

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: Вебзастосунок для відслідковування макро- та мікронутрієнтів у
раціоні

Виконав студент IV курсу, групи ІП-15
(шифр групи)

Хільчук Артем Валерійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник старший викладач, Марченко О. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант асистент, Шулькевич Т. В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент професор, Корнага Я. І. _____
(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

(підпис) Едуард ЖАРІКОВ
(ім'я прізвище)

“ ____ ” _____ 2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Хільчуку Артему Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Вебзастосунок для відслідковування макро- та мікронутрієнтів у раціоні

керівник проєкту Марченко Олена Іванівна, старший викладач
(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) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.

4) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна діяльності

2) Схема варіантів використання

3) Схема структурна компонентів програмного забезпечення

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	18.03.2025	
3	Постановка та формалізація задачі	20.03.2025	
4	Розробка інформаційного забезпечення	10.04.2025	
5	Алгоритмізація задачі	14.04.2025	
6	Обґрунтування вибору використаних технічних засобів	20.04.2025	
7	Розробка програмного забезпечення	21.04.2025	
8	Налагодження програми	17.05.2025	
9	Виконання графічних документів	20.05.2025	
10	Оформлення пояснювальної записки	25.05.2025	
11	Подання ДП на попередній захист	02.06.2025	
12	Подання ДП рецензенту	10.06.2025	
13	Подання ДП на основний захист	16.06.2025	

Студент

(підпис)

Артем ХІЛЬЧУК

(ініціали, прізвище)

Керівник

(підпис)

Олена МАРЧЕНКО

(ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п'яти розділів, містить 68 таблиць, 40 рисунків та 34 джерел – загалом 111 сторінок.

Дипломний проєкт присвячений розробці вебзастосунку для відслідковування макро- та мікронутрієнтів у раціоні.

Метою розробки є підвищення ефективності процесу моніторингу макро- та мікронутрієнтів у дієті шляхом забезпечення гнучкості налаштування цільових поживних складових, що будуть відслідковуватися, а також впровадження допоміжного функціоналу, орієнтованого на підвищення зручності користування та автоматизацію окремих складових процесу.

У першому розділі сформовано модель предметної області, проведено порівняльний аналіз існуючих рішень, обґрунтовано архітектуру та алгоритмічні рішення й виконано формалізований аналіз вимог із BPMN-моделюванням бізнес-процесу.

У другому розділі здійснено аналіз вимог до ПЗ, сформульовано функціональні та нефункціональні, обґрунтовано конфігурацію технічних ресурсів та економічні оцінки проєкту.

У третьому розділі виконано проєктування архітектури програмного засобу та моделювання даних, визначено алгоритмічні підходи й заходи дотримання безпеки застосунку, а також реалізовано програмне забезпечення відповідно до технічного завдання.

У четвертому розділі здійснено статичний аналіз якості ПЗ, виконано комплекс мануальних тестувань основного функціоналу й наведено контрольний тестовий сценарій типового циклу користування застосунком.

У п'ятому розділі сформульовано покрокову інструкцію з розгортання застосунку на хмарних платформах, а також визначено основні заходи для підтримки та оновлення ПЗ.

КЛЮЧОВІ СЛОВА: ASP.NET CORE, EF CORE, ВЕБЗАСТОСУНОК, POSTGRESQL, QDRANT, REDIS, RABBITMQ, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, МІКРОСЕРВІС, МОНІТОРИНГ РАЦІОНУ.

ABSTRACT

The explanatory note of the diploma project consists of five sections, contains 68 tables, 40 figures and 34 sources – in total 111 pages.

The purpose of the diploma project is to enhance the efficiency of monitoring macro- and micronutrients in the diet by providing flexible configuration of target nutrients to be tracked, as well as introducing auxiliary functionality aimed at improving usability and automating individual process components.

In the first chapter, a domain model was established, a comparative analysis of existing solutions was conducted, the architecture and algorithmic approaches were justified, and a formal requirements analysis with BPMN business-process modeling was performed.

In the second chapter, the software requirements were analyzed, functional and non-functional requirements were formulated, and the technical-resource configuration and economic project estimates were substantiated.

In the third chapter, the software architecture was designed and data was modeled, algorithmic approaches and security measures were defined, and the software was implemented in accordance with the technical specification.

In the fourth chapter, a static quality analysis of the software was carried out, a set of manual tests of the core functionality was executed, and a control test scenario illustrating a typical user cycle was presented.

In the fifth chapter a step-by-step deployment guide for the application on cloud platforms was formulated, and the main measures for software maintenance and updates were specified.

KEYWORDS: ASP NET CORE, EF CORE, WEB APPLICATION, POSTGRESQL, QDRANT, REDIS, RABBITMQ, CLIENT-SERVER ARCHITECTURE, MICROSERVICE, DIET MONITORING

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ВЕБЗАСТОСУНОК ДЛЯ ВІДСЛІДКОВУВАННЯ МАКРО- ТА
МІКРОНУТРІЄНТІВ У РАЦІОНІ**

Технічне завдання

КПІ.ПІ-1527.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена МАРЧЕНКО

Нормоконтроль:

_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:

_____ Артем ХІЛЬЧУК

Київ – 2025

ЗМІСТ

1	НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ	3
2	ПІДСТАВА ДЛЯ РОЗРОБКИ.....	4
3	ПРИЗНАЧЕННЯ РОЗРОБКИ	5
4	ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	6
4.1	Вимоги до функціональних характеристик	6
4.1.1	Користувацького інтерфейсу:	6
4.1.2	Облік користувачів: користувач, адміністратор	22
4.2	Вимоги до надійності	24
4.3	Умови експлуатації.....	24
4.3.1	Вид обслуговування	24
4.3.2	Обслуговуючий персонал.....	24
4.4	Вимоги до складу і параметрів технічних засобів	24
4.5	Вимоги до інформаційної та програмної сумісності	25
4.5.1	Вимоги до вхідних даних	25
4.5.2	Вимоги до вихідних даних	25
4.5.3	Вимоги до мови розробки.....	25
4.5.4	Вимоги до середовища розробки.....	25
4.5.5	Вимоги до представленню вихідних кодів	25
4.6	Вимоги до маркування та пакування	26
4.7	Вимоги до транспортування та зберігання	26
4.8	Спеціальні вимоги.....	26
5	ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	27
5.1	Попередній склад програмної документації	27
5.2	Спеціальні вимоги до програмної документації.....	27
6	СТАДІЇ І ЕТАПИ РОЗРОБКИ	28
7	ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	29

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: вебзастосунок для відслідковування макро- та мікронутрієнтів у раціоні.

Галузь застосування:

Наведене технічне завдання поширюється на розробку вебзастосунку для відслідковування макро- та мікронутрієнтів у раціоні, котра використовується для ведення та планування записів споживання з можливістю перегляду каталогу харчових продуктів, що розширена пошуковою підтримкою як за назвою, так і за штрих-кодом, створення та управління рецептами, налаштування відстежуваних нутрієнтів, автоматичного формування персоналізованого раціону на основі поставлених нутріційних цілей, а також перегляду звітності й статистики споживання за обрані періоди та призначена для осіб, які прагнуть підтримувати раціональне харчування, дотримуються специфічних медично обґрунтованих дієтичних обмежень або орієнтовані на досягнення високо персоналізованих нутріційних цілей.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки вебзастосунку для відслідковування макро- та мікронутрієнтів у раціоні 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 ведення та планування харчових записів, а також перегляду статистики споживання налаштовуваного набору макро- та мікронутрієнтів.

Метою розробки є розширення можливостей персоналізації відслідковування споживання нутрієнтів та спрощення мануальної послідовності дій, пов'язаних із введенням, редагуванням й аналізом даних харчового щоденника за рахунок цифровізації та часткової автоматизації.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Користувацького інтерфейсу:

- домашня сторінка (рисунок 4.1);
- реєстрації користувача (рисунок 4.2);
- входу в обліковий запис (рисунок 4.3);
- перегляду профілю користувача (рисунок 4.4);
- редагування профілю користувача (рисунок 4.5);
- відновлення паролю (рисунок 4.6);
- перегляду харчового щоденника (рисунок 4.7);
- додавання записів споживання їжі (рисунок 4.8);
- редагування записів споживання їжі (рисунок 4.9);
- додавання записів споживання рецепта (рисунок 4.10);
- редагування записів споживання рецепта (рисунок 4.11);
- перегляду приватних рецептів (рисунок 4.12);
- перегляду приватного рецепту (рисунок 4.13);
- створення приватних рецептів (рисунок 4.14);
- редагування приватних рецептів (рисунок 4.15);
- перегляду публічних рецептів (рисунок 4.16);
- перегляду публічного рецепту (рисунок 4.17);
- створення публічних рецептів (рисунок 4.18);
- редагування публічних рецептів (рисунок 4.19);
- отримання рекомендованого раціону (рисунок 4.20);
- перегляду активних планувань (рисунок 4.21);
- створення нових планувань (рисунок 4.22);
- перегляду усієї їжі (рисунок 4.23);

- перегляд їжі (рисунок 4.24);
- додавання їжі (рисунок 4.25);
- перегляд статистики споживання (рисунок 4.26);
- експорт статистики споживання у файловому форматі (рисунок 4.27).

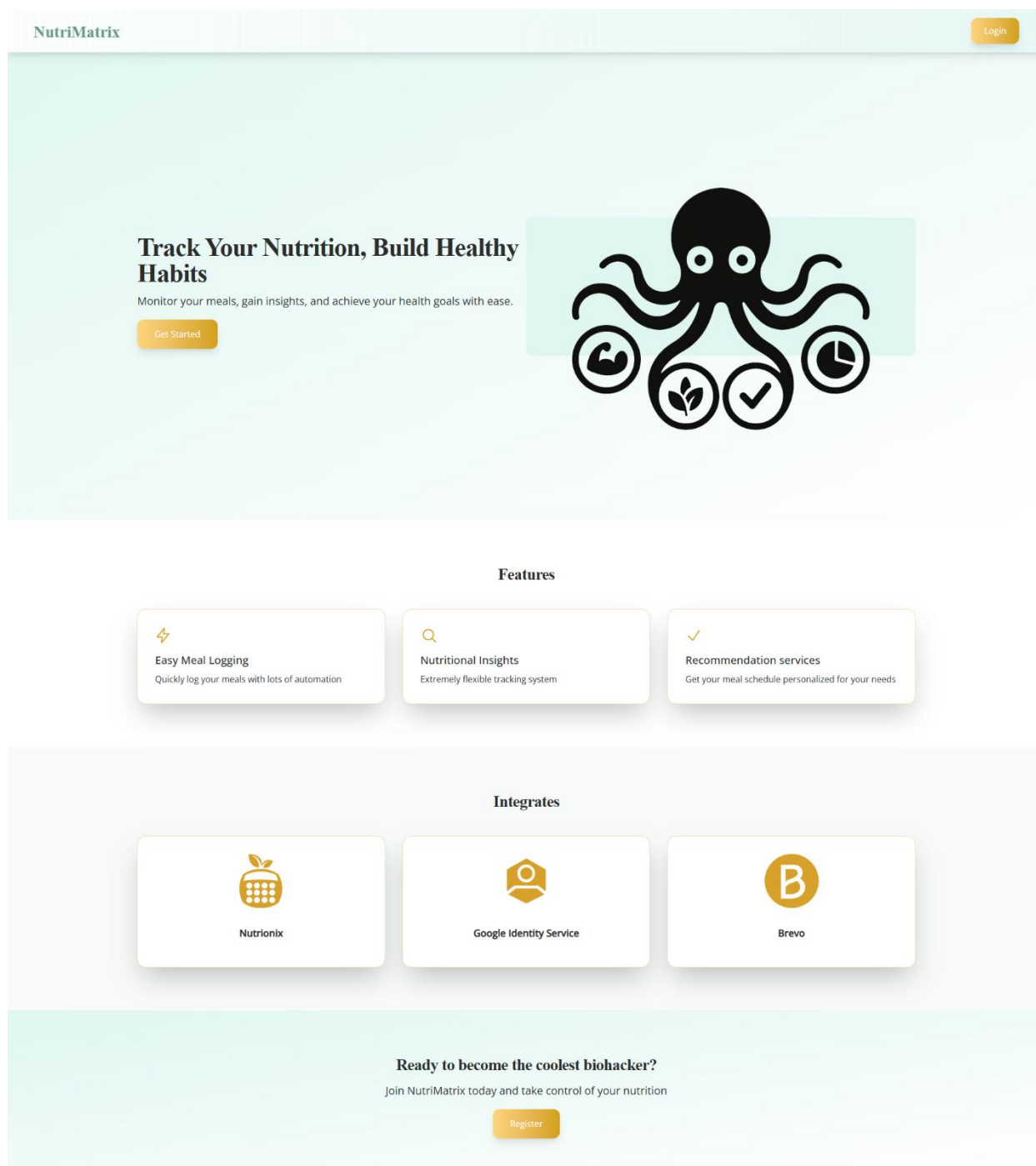


Рисунок 4.1 – Домашня сторінка

NutriMatrix Login

Register

Create your account to start your journey into NutriMatrix!

Enter your name

Enter your email address

Enter your password

Register →

or register with



Already have an account? [Login here!](#)



Рисунок 4.2 – Реєстрація користувача

NutriMatrix Login

Log In

Log in to setup your journey into world of food tracking!

