client)
    .AddRazorRenderer(emailConfig.TemplatesPath);

builder.Services.AddScoped<IEmailSender, EmailSender>();

//Utility
builder.Services.AddCors();
builder.Services.AddMemoryCache();

//Mapper
builder.Services.AddAutoMapper(typeof(UserProfile));

//Validators
builder.Services.AddValidatorServiceFromAssemblyContaining<SignInDTOValidator>();

//Logger
var logger = new LoggerConfiguration()
    .MinimumLevel.Information()
    .WriteTo.Map(
        evt => evt.Level,
        (level, wt) =>
            wt.File(
                $@"{appDataConfig.AppDataPath}\{appDataConfig.LogDirectory}\{level}-{DateTime.Today:yyyy-
MM-dd}.log"))
    .CreateLogger();
builder.Logging.AddSerilog(logger, dispose: true);

//Auth
builder.Services.AddIdentity<User, IdentityRole<Guid>>()
    .AddEntityFrameworkStores<ApplicationDbContext>()
    .AddTokenProvider<DataProtectorTokenProvider<User>>(TokenOptions.DefaultProvider);

var tokenValidationParameters = new TokenValidationParameters
{
```

```
    ValidateIssuerSigningKey = true,
    ValidateIssuer = true,
    ValidateAudience = true,
    ValidateLifetime = true,
    IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtConfig.Secret)),
    ValidIssuer = jwtConfig.Issuer,
    ValidAudience = jwtConfig.Audience,
    ClockSkew = jwtConfig.ClockSkew
};
builder.Services.AddSingleton(tokenValidationParameters);

builder.Services.AddAuthentication(options =>
    {
        options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
        options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
        options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
    })
    .AddJwtBearer(options =>
    {
        options.SaveToken = true;
        options.TokenValidationParameters = tokenValidationParameters;
    });

//Swagger
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "HealthyTracker API", Version = "v1" });

    c.AddSecurityDefinition("Bearer",
        new OpenApiSecurityScheme
        {
            Description = "Standard Authorization header using the Bearer scheme. Example: \"bearer {token}\"",
            In = ParameterLocation.Header,
            Name = "Authorization",
            Type = SecuritySchemeType.ApiKey
        });

    c.AddSecurityRequirement(new OpenApiSecurityRequirement
    {
        {
            new OpenApiSecurityScheme
            {
```

```
            Reference = new OpenApiReference { Type = ReferenceType.SecurityScheme, Id = "Bearer" }
        },
        Array.Empty<string>()
    }
  });
});


builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();

var app = builder.Build();

app.MigrateDatabase();

await app.ValidateConfigsAsync();


if (app.Environment.IsDevelopment())
{
  app.UseSwagger();
  app.UseSwaggerUI();
  app.UseDeveloperExceptionPage();
}

app.UseHttpsRedirection();

app.UseStaticFiles();
app.UseCors(x => x.AllowAnyHeader()
  .AllowAnyOrigin()
  .AllowAnyMethod());

app.UseAuthorization();

app.UseAuthentication();

app.MapControllers();

app.Run();
```

Файл NutritionCalculator.tsx

Реалізація функціональної задачі розрахунку виконання норми харчування:

```
import React, { useState, ChangeEvent } from 'react';
import Layout from '../../layout/Layout';
import { Box, Button, Card, FormControl, FormControlLabel, FormLabel, MenuItem, Radio, RadioGroup, Select, SelectChangeEvent, Slider, TextField, Typography } from '@mui/material';
import "./../../../styles/pages/nutritionCalculator.css";
import NutritionInput from './NutritionInput';
import { setFlagsFromString } from 'v8';
import { calculateCaffeineIntake, calculateDailyCalories, calculateFiberIntake, calculateMacronutrients, calculateSaltIntake, calculateWaterIntake } from './calculationFunctions';



const NutritionCalculator = () => {
        const [weight, setWeight] = useState<number>(70); // initial weight
        const [height, setHeight] = useState<number>(170); // initial height
        const [age, setAge] = useState<number>(25); // initial age
        const [bodyFatPercentage, setBodyFatPercentage] = useState<number>(20); // initial body fat percentage
        const [activityLevel, setActivityLevel] = useState<string>('1.375');
        const [goal, setGoal] = useState<string>('maintenance');
        const [gender, setGender] = useState<string>('male');
        const activityLevels = [
                { value: '1.2', label: 'Sedentary lifestyle' },
                { value: '1.375', label: 'Light physical activity one to three days per week' },
                { value: '1.55', label: 'Moderate physical activity six to seven days per week' },
                { value: '1.75', label: 'Intense physical activity daily or twice a day' },
                { value: '1.9', label: 'Intense physical activity two or more times a day' },
        ];
        const goalOptions = [
                { value: 'deficit', label: 'Weight Loss' },
                { value: 'maintenance', label: 'Weight Maintenance' },
                { value: 'surplus', label: 'Weight Gain' },
        ];

        const handleGoalChange = (event: SelectChangeEvent) => {
                setGoal(event.target.value as string);
        };

        const handleSelectChange = (event: SelectChangeEvent) => {
                setActivityLevel(event.target.value as string);
        };

        const handleInputChange = <T extends HTMLInputElement | HTMLTextAreaElement>(
```

```
                event: ChangeEvent<T>,
                setStateFunction: React.Dispatch<React.SetStateAction<number>>
        ) => {
                const newValue = parseFloat(event.target.value);
                if (!isNaN(newValue)) {
                        setStateFunction(newValue);
                }
        };

        const handleGenderChange = (event: ChangeEvent<HTMLInputElement>) => {
                setGender(event.target.value);
        };

        const handleSliderChange = (_event: Event, newValue: number | number[], setStateFunction:
React.Dispatch<React.SetStateAction<number>>) => {
                setStateFunction(newValue as number);
        };

        const dailyCalories = calculateDailyCalories(weight, height, age, bodyFatPercentage, gender,
parseFloat(activityLevel), goal).toFixed(2);
        const macronutrients = calculateMacronutrients(weight, height, age, bodyFatPercentage, gender,
parseFloat(activityLevel), goal);
        const waterIntake = calculateWaterIntake(weight, bodyFatPercentage).toFixed(2);
        const fiberIntake: string = calculateFiberIntake(parseFloat(dailyCalories)).toFixed(2);
        const saltIntake = calculateSaltIntake(weight, bodyFatPercentage).toFixed(2);
        const caffeineNormal = calculateCaffeineIntake(weight, false).toFixed(2);
        const caffeineMax = calculateCaffeineIntake(weight, true).toFixed(2);

        return (
                <Layout>
                        <Box className="nutrition-calculator">
                                <p className="title">Calculate your daily nutrition</p>
                                <Box className="input-main">
                                        <Box className="input-container">
                                                <Box className="input">
                                                        <NutritionInput
                                                                label="Weight"
                                                                metric="kg"
                                                                min={1}
                                                                max={220}
                                                                value={weight}
                                                                setValue={setWeight}
```

```
                                        />
                                    </Box>
                                    <Box className="input">
                                        <NutritionInput
                                            label="Height"
                                            metric="cm"
                                            min={1}
                                            max={220}
                                            value={height}
                                            setValue={setHeight}
                                        />
                                    </Box>
                                </Box>
                                <Box className="input-container">
                                    <Box className="input">
                                        <NutritionInput
                                            label="Age"
                                            metric="years"
                                            min={1}
                                            max={100}
                                            value={age}
                                            setValue={setAge}
                                        />
                                    </Box>
                                    <Box className="input">
                                        <NutritionInput
                                            label="Body fat
percentage"
                                            metric="percent"
                                            min={1}
                                            max={100}

                value={bodyFatPercentage}

                setValue={setBodyFatPercentage}
                                        />
                                    </Box>
                                </Box>
                            </Box>
                            <Box className="main-container">
                                <Box className="main-calculator">
```