Enter your email address

Enter your password

[Forgot password](#) Login →

or log in with



Don't have an account? [Register now!](#)



Рисунок 4.3 – Вхідження в обліковий запис

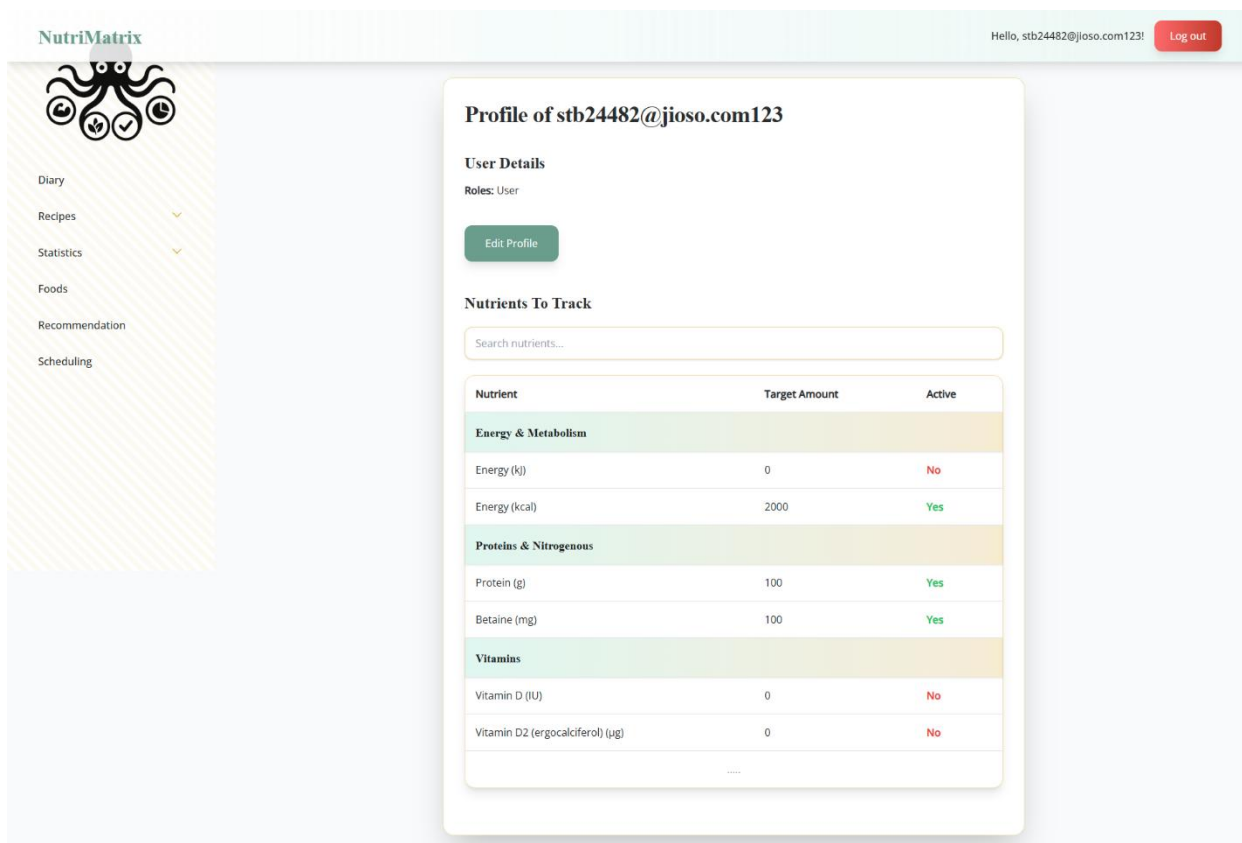


Рисунок 4.4 – Перегляд профілю користувача

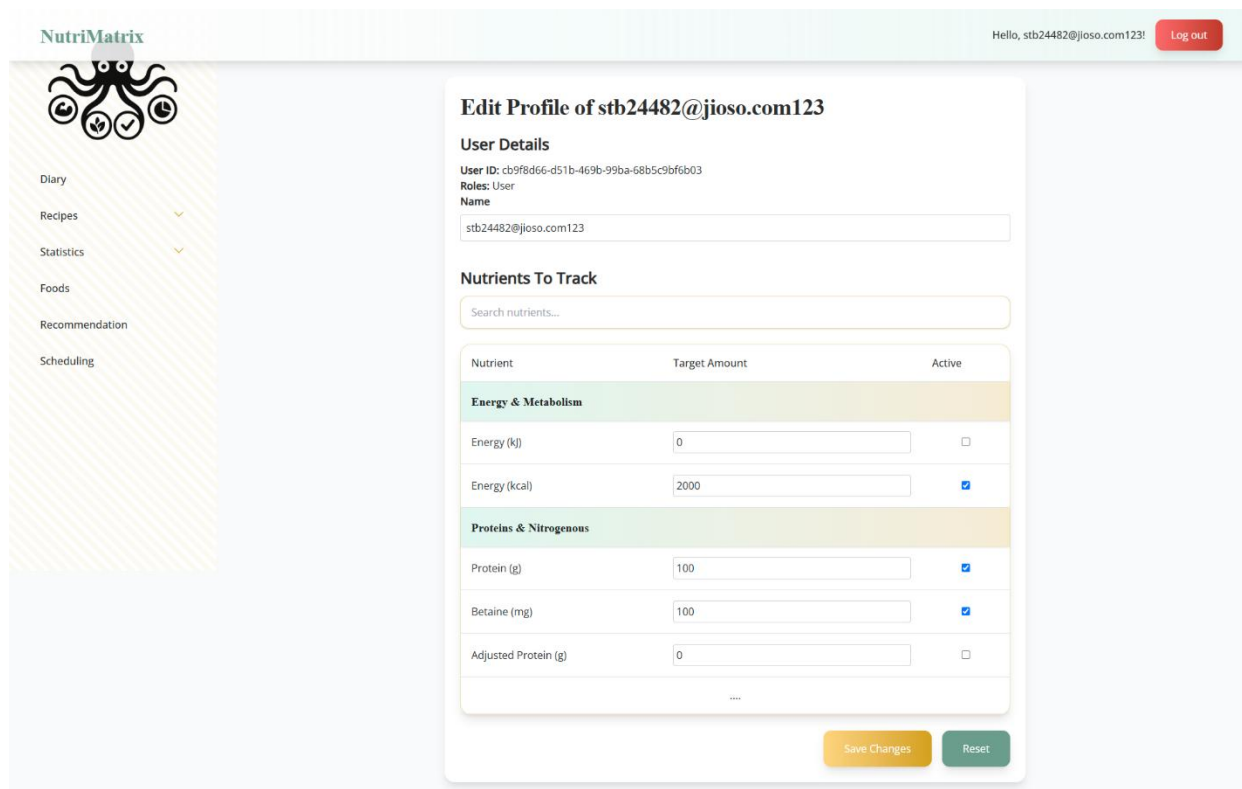


Рисунок 4.5 – Редагування профілю користувача

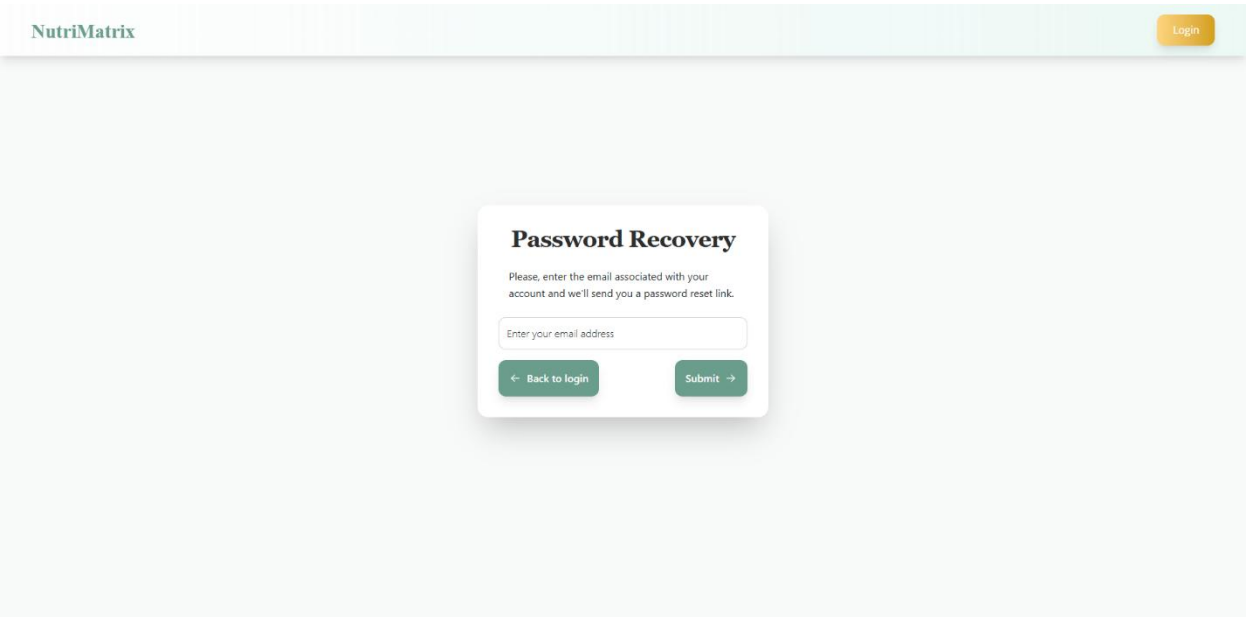


Рисунок 4.6 – Відновлення паролю

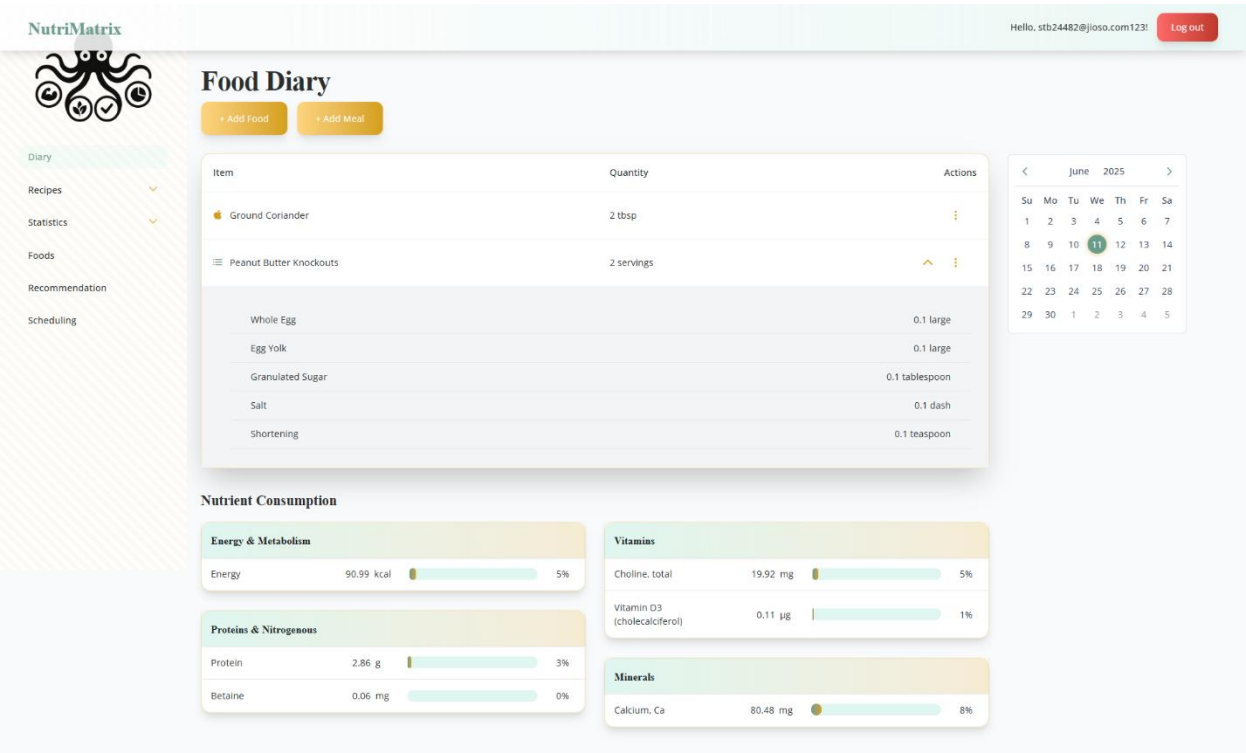


Рисунок 4.7 – Перегляд харчового щоденника

Add Food Entry



Ground Coriander

Amount

1

Measure

teaspoon — 2.17 g

Add to Dashboard



Nutritional Details

- **Calcium, Ca** : 15.36 mg
- **Energy** : 6.46 kcal
- **Protein** : 0.27 g
- **Betaine** : 0 mg
- **Vitamin D3 (cholecalciferol)** : 0 µg
- **Choline, total** : 0 mg

Рисунок 4.8 – Додавання запису споживання їжі



Edit Food Entry

ground cori



Ground Coriander

Ground Coriander

Amount

2

Measure

tbsp — 5 g



Add to
Dashboard



Nutritional Details

- **Calcium, Ca** : 70.79 mg
- **Energy** : 29.75 kcal
- **Protein** : 1.24 g
- **Betaine** : 0 mg
- **Vitamin D3 (cholecalciferol)** : 0 µg
- **Choline, total** : 0 mg