```
                                        <Box style={{ display: 'flex', flexDirection:
'column', gap: '10px' }}>
                                            <Select
                                                labelId="activity-level-
label"
                                                id="activity-level"
                                                value={activityLevel}
        onChange={handleSelectChange}
                                            >
                                                <MenuItem value="">
                                                </MenuItem>
        {activityLevels.map((level) => (
                                                    <MenuItem
key={level.value} value={level.value}>
        {level.label}
        </MenuItem>
                                                ))}
                                            </Select>
                                            <Box>
                                                <Select
        labelId="goal-label"
                                                    id="goal"
                                                    value={goal}
        onChange={handleGoalChange}
                                                >
        {goalOptions.map((option) => (
        <MenuItem key={option.value} value={option.value}>
        {option.label}
        </MenuItem>
                                                    ))}
                                                </Select>
                                            </Box>
```

```
                                                    <Box>
                                                        <FormControl>
                                                            <FormLabel
id="gender-label">Gender</FormLabel>
                                                            <RadioGroup
                aria-labelledby="gender-label"
                defaultValue="female"
                name="gender-group"
                value={gender}
                onChange={handleGenderChange}
                                                            >
                <FormControlLabel value="female" control={<Radio />} label="Female" />
                <FormControlLabel value="male" control={<Radio />} label="Male" />
                </RadioGroup>
                                                        </FormControl>
                                                    </Box>
                                                </Box>
                                            </Box>
                                            <Card sx={{ padding: '20px', marginTop: '20px' }}>
                                                <Typography variant="h5">Calculation
Results</Typography>
                                                <Typography>Daily Calories:
{dailyCalories}</Typography>
                                                <Typography variant="h6">Macronutrients
(grams)</Typography>
                                                <Typography>Protein:
{macronutrients.protein.toFixed(2)}</Typography>
                                                <Typography>Fat:
{macronutrients.fat.toFixed(2)}</Typography>
                                                <Typography>Carbohydrates:
{macronutrients.carbohydrates.toFixed(2)}</Typography>
                                                <Typography>Water:
{waterIntake}</Typography>
```

```
                                                    <Typography>Fiber:
{fiberIntake}</Typography>
                                                    <Typography>Salt: {saltIntake}</Typography>
                                                    <Typography>Caffeine Normal(mg):
{caffeineNormal}</Typography>
                                                    <Typography>Caffeine Max(mg):
{caffeineMax}</Typography>
                                                </Card>
                                            </Box>
                                            <Button variant="contained">Save</Button>
                                        </Box>
                                    </Layout >
                                );
};

export default NutritionCalculator;
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРАХУНКУ ТА ТРЕКІНГУ ЗДОРОВОГО ХАРЧУВАННЯ

Програма та методика тестування

КПІ.ІП-1106.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Катерина ЛІЩУК

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Даниїл ГІЖИЦЬКИЙ

Київ – 2025

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є тестування коректнотсі роботи програмного забезпечення, а саме: розрахунок харчування, збереження даних, перевірка виконання норми харчування.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка збереження даних;
- знаходження проблем, помилок і недоліків з метою їх усунення.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;
- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Під час проведення тестування будуть використовуватись наступні допоміжні засоби:

- Postman для тестування API запитів;

Порядок проведення тестування буде наступним:

- статичний аналіз коду;
- тестування на виведення повідомлень про помилку, коли це необхідно;
- функціональне тестування на відповідність функціональним вимогам.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРАХУНКУ ТА ТРЕКІНГУ ЗДОРОВОГО ХАРЧУВАННЯ

Керівництво користувача

КПІ.ІП-1106.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Катерина ЛІЩУК

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Даниїл ГІЖИЦЬКИЙ

Київ – 2025

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«HealthyTracker» – це програмне забезпечення для розрахунку та трекінгу здорового харчування. Також користувачу доступна функція зміни персональної інформації, і зручний інтерфейс.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i5;

– об‘єм ОЗП: 8 Гб;

– підключення до мережі Інтернет зі швидкістю від 5 Мбіт/с.

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i7;

– об‘єм ОЗП: 16 Гб;

– підключення до мережі Інтернет зі швидкістю від 1000 мегабіт.

2.2 Завантаження застосунку

Наразі вебзастосунок розгорнутий за адресою https://healthytracker.com. Для цього необхідно відкрити дане посилання в будь-якому браузері на комп’ютері.

2.3 Перевірка коректної роботи

По завершенню завантаження програмного забезпечення у вікні браузера має відобразитись головна сторінка для роботи із програмним забезпеченням.

3 ВИКОНАННЯ ПРОГРАМИ

При відкритті програмного забезпечення користувач направляється на основну сторінку (рис. 3.1)

Рисунок 3.1 – Головна сторінка застосунку

Далі здійснюється перехід до форми авторизації (рис. 3.2).

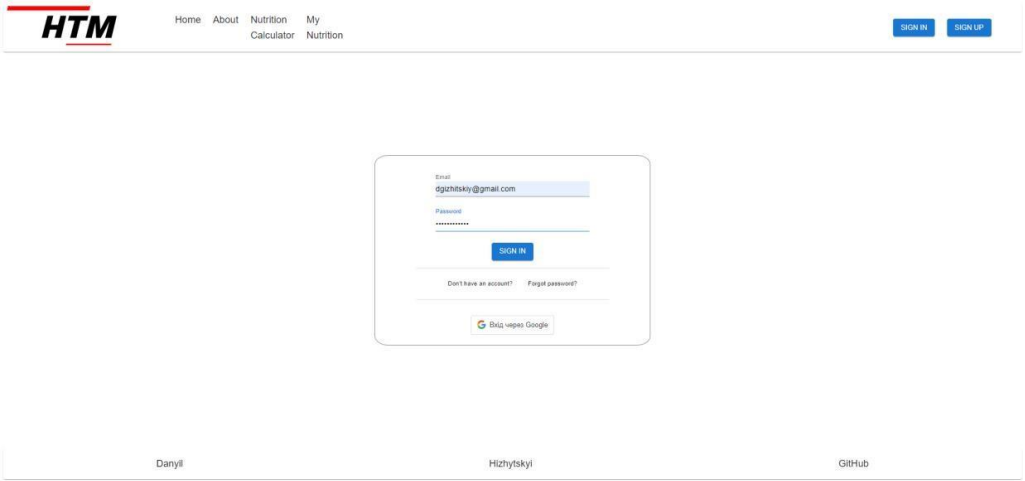


Рисунок 3.2 – Сторінка авторизації користувача

Після успішного введення облікових даних система перенаправляє користувача на його особисту головну сторінку. Інтерфейс завантажується без помилок, і з цього моменту стає доступним основний функціонал веб-застосунку.

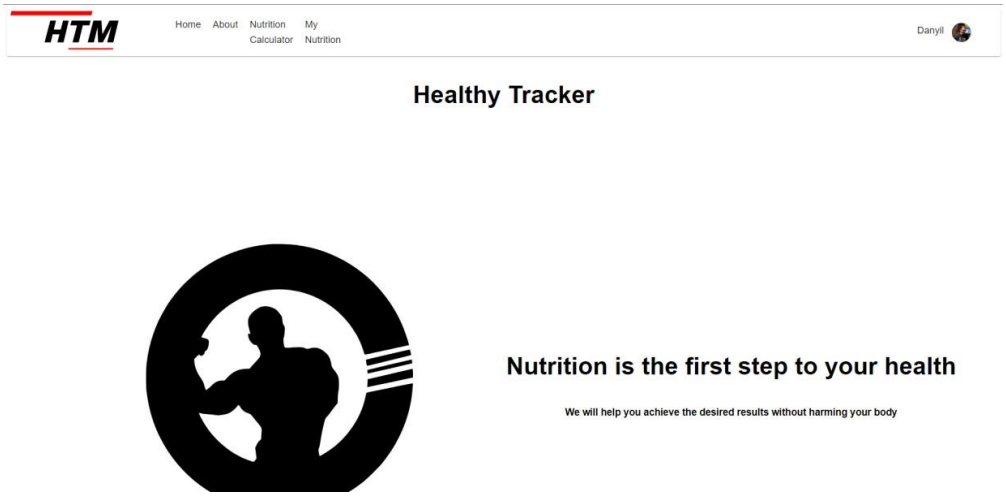


Рисунок 3.3 – Головна сторінка користувача

Після цього користувач переходить на сторінку, призначену для розрахунку добової норми харчування, і вводить відповідні фізіологічні параметри(рис. 3.4).