Рисунок 4.9 – Редагування запису споживання їжі



Add Meal Entry

peanut

Peanut Butter Knockouts

Peanut Butter Knockouts

Servings

2

Add to Dashboard

Nutritional Details (Total for 2.0 Servings)

- **Calcium, Ca** : 9.69 mg
- **Energy** : 61.23 kcal
- **Protein** : 1.62 g
- **Betaine** : 0.06 mg
- **Vitamin D3 (cholecalciferol)** : 0.11 µg
- **Choline, total** : 19.92 mg

Рисунок 4.10 – Додавання запису споживання рецепту



Edit Meal Entry

peanut

Peanut Butter Knockouts

Servings

20

Add to Dashboard

Nutritional Details (Total for 20.0 Servings)

- Calcium, Ca : 96.88 mg
- Energy : 612.33 kcal
- Protein : 16.24 g
- Betaine : 0.64 mg
- Vitamin D3 (cholecalciferol) : 1.07 µg
- Choline, total : 199.22 mg

Рисунок 4.11 – Редагування запису споживання рецепту

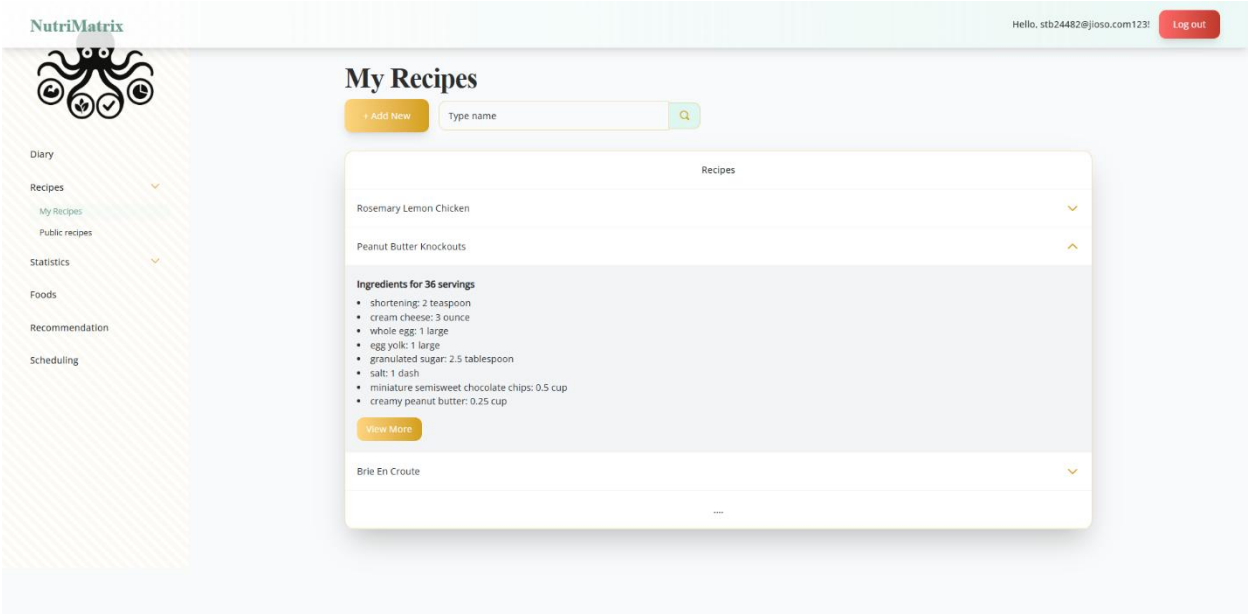


Рисунок 4.12 – Перегляд власних рецептів

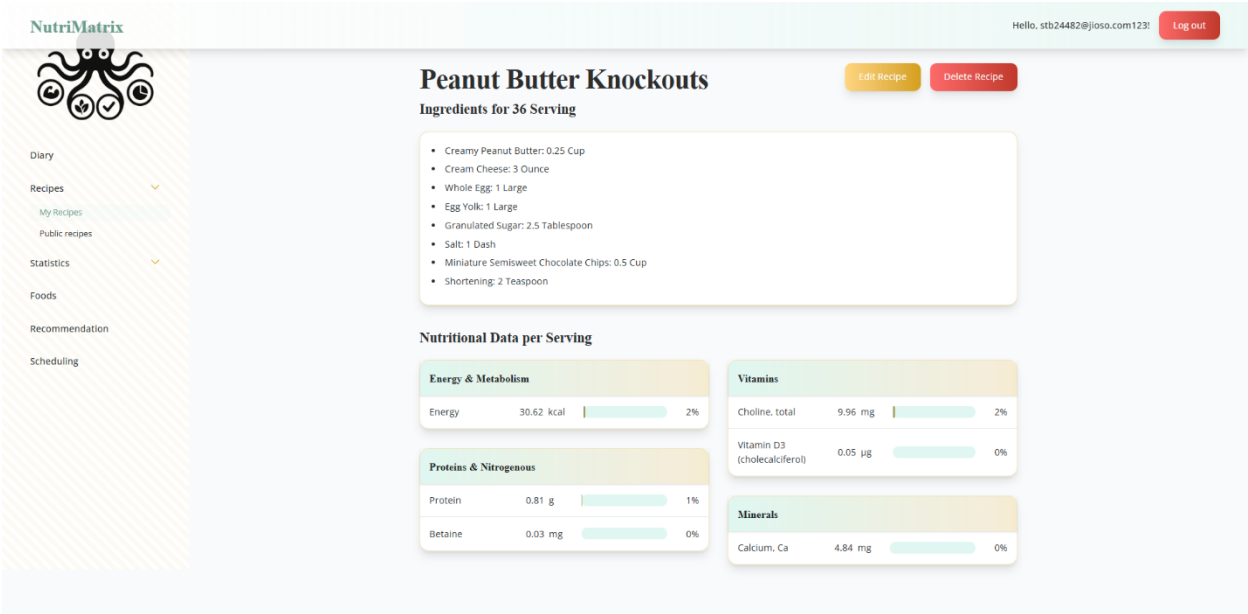


Рисунок 4.13 – Перегляд власного рецепту

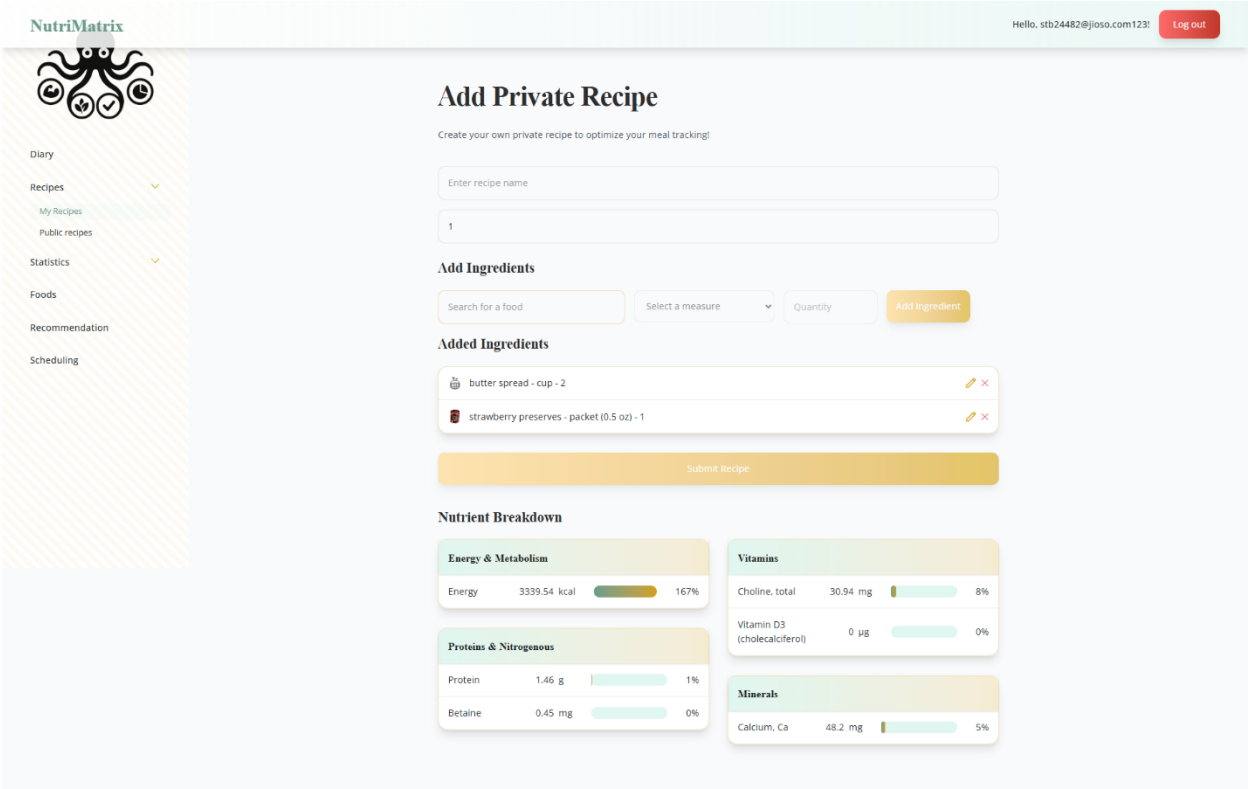


Рисунок 4.14 – Додавання власного рецепту

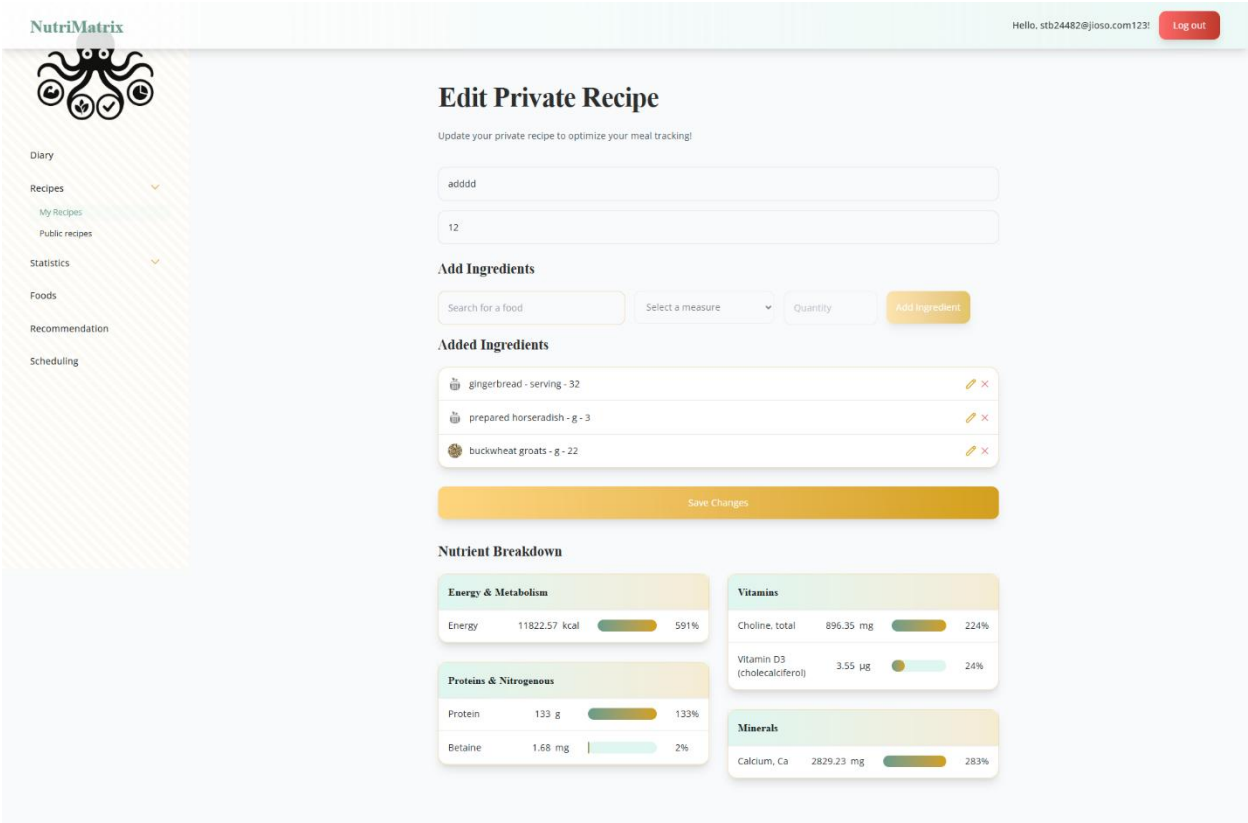


Рисунок 4.15 – Редагування власного рецепту

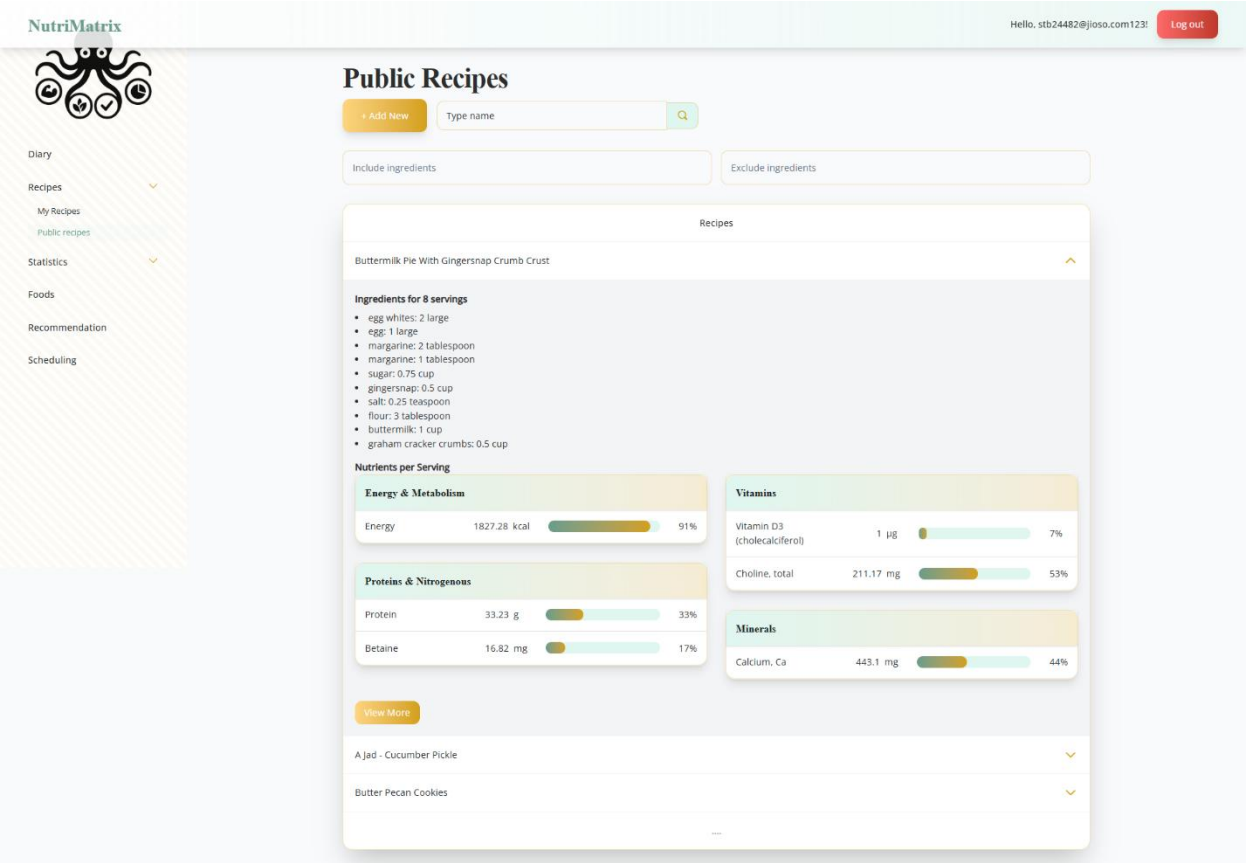


Рисунок 4.16 – Перегляд публічних рецептів

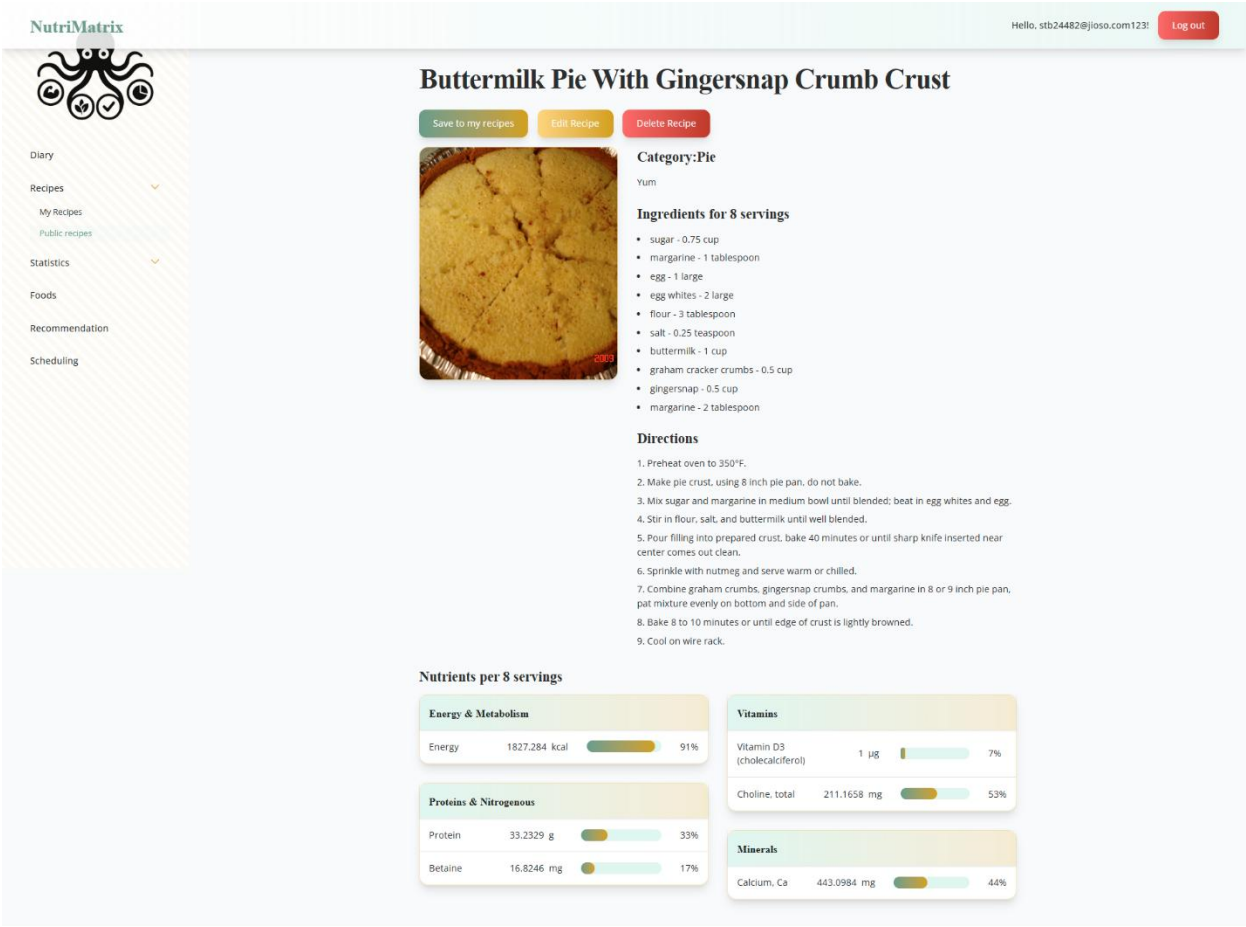


Рисунок 4.17 – Перегляд публічного рецепту

NutriMatrix



Diary

Recipes

My Recipes

Public recipes

Statistics

Foods

Recommendation

Scheduling

Create Public Recipe

Enter recipe title

Enter category

1

Enter description

Directions

Enter step 1

Add Step

Enter photo URL (optional)

Add Ingredients Manually

Search for a food

Select a measure

Quantity

Add Ingredient

Show Automatic Ingredient Parser

Create Recipe

Nutrient Breakdown

No nutrient data available.

Рисунок 4.18 – Додавання публічного рецепту

NutriMatrix



Diary

Recipes

My Recipes

Public recipes

Statistics

Foods

Recommendation

Scheduling

Edit Public Recipe

Breakfast Burritos

Breakfast

4

Breakfast Burritos

Directions

Warm tortillas in microwave on medium for about 30 seconds until soft enough to roll up without cracking.

Place about 1/4 cup egg mixture in tortilla and roll up.

Add Step

Enter photo URL (optional)