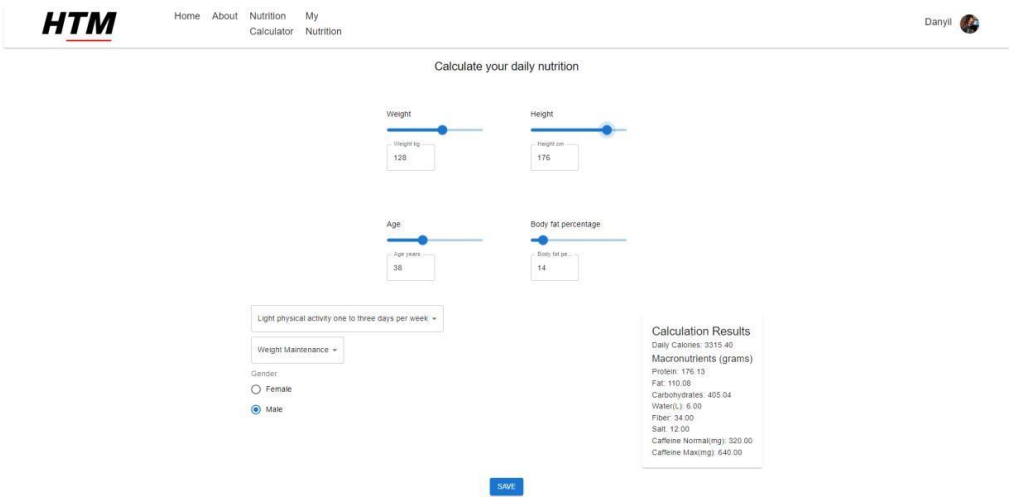


Рисунок 3.4 – Сторінка розрахунку харчування

Після завершення розрахунку перевіряється можливість збереження результату. Користувач натискає кнопку «Зберегти», після чого з’являється повідомлення про успішне збереження даних(рис. 3.5).

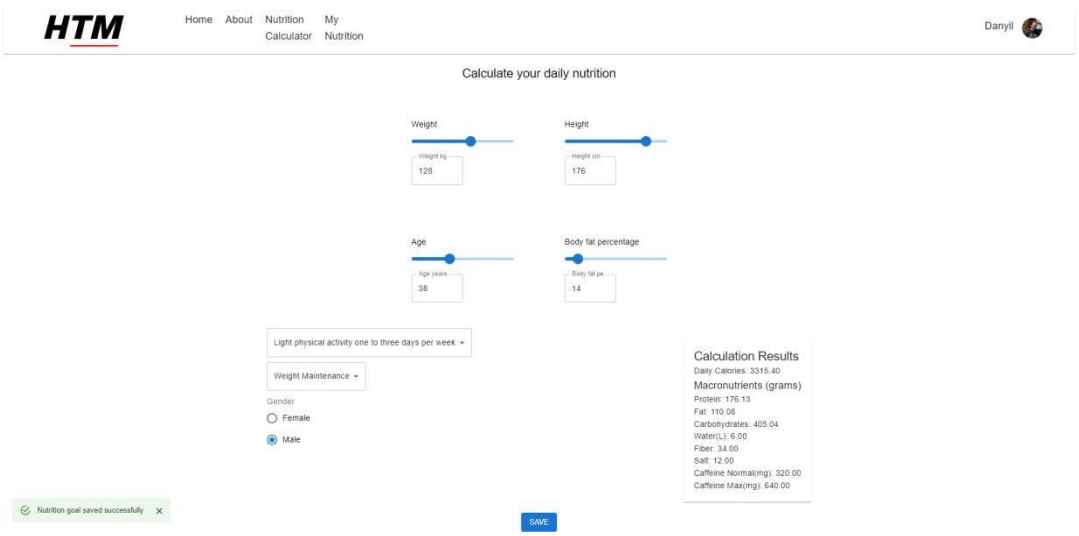


Рисунок 3.5 – Розраховане харчування збережене для користувача.