Add Ingredients

Search for a food

Select a measure

Quantity

Add Ingredient

Added Ingredients

sausage - lb - 1

flour tortillas - tortilla (approx 7-8" dia) - 10

Update Recipe

Nutrient Breakdown

Energy & Metabolism

Energy	2,994.13	kcal
Energy	12,538.15	kJ

Proteins & Nitrogenous

Protein	96.97	g
Betaine	8.16	mg
Adjusted Protein	0.0	g

Рисунок 4.19 – Редагування публічного рецепту

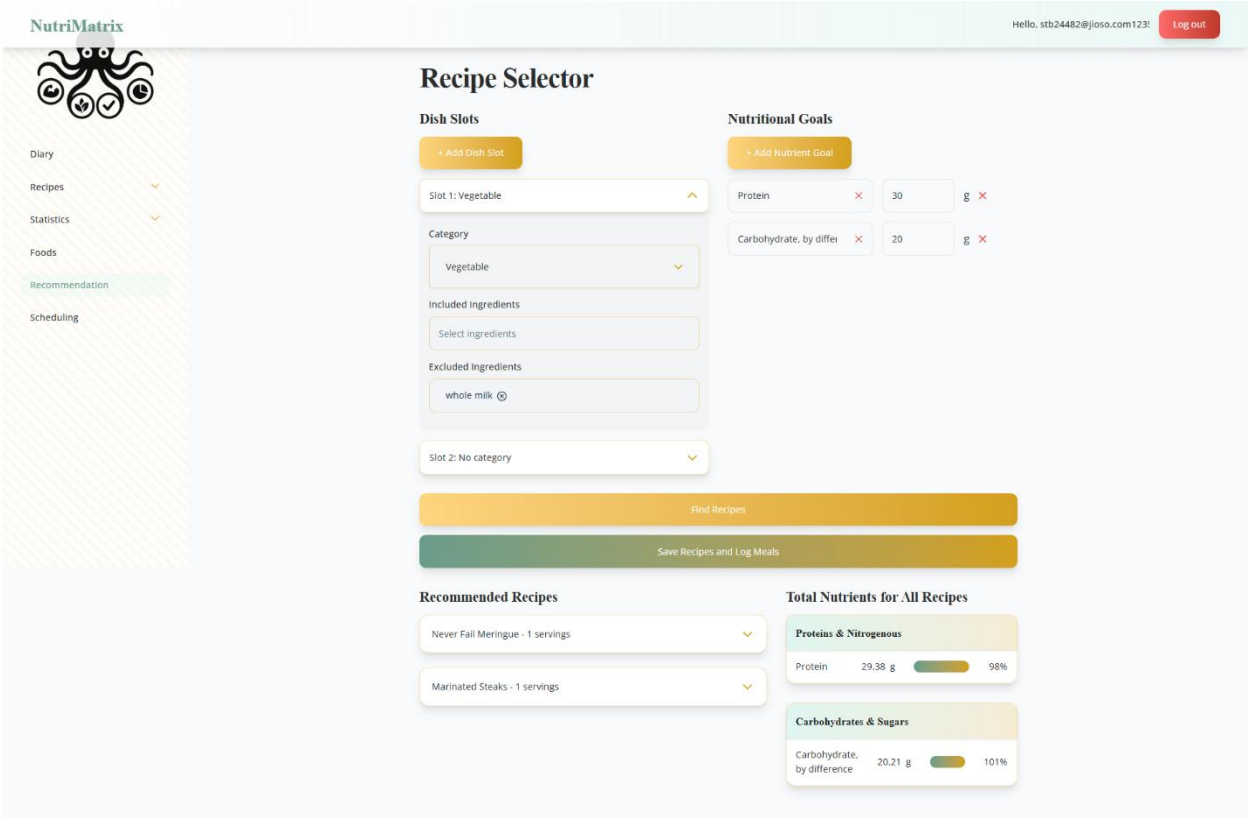


Рисунок 4.20 – Отримання рекомендованого раціону

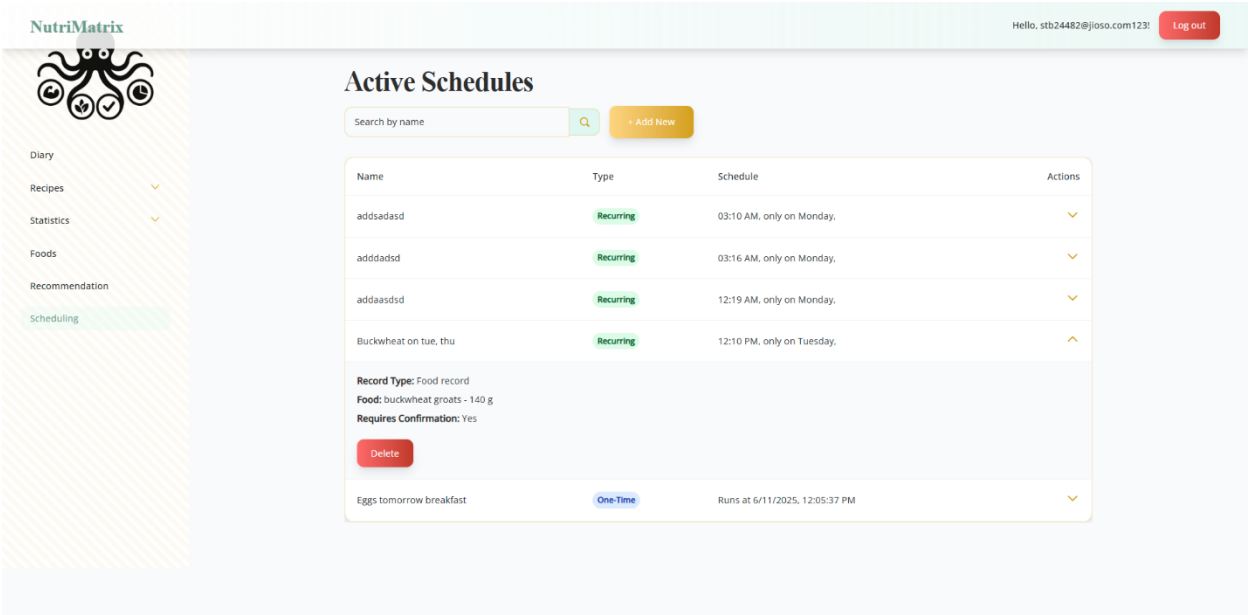


Рисунок 4.21 – Перегляд активних планувань

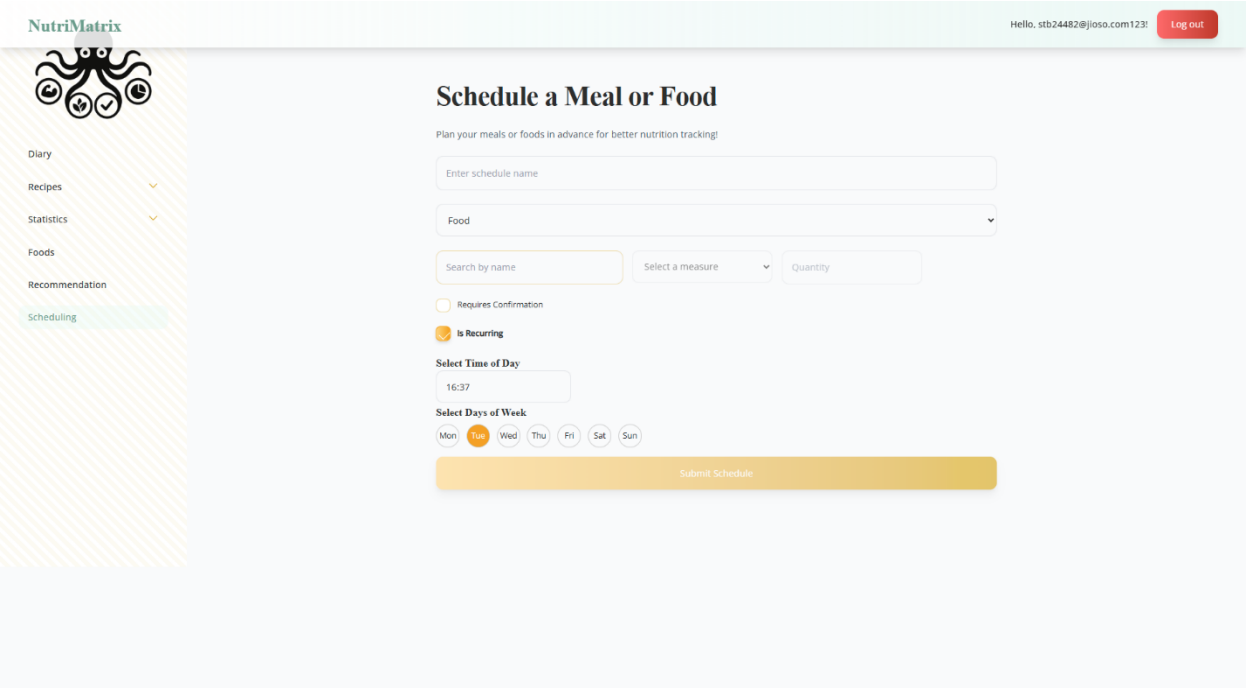


Рисунок 4.22 – Створення планування

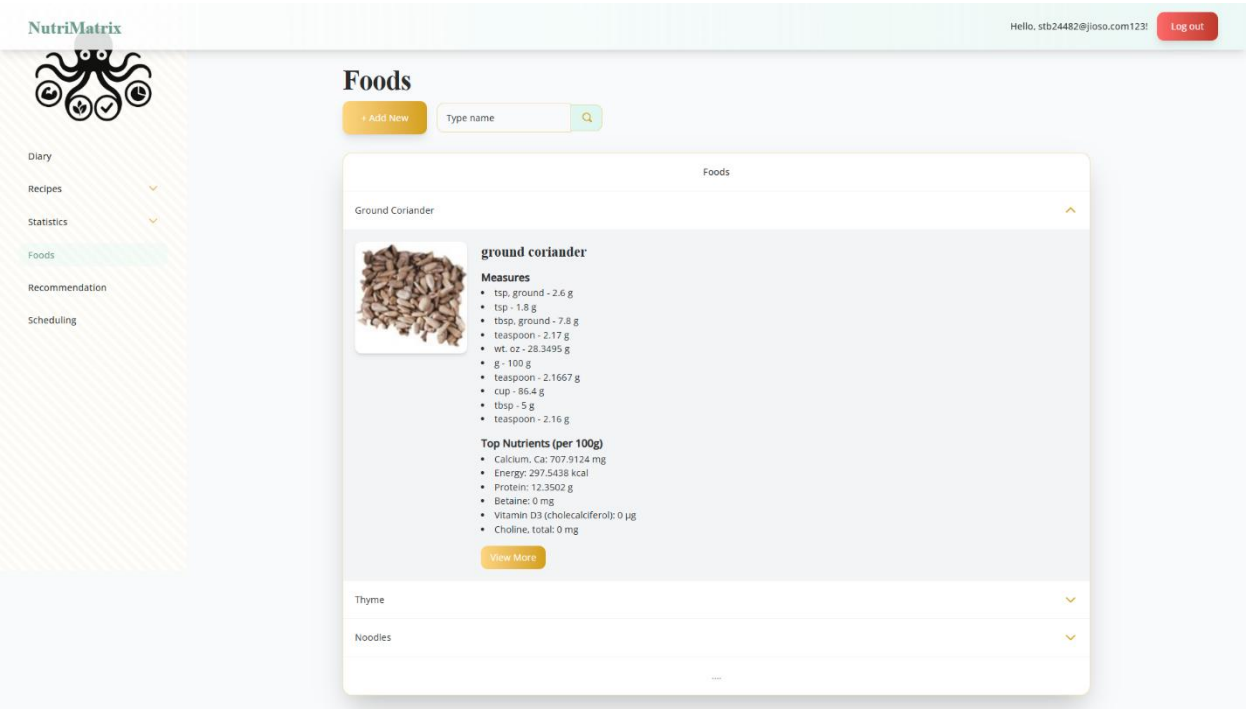


Рисунок 4.23 – Перегляд усієї їжі

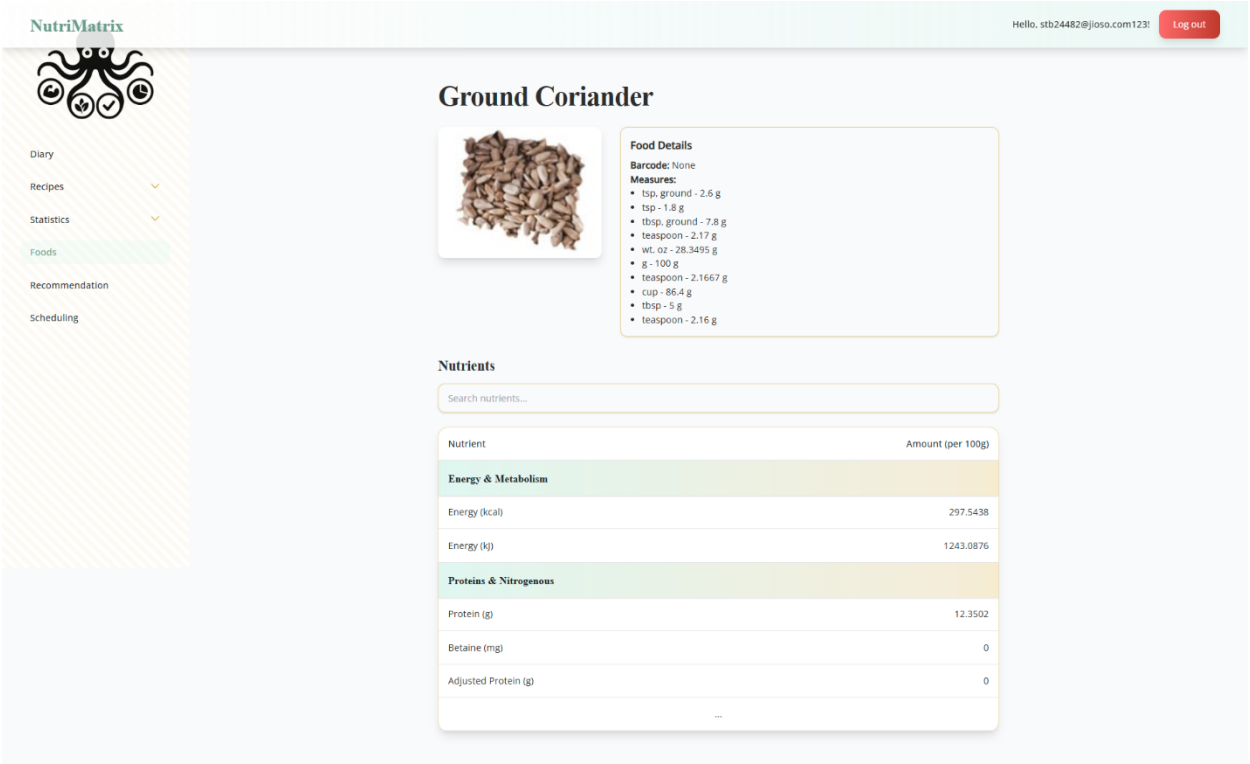


Рисунок 4.24 – Перегляд їжі

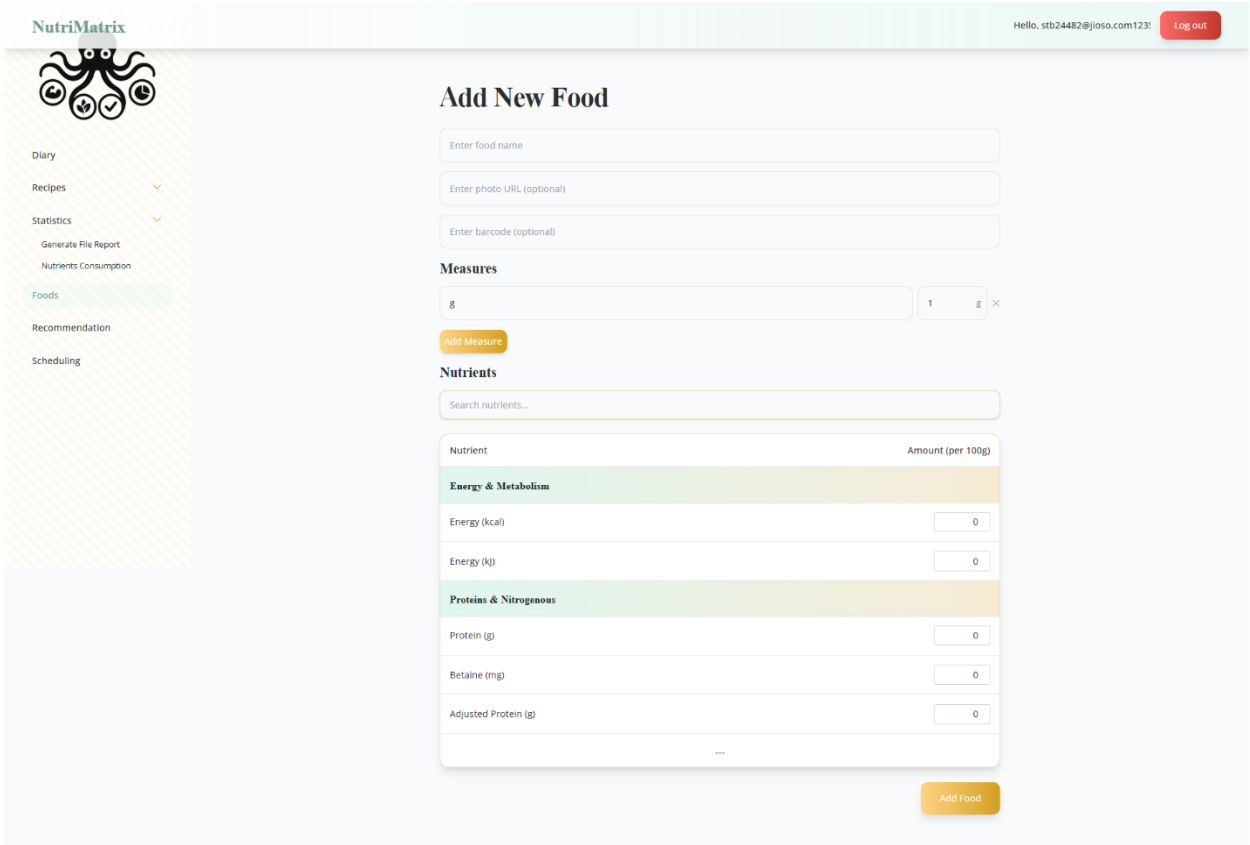


Рисунок 4.25 – Додавання їжі

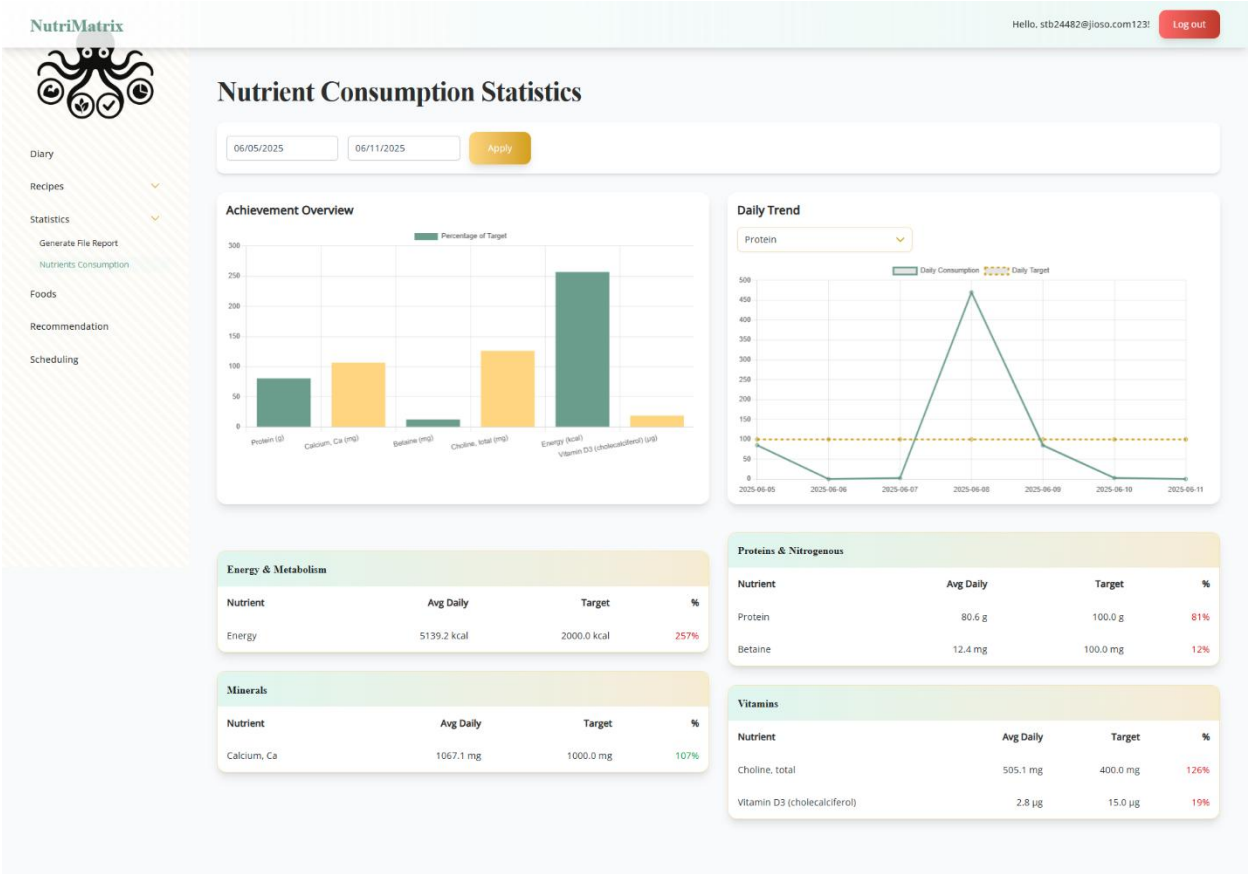


Рисунок 4.26 – Перегляд статистики споживання нутрієнтів

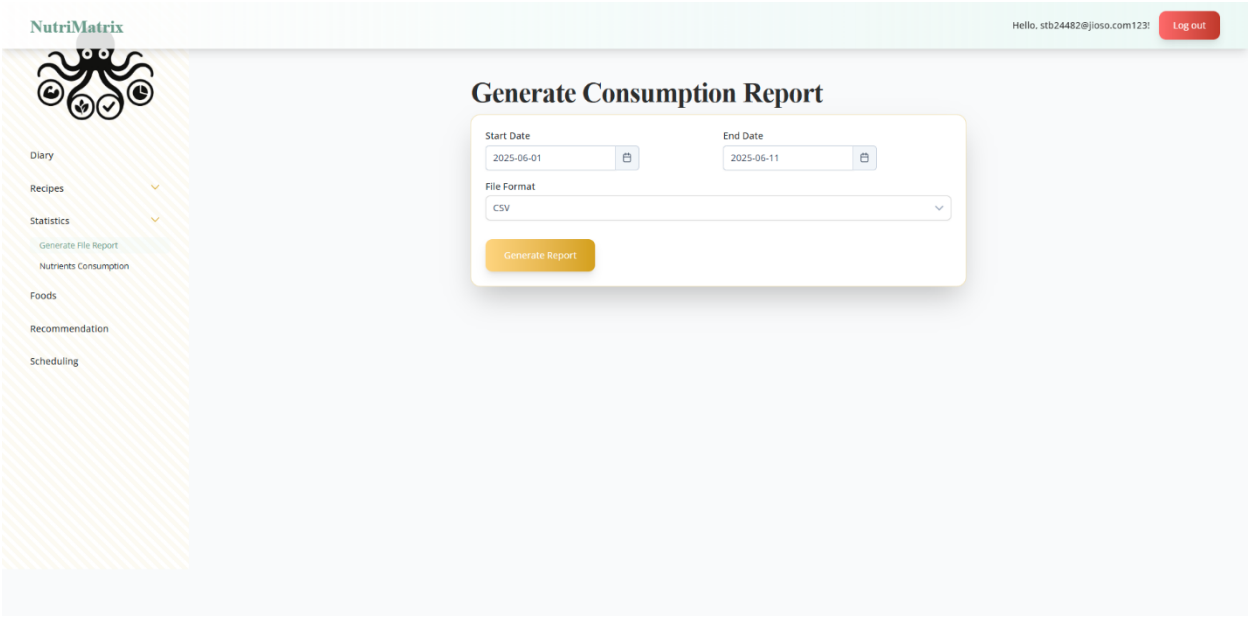


Рисунок 4.27 – Отримання файлового звіту споживання

4.1.2 Облік користувачів: користувач, адміністратор

4.1.2.1 Для користувача:

- реєстрація в системі;
- автентифікація в системі;

- перегляд власних рецептів;
- створення власних рецептів;
- редагування власних рецептів;
- видалення власних рецептів;
- перегляд публічних рецептів;
- перегляд коротких відомостей щодо їжі;
- збереження публічних рецептів у колекцію власних;
- створення записів споживання їжі;
- редагування записів споживання їжі;
- видалення записів споживання їжі;
- підтвердження записів споживання їжі;
- створення записів споживання рецепту;
- редагування записів споживання рецепту;
- видалення записів споживання рецепту;
- підтвердження записів споживання рецепту;
- перегляд профілю користувача;
- редагування профілю користувача;
- перегляд плануваль;
- видалення плануваль;
- створення плануваль;
- автоматичне формування рекомендованого раціону для поставлених нутріційних цілей та обмежень страв;
- перегляд статистики споживання для заданого проміжку часу;
- експорт файлового звіту споживання за заданий період.

4.1.2.2 Для адміністратора:

- створення публічних рецептів;
- редагування публічних рецептів;
- перегляд повних відомостей їжі;
- додавання їжі.

4.1.2.3 Додаткові вимоги:

- автоматичне додавання записів споживання в щоденник харчування користувача у відповідності до планувань;
- хешування збережених паролів користувачів.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних.

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

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення серверної частини повинно функціонувати на серверах з операційною системою Ubuntu.

Мінімальна конфігурація технічних засобів:

- кількість ядер: 4;
- об'єм ОЗП: 8 Гб;
- підключення до мережі Інтернет зі швидкістю від 20 мегабіт.

Рекомендована конфігурація технічних засобів:

- кількість ядер: 12;
- об'єм ОЗП: 16 Гб;
- підключення до мережі Інтернет зі швидкістю від 100 мегабіт.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows XP, Windows NT і т.д.) або Unix та бути сумісним з браузерами Chrome 119.0.6045.200(1920x1080) та Opera Stable 105.0.4970.29(1920x1080).

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: запити за протоколом HTTPS з використанням формату даних JSON.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступних форматах: JSON, HTML, JS.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування C#. Реалізацію контролерів буде проведено з використанням технологічного середовища ASP.Net Core. Взаємодія із базою даних проводитиметься за підтримки Entity Framework Core. Автентифікацію та авторизацію користувачів буде здійснено за допомогою Identity ASP.Net core. У якості БД буде використано PostgreSQL. Клієнтську частину системи буде реалізовано за допомогою технологічного середовища Angular та мови програмування TypeScript.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі середовищ Visual Studio 2022 версії 17.8.22 та WebStorm 2023.2.5 версії 232.10227.9

4.5.5 Вимоги до представлення вихідних кодів

Вихідний код програми має бути представлений у вигляді у вигляді файлів з розширеннями .html, .css, .ts, .cs, .csproj, .sln та .json.

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 відсутні.

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 діяльності;
- схема структурна варіантів використання;
- схема структурна компонентів програмного забезпечення.

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 літератури за тематикою проєкту	23.02	