Далі користувач розгортає відповідний прийом їжі та обирає опцію «Додати продукт». У полях, що з’являються, він зазначає назву продукту та його вагу(рис. 3.6).

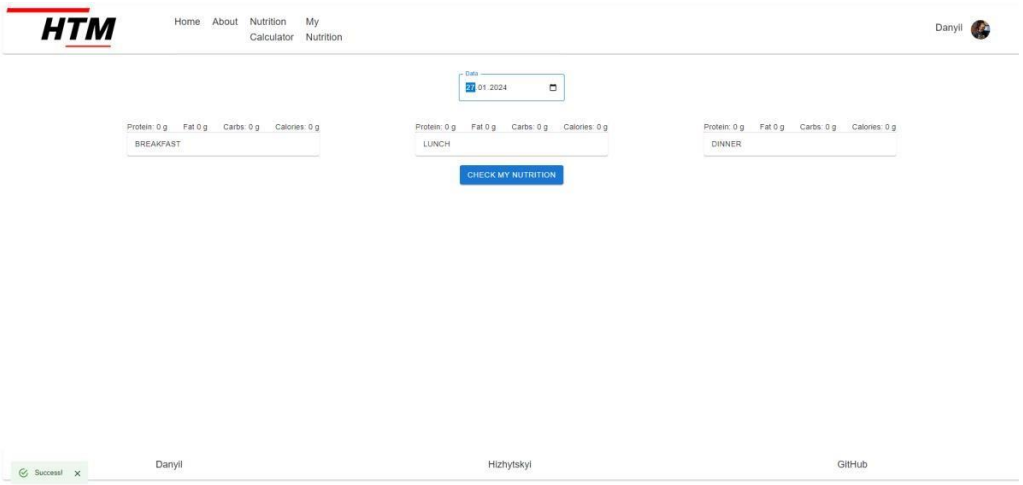


Рисунок 3.6 – Сторінка харчування користувача

Далі користувач розгортає відповідний прийом їжі та обирає опцію «Додати продукт». У полях, що з’являються, він зазначає назву продукту та його вагу(рис. 3.7).