2.	Розробка технічного завдання	13.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	21.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	26.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	14.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	18.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проєкту	19.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проєкту	21.05	Графічний матеріал проєкту
9.	Оформлення технічної документації проєкту	30.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

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

**Пояснювальна записка
до дипломного проєкту**

на тему: **Вебзастосунок для відслідковування макро- та мікронутрієнтів у раціоні**

КПІ.ПІ-1527.045440.02.81

ЗМІСТ

ВСТУП	5
1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ	7
1.1 Постановка завдання дипломного проєктування	7
1.2 Аналіз предметної області	7
1.3 Аналіз існуючих рішень.....	10
1.3.1 Аналіз відомих програмних продуктів	10
1.3.2 Аналіз відомих алгоритмічних та технічних рішень.....	16
1.4 Аналіз та моделювання бізнес-процесів.....	21
Висновки до розділу	22
2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	24
2.1 Варіанти використання програмного забезпечення.....	24
2.2 Розроблення функціональних вимог	24
2.3 Розроблення нефункціональних вимог.....	30
2.4 Аналіз системних вимог.....	31
2.5 Аналіз економічних показників програмного забезпечення.....	32
2.6 Постановка завдання на розробку програмного забезпечення	35
Висновки до розділу	36
3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	37
3.1 Архітектура програмного забезпечення.....	37
3.2 Архітектурні рішення та обґрунтування вибору засобів розробки	38
3.3 Конструювання програмного забезпечення.....	41
3.3.1 Розробка алгоритму рекомендації плану харчування	41
3.3.2 Опис структури бази даних	46
3.4 Аналіз безпеки даних.....	60
Висновки до розділу	61
4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	63
4.1 Аналіз якості ПЗ.....	63

4.2	Опис процесів тестування.....	64
4.3	Опис контрольного прикладу	73
	Висновки до розділу	78
5	РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	80
5.1	Розгортання програмного забезпечення.....	80
5.2	Супровід програмного забезпечення	87
	Висновки до розділу	87
	ВИСНОВКИ.....	88
	СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	90
	ДОДАТОК А ЗВІТ ПОДІБНОСТІ.....	94
	ДОДАТОК Б ОПИС ВАРІАНТІВ ВИКОРИСТАННЯ.....	95

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	– Integrated Development Environment – інтегроване середовище розробки.
API	– Application programming interface, прикладний програмний Інтерфейс.
SDK	– Software development kit.
ER	– Entity-Relation diagram.
OC	– Операційна система.
БД	– База даних.
CSR	– Client Side Rendering – формування відображення на клієнтській стороні.
SSR	– Server Side Rendering – формування відображення на серверній стороні.
SPA	– Single Page Application – односторінковий застосунок.

ВСТУП

Повноцінне, збалансоване харчування є одним із ключових факторів, що передбачають стан здоров'я та загальне самопочуття індивіда: воно корелює з покращеним когнітивним функціонуванням, сприяє профілактиці як хронічних, так і гострих захворювань, а також асоціюється з підвищеним рівнем суб'єктивного благополуччя. Існує широкий спектр заходів, що фасилітують дотримання належного раціону, але наріжним каменем у цьому процесі, беззаперечно, слугує систематичний моніторинг спожитої їжі, а також її макро- та мікронутрієнтного профілю.

Традиційно відстежування спожитих нутрієнтів здійснювалося шляхом ручного ведення харчових щоденників, що містили відомості про спожиту їжу та її кількість, з подальшим перехресним звірянням даних поживної цінності з паперових довідників та мануальним обчисленням вмісту кожного з біоактивних компонентів. Тоді як даний підхід був оптимальним для свого часу, він, тим не менш, мав низку очевидних недоліків: фіксація даних щодо спожитої їжі та підбиття підсумків потребують значних часових затрат, невід'ємною є потреба в нутріційних довідниках, а самі відомості в них нерідко можуть бути обмеженими – як за кількістю представлених продуктів, так і в обсязі даних про їхній нутріційний профіль.

Водночас, у міру становлення технологічних рішень та засобів усе доступнішими, а їхніх можливостей більшими, почала окреслюватися нова ніша для інтеграції інформаційних технологій у процес моніторингу харчування. Дійсно, можливість автоматизації внесення й обробки даних про споживання їжі, відсутність потреби у використанні паперових носіїв та фактично необмежений обсяг довідкових даних роблять цю перспективу дійсно привабливою. Це й відображає ринок – на сьогодні можна знайти широкий перелік застосунків даного спрямування, які досягли значного успіху, що зокрема свідчить про значний попит на сервіси даного типу.

Тим не менш, більшість популярних доробок представлених на ринку мають низку недоліків. Очевидно, дані продукти орієнтуються на

середньостатистичного користувача, для якого індивідуалізація нутріційних цілей та дивергенція від типових дієтичних настанов не є пріоритетними, а тому основний акцент при відстежуванні робиться на макронутрієнтній складовій, тоді як заглиблення в мікронутрієнтну залишається недоопрацьованим, а можливості персоналізації є мінімальними. Окрім цього, більшість сервісів функціонує за моделлю “freemium”, яка передбачає мінімальне задоволення функціональних потреб користувача, аби він продовжував “залишатися в орбіті” застосунку, але функції, що забезпечують комфортне користування, потребують оформлення платної підписки. Варто звернути увагу й на те, що переважна більшість продуктів такого спрямування сфокусована суто на мобільні платформи, тоді як вебзастосунки, доступні через браузер, становлять значно меншу частку ринку.

Дана робота присвячена розробці вебзастосунку для відслідковування макро- та мікронутрієнтів у раціоні, що забезпечуватиме спрощення мануальних операцій, котрі беруть участь у даному процесі, а також надаватиме можливість гнучкої персоналізації. Вона включатиме в себе аналіз наявних рішень та потенційних вимог до такого програмного забезпечення, а також розробку відповідного продукту з дотриманням сучасних архітектурних рішень та належної якості.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання дипломного проєктування

Дипломне проєктування передбачає виконання наступних завдань:

- аналіз предметної області та опис її ключових бізнес-процесів, визначення загального завдання розробки у рамках ДП;
- аналіз існуючих рішень (як програмних продуктів, так і алгоритмічних чи технічних рішень) обраного завдання розробки у рамках ДП;
- аналіз та моделювання бізнес-процесів;
- розроблення функціональних, нефункціональних та системних вимог до програмного забезпечення;
- аналіз економічних показників програмного забезпечення ДП;
- постановка завдання на розробку програмного забезпечення ДП;
- розроблення архітектури програмного забезпечення;
- розроблення архітектурних рішень та обґрунтування вибору засобів розробки програмного забезпечення;
- конструювання та розроблення програмного забезпечення;
- аналіз безпеки даних програмного забезпечення;
- аналіз якості та тестування програмного забезпечення;
- розгортання та супровід програмного забезпечення;
- створення супроводжувальної документації до розробленого програмного забезпечення.

1.2 Аналіз предметної області

За визначенням, нутрієнт – це органічна або неорганічна сполука, яку живим організмам необхідно отримувати з оточуючого середовища для підтримки основних процесів життєдіяльності, які включають в себе базальний метаболізм, ріст та підтримка тканин, активність, репродукцію та підтримку загального здоров'я [1]. Існує значний перелік можливих

класифікацій нутрієнтів, але одним із основних є поділ за обсягом, який необхідний організму для належного функціонування. Таким чином формуються дві категорії: макронутрієнти – такі, що споживаються у великих кількостях та є основним джерелом енергії, як-от білки, жири, вуглеводи – а також мікронутрієнти – вони споживаються в менших кількостях і переважно відіграють роль прекурсорів, регуляторів, коферментів, антиоксидантів, тощо, як-от вітаміни, мінерали, амінокислоти та інші [1].

З огляду на широкий спектр фізіологічних функцій, які нутрієнти виконують в організмі, постає доцільність у контролі отримання належних рівнів макро- та мікронутрієнтів. Оскільки основним джерелом надходження нутрієнтів або прекурсорів для них в організм людини є харчові продукти, чи не найважливішим чинником у забезпеченні оптимального нутрієнтного статусу виступає моніторинг та контроль раціону індивіда.

Насьогодні ринок сервісів, що забезпечують такий функціонал, є доволі насиченим та представлений значним переліком застосунків. Виходячи з сутності предметної області, стає зрозумілим, що наріжним каменем будь-якого бізнес-процесу в таких застосунках виступає створення записів про споживання їжі, а вже навколо нього додаються функціональності, що підвищують “якість користування”.

Таких функціональностей може бути багато. Наприклад, дослідження найпопулярніших мобільних застосунків для відслідковування нутрієнтів проаналізувало найпоширеніші функціональності, що зустрічалися в цих додатках [2]. Серед функцій введення найпоширенішими були: текстовий пошук їжі, вбудований сканер штрих-кодів, диверсифікація одиниць виміру – наприклад, грами, чайні й столові ложки, чашки, тощо – при додаванні записів, відслідковування споживання їжі, налаштування макронутрієнтів, створення власних рецептів, тощо. Найпоширенішими ж функціями виведення слугують: обчислення сумарної енергії в кілокалоріях, визначення відсоткового співвідношення макронутрієнтів, відстежування відповідності спожитій мікронутрієнтів поставленим цілям, перегляд нутріційних відомостей для

кожного запису, тощо. Також варто відзначити, що один із наведених застосунків також надавав функціонал автоматичної рекомендації раціону для поставлених фітнес-цілей.

Інше ж дослідження поставило на меті визначення які основні фактори або функції роблять додаток більш привабливим для користувачів [3]. Таким чином було встановлено, що найпривабливішими рисами були: простота використання, доступність без оплати, автоматична оцінка калорій та нутрієнтів, автоматична оцінка даних про їжу (наприклад, по штрих коду), підтримка зберігання історичних записів, можливість створення власних рецептів, а також інші. Найбільшими ж бар'єрами до вибору сервісу були: відсутність відомостей про поширену їжу, неправильна оцінка калорій та нутрієнтів, відсутність відомостей про місцеву їжу, відсутність функцій персоналізації, мануальне внесення є єдиним способом додання даних, неправильне автоматичне розпізнавання їжі, тощо. Також цікавим спостереженням є факт, що лише 4 особи з майже трьохтисячної вибірки опитуваних користувалися немобільною операційною системою. Хоча саме по собі це не може бути підставою для обґрунтованих стверджень, воно, тим не менш, спонукає до роздумів щодо причин такої розбіжності. Оскільки попит на такі застосунки є значним, а немобільні платформи є одними з найпоширеніших, можна припустити, що це частково зумовлено недостатньою представленістю вебзастосунків, що надають такі послуги, на ринку.

На основі сформульованої моделі розуміння предметної області очевидно, що для побудови конкурентного програмного засобу, функціонал систем має адресувати такі вимоги: централізація нутріційних даних продуктів зі створенням обширної вибірки продуктів, підкріпленої широким вибором можливих одиниць виміру, можливість ведення записів про споживання їжі, підтримка обліку макро- та мікронутрієнтів, автентифікація користувачів для подальшого збереження їхніх нутріційних цілей, створення звітності про споживання, а також заходів автоматизації ведення обліку споживання. Крім

того, додатковою перевагою буде реалізація функцій автоматичної рекомендації і зручних механізмів введення, як-от сканування штрих-кодів.

1.3 Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації вебзастосунку для відслідковування макро- та мікронутрієнтів у раціоні. Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

1.3.1 Аналіз відомих програмних продуктів

На IT-ринку існує доволі широкий спектр програмних забезпечень, що застосовуються для відслідковування макро- та мікронутрієнтів у раціоні: як безкоштовні, так і ліцензійні, як десктопні чи мобільні додатки, так і вебзастосунки. У рамках даної роботи фокус припадатиме саме на останні. Розгляньмо детально приклади деяких з них у даному розділі, а опісля проведемо їхній порівняльний аналіз.

– MyNetDiary

Одним із першопроходців даної ніші є MyNetDiary. MyNetDiary – це повнофункціональний та зручний мультиплатформенний сервіс для моніторингу спожитої їжі, а також її макро- та мікронутрієнтої складових [4]. Розроблений у 2005 році, застосунок знайшов своє місце серед користувачів, що прагнуть схуднення, підтримки ваги або загального стану здоров'я. Даний сервіс надає користувачу можливість обирати їжу з великої бази даних, що постійно оновлюється, створювати записи про споживання їжі, а також проводить автоматичне обчислення нутрієнтої складової спожитої їжі (рисунок 1.1). Сервіс також надає певний ступінь автоматизації процесу ведення щоденника харчування за допомогою сканування штрих-кодів продуктів з мобільного застосунку, створення власних рецептів, а також отримання детальної статистики. Даний застосунок має як безкоштовну версію з базовим функціоналом, так і платну на основі підписки, що дозволяє

змінювати нутріційні цілі, а також відслідковувати більше нутрієнтів – до 108. Також є функція експорту звітів у форматі PDF та CSV.

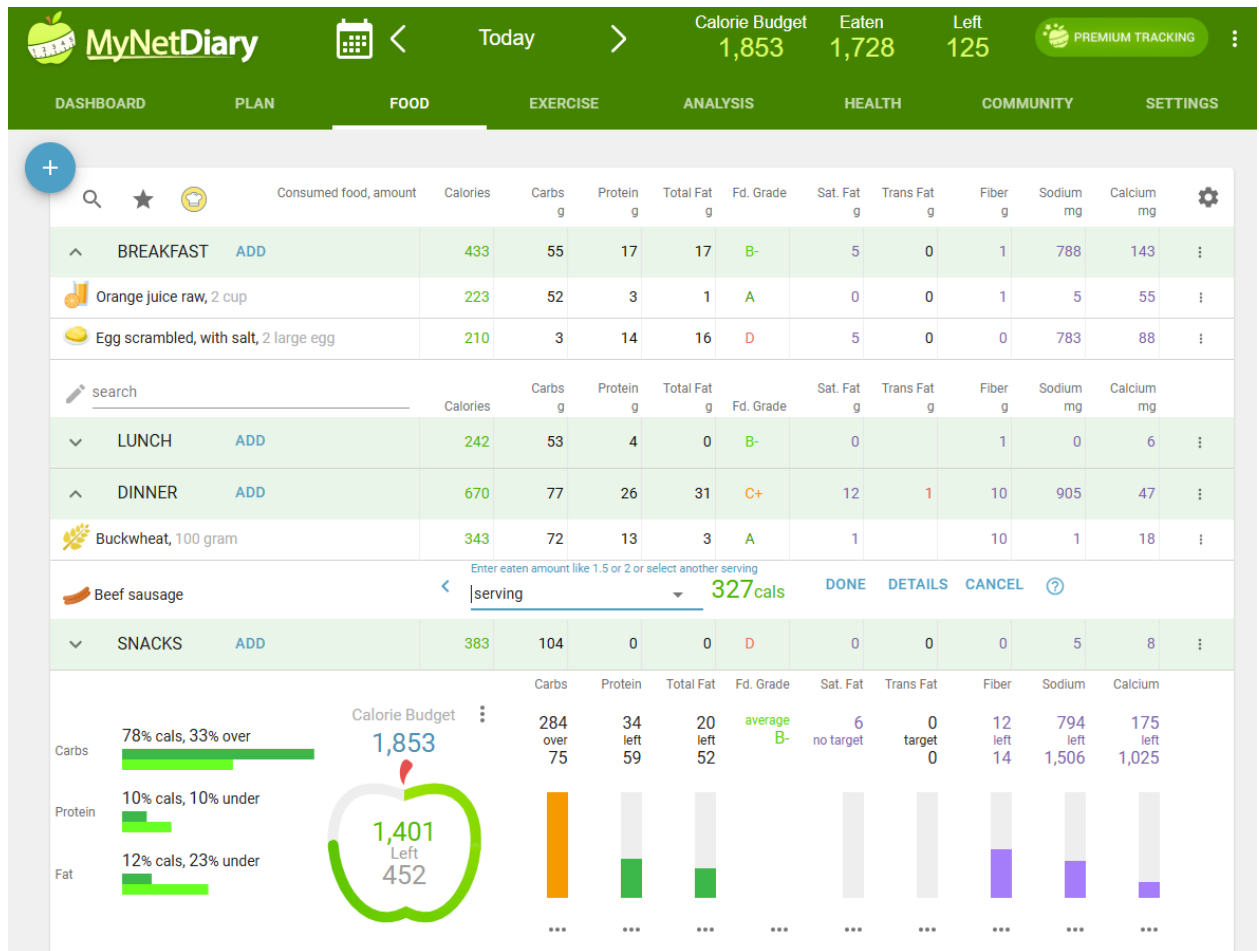


Рисунок 1.1 – сторінка MyNetDiary з інформацією про спожиту їжу, а також її нутрієнтну характеристику

– EatThisMuch

Відносно новим продуктом на ринку програмного забезпечення даного спрямування є EatThisMuch. EatThisMuch – застосунок для моніторингу спожитої їжі, що не лише надає базові функції відслідковування спожитої їжі, а й виділяється на фоні своїх конкурентів можливістю постановки дієти “на автопілот” за допомогою функції планування раціону (рисунок 1.2) [5]. Дана функція зчитує встановлені користувачем нутріційні цілі за макронутрієнтами, а тоді звертається до своєї бази даних рецептів і формує з них режим харчування на день. Згенерований розпорядок опісля можна зекспортувати в щоденник харчування. Звітність проводиться за 92 нутрієнтами, але цілі можна задавати лише для макронутрієнтів і холестерину.