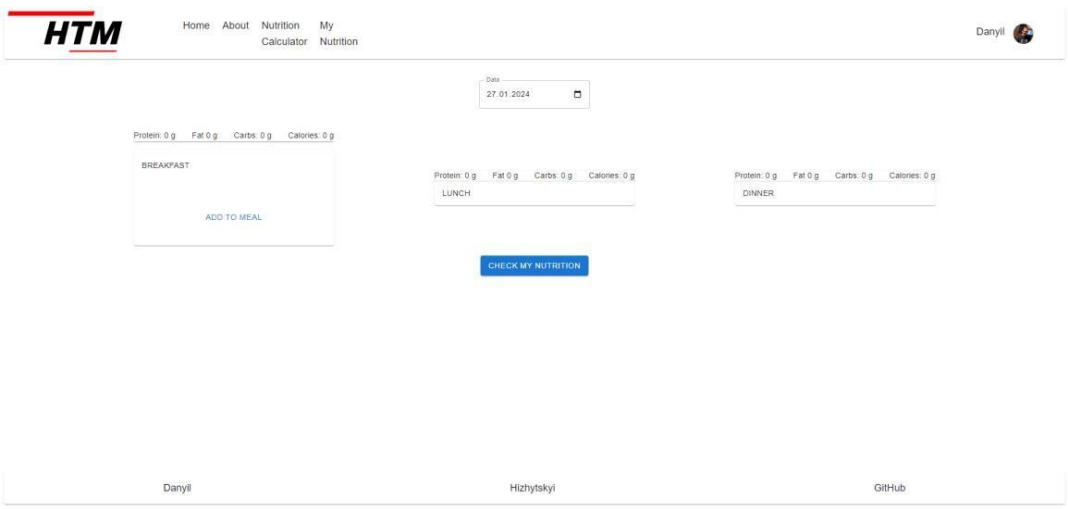


Рисунок 3.7 – Розгорнутий прийом їжі

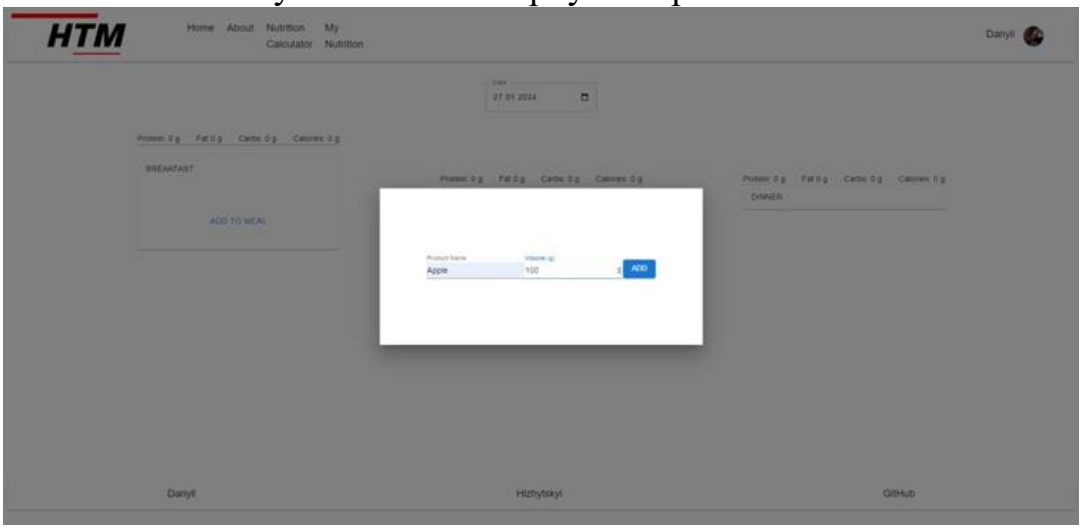


Рисунок 3.8 – Форма вводу продукту

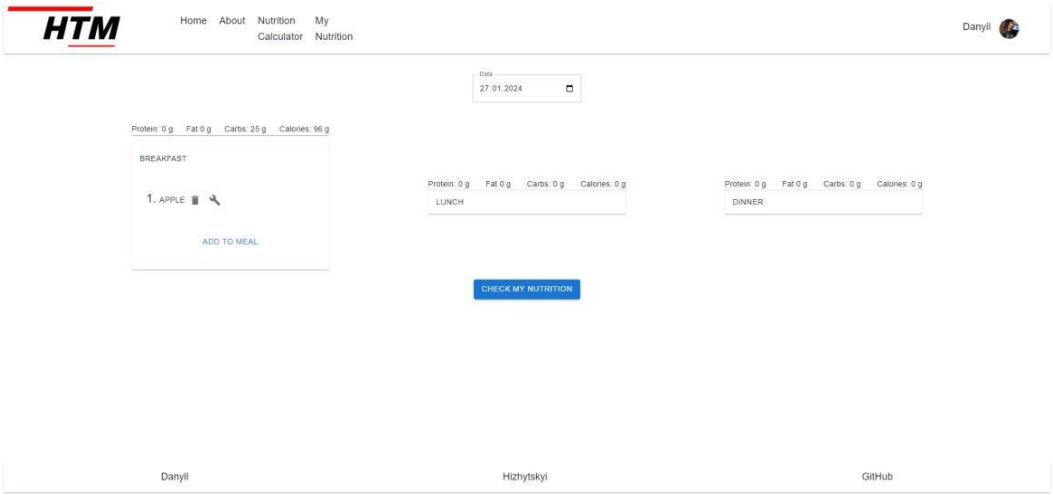


Рисунок 3.9 – Продукт доданий до харчування

Наступним етапом є перевірка виконання встановленої добової норми на основі доданих продуктів. Для цього виконується відповідна перевірка(рис. 3.10).

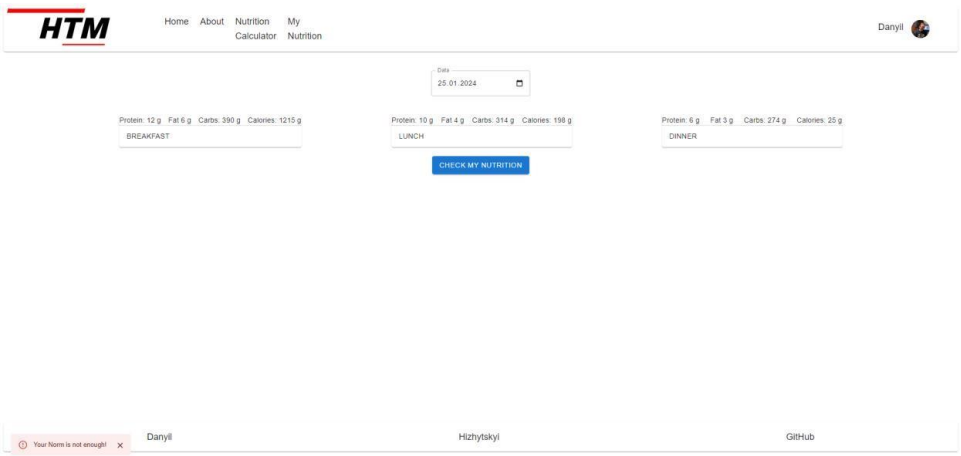


Рисунок 3.10 – Повідомлення що норма не виконана

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРАХУНКУ ТА ТРЕКІНГУ ЗДОРОВОГО ХАРЧУВАННЯ

Графічний матеріал

КПІ.ІП-1106.045440.06.99

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Катерина ЛІЩУК

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Даниїл ГІЖИЦЬКИЙ

Київ – 2025

					КПІ.ІП-1106.045440.06.99.КЕ			
						Лит.	Маса	Масштаб
Зм.	Арк.	№ докум.	Підп.	Дата	Схема вигляду екранних форм			
Розробив		Гіжицький Д.О.				Аркуш	Аркушів	
Перевірив		Ліщук К.І.						
Т. контр.					Програмне забезпечення для розрахунку та трекінгу здорового харчування	КПІ ім.Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-11		
Н. контр.		Ліщук К.І.						
Затвердив		Жаріков Е.В.						

					КПІ.ІП-1106.045440.06.99.ССВ			
						Лит.	Маса	Масштаб
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна варіантів використання			
Розробив		Гіжицький Д.О.				Аркуш	Аркушів	
Перевірив		Ліщук К.І.						
Т. контр.					Програмне забезпечення для розрахунку та трекінгу здорового харчування	КПІ ім.Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-11		
Н. контр.		Ліщук К.І.						
Затвердив		Жаріков Е.В.						

					КПІ.ІП-1106.045440.06.99.СБД			
						Лит.	Маса	Масштаб
Зм.	Арк.	№ докум.	Підп.	Дата	Схема бази даних			
Розробив		Гіжицький Д.О.				Аркуш	Аркушів	
Перевірив		Ліщук К.І.			Програмне забезпечення для розрахунку та трекінгу здорового харчування	КПІ ім.Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-11		
Т. контр.								
Н. контр.		Ліщук К.І.						
Затвердив		Жаріков Е.В.						

					КПІ.ІП-1106.045440.06.99.ССМ			
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна компонентів програмного забезпечення	Лит.	Маса	Масштаб
Розробив		Гіжицький Д.О.						
Перевірив		Ліщук К.І.				Аркуш	Аркушів	
Т. контр.					Програмне забезпечення для розрахунку та трекінгу здорового харчування	КПІ ім.Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-11		
Н. контр.		Ліщук К.І.						
Затвердив		Жаріков Е.В.						