Додаток, як було згадано раніше, має колекцію рецептів, які можна переглядати та додавати до свого раціону користувачам. Присутня й функція створення власних рецептів. Аналогічно до свого попередника є як безкоштовний план, так і платний: платна підписка дозволяє експортувати звіти у форматі PDF та надає можливість автономно формувати списки покупок на основі запланованих розпорядків харчування та автоматично проводити купівлю й доставку продуктів за адресою з онлайн-платформ.

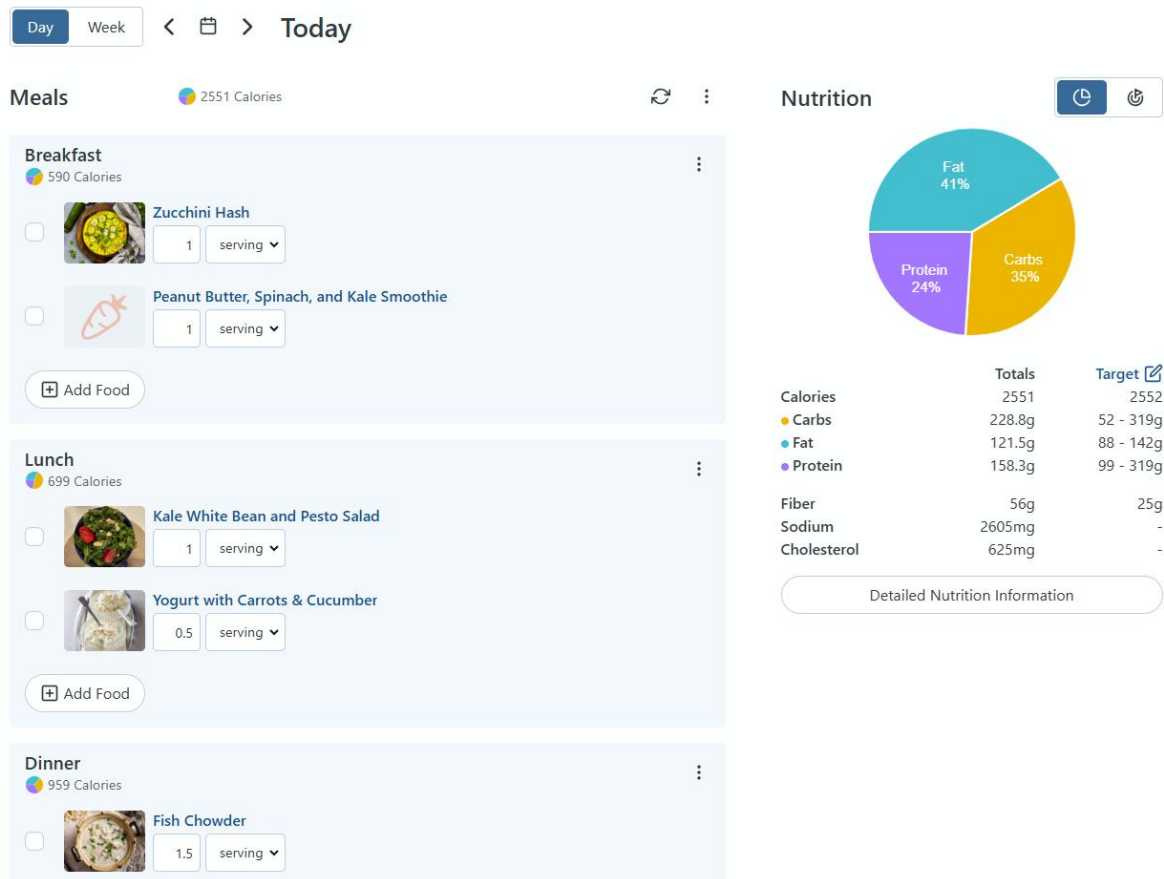


Рисунок 1.2 – сторінка планувальника дієти EatThisMuch

– Cronometer

Представлений як веб-клієнтом, так і мобільним додатком, даний застосунок є чи не найбільшим на ринку сервісів для відстеження харчування. Cronometer позиціонує себе як один з найбільш точних інструментів свого спрямування: він використовує лише верифіковані бази даних нутріційних відомостей продуктів, як-от Nutrionix, NCCDB, USDA, тощо [6]. Даний застосунок виділяється серед конкурентів своєю прагматичністю: простий дизайн доповнює інтуїтивний інтерфейс, уся інформація про нутрієнти

подається акуратно та систематизовано без потреби в окремих діях, а базовий функціонал не потребує додаткових налаштувань чи навігації по меню. Окрім же базових функцій, представлений і функціонал покращення якості користування: можна створювати власні рецепти, додавати власну їжу, проводити як текстовий пошук їжі, так й ідентифікацію за штрих-кодом. Тоді як базові можливості доступні безкоштовно, даний застосунок також має платну підписку, що розширить функціонал можливістю підбору найбільш підходящої їжі за цільовим нутрієнтом, допоможе підібрати рецепт на основі фільтру та поставлених макронутрієнтних цілей, створення розкладу автоматичного логування продуктів та низкою інших корисних функцій. Усього застосунок дозволяє відслідковування до 84 нутрієнтів.

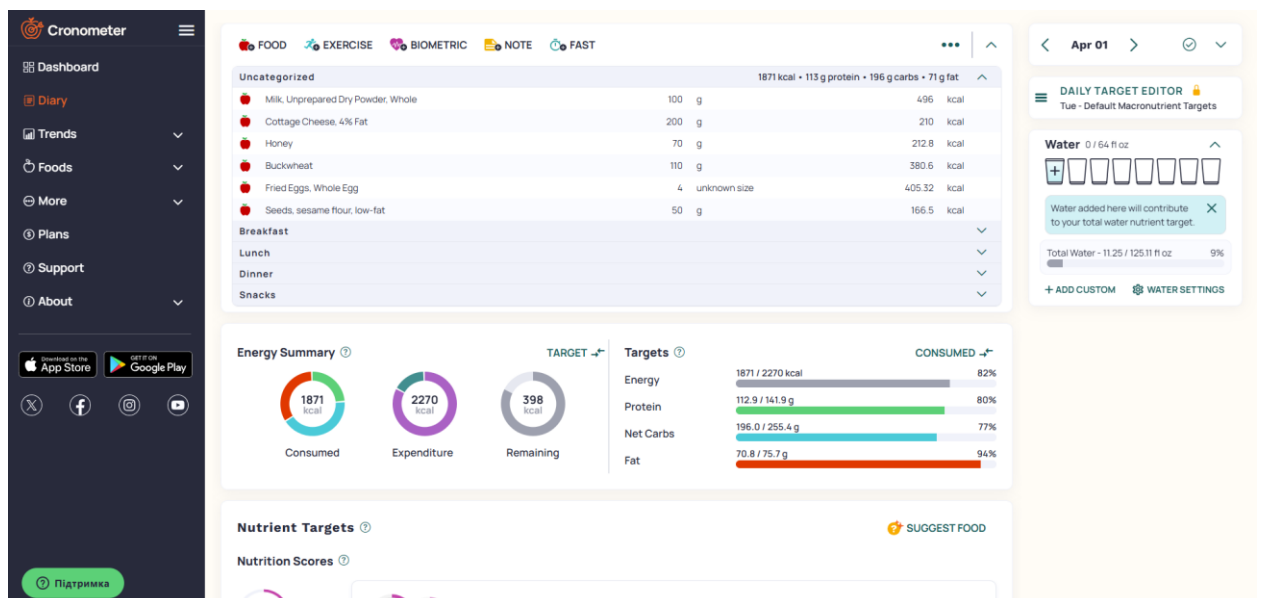


Рисунок 1.3 – сторінка щоденника харчування Cronometer

Тепер проведемо порівняльний аналіз застосунків та даної роботи. Порівняння наведено в таблиці 1.1

Таблиця 1.1 – Порівняння з аналогами

Функціонал	NutriMatriх	MyNetDiarу	EatThisMunch	Cronometer	Пояснення
Відстеження макрота мікронутрієнтів	Так	Так	Так	Так	Підтримується всіма сервісами
Підтримувані платформи	Вебзастосунок	Вебзастосунок, мобільні	Вебзастосунок, мобільні	Вебзастосунок, мобільні	Усі комерційні продукти мають як мобільні клієнти, так і веб-сторінки
Персоналізація переліку нутрієнтів для відслідковування	Так	Так (платно)	Ні	Так	MyNetDiarу надає вкрай обмежений перелік нутрієнтів, потребує підписки
Формування списків покупок	Ні	Ні	Так	Ні	Не є цільовим для ДП

Продовження таблиці 1.1

Функціона л	NutriMatri x	MyNetDiar y	EatThisMu ch	Cronometer	Пояснення
Максимальна кількість нутриєнтів для відслідков ування	161	108	92	84	Розробка пропонує найбільши й вибір
Функція помічника підбору раціону	Так	Ні	Частково	Частково (платно)	Усі комерційні розробки дозволяють ставити цілі лише за макронутр ієнтами
Функція створення шаблонів страв з інгредієнта в	Так	Так	Так	Так	Підтримує тільки всіма сервісами
Сканування штрих- кодів продуктів	Так	Частково	Частково	Частково	Частково – дозволяють лише на мобільних клієнтах

Продовження таблиці 1.1

Функціонал	NutriMatriх	MyNetDiarу	EatThisMunch	Cronometer	Пояснення
Експорт даних	Так	Так	Так	Так	Підтримується всіма сервісами
Трекінг метрик тіла	Ні	Так	Ні	Так	Не є цільовим для ДП
Автоматизація логування їжі	Так	Ні	Опосередковано	Так	EatThisMunch допускає часткову автоматизацію окремих аспектів
Має публічну базу рецептів	Так	Ні	Так	Ні	Передбачено тільки одним комерційним сервісом

1.3.2 Аналіз відомих алгоритмічних та технічних рішень

Перш за все, буде застосовано клієнт-серверний тип архітектури, що передбачає розбиття системи на серверну частину, котра відповідає за централізовану обробку і управління даними чи ресурсами, а також виконання більшої частини обчислень, та клієнт, котрий несе відповідальність за

взаємодію з користувачем, представлення інформації, а також запити до серверної частини. Дана архітектура має перевагу над монолітним типом, оскільки вона сприяє розділенню обов'язків, полегшенню масштабованості, а також вона надзвичайно добре та природньо інтегрує в себе концепцію CSR, що у свою чергу в порівнянні з традиційним SSR дозволяє заощаджувати ресурси хостингу [7]. Тим не менш з таким підходом з'являються й декілька недоліків, як от часткове збільшення обсягів даних, що передаються між частинами застосунку, а також збільшення складності розгортання, оскільки на відміну від монолітної архітектури, розгортати клієнтську та серверну частини потрібно окремо. Дані недоліки компенсуються наявними перевагами, отож, дану архітектуру було обрано за основу додатку.

Крім цього, клієнтська частина природньо містить велику кількість комплексної логіки, тому доцільно буде розглянути можливість застосування архітектурного патерну й для неї. Серед можливих альтернатив є:

- Монолітна архітектура: традиційний підхід, у якому застосунок представлений одним блоком. Оскільки всі елементи знаходяться в одному місці, цей метод простий у впровадженні та використанні. Тим не менш, монолітна архітектура часто може стати значним обмеженням у розробці, якщо потрібний великий масштабований проект, перехід із вимог у реалізацію може бути не ідеальним, що є надзвичайно частим явищем та навіть стало одним зі стимулів формування цілих життєвих циклів розробки програмного забезпечення. Крім цього, також часто виникають проблеми із супроводженням по мірі зростання обсягів кодової бази та складності з часом. По мірі збільшення розмірів додатку ці проблеми набуватимуть більших масштабів [8].

- Тришарова архітектура: даний патерн розбиває застосунок на три основні шари: шар представлення, шар бізнес-логіки та шар доступ до даних. Шар представлення відповідає за представлення інтерфейсу для користувача, бізнес-логіка виконує необхідні комплексні операції, а шар доступу до даних взаємодіє з джерелами даних, зазвичай БД або файловими джерелами.

Особливість комунікації між шарами полягає в тому, що кожен наступний шар може взаємодіяти лише з попереднім. Чітке розподілення обов'язків між компонентами, зручність управління та підтримка розширюваності становлять головні переваги, що забезпечуються даною архітектурою. Однак притаманна складність, і потенційні проблеми зі швидкістю при обробці значних обсягів даних між різними шарами є потенційними проблемами, що можуть виникати в даній архітектурі [9].

– Мікросервісна архітектура: за даної архітектури застосунок розбивається на невеликі самодостатні компоненти – мікросервіси. Кожен мікросервіс має чітко встановлену мету та володіє власним інтерфейсом для взаємодії з іншими мікросервісами та клієнтом. Зазвичай зовнішній доступ до мікросервісного додатку забезпечується за рахунок API-шлюзів, що контролюють методи, доступні для виклику. Дана архітектура має значну кількість переваг: мікросервіси володіють низьким ступенем зв'язності, оскільки напряду не залежать один від одного, а звідси й постає простота масштабування та зменшення впливу наявності дефектів у компоненті чи його повної відмови на роботу системи в цілому. Водночас з цим і приходить великий недолік – перш за все розгортання мікросервісного додатку є надзвичайно складним завданням, яке потребує значних зусиль та вмінь, а сама розробка таких застосунків супроводжується підвищеною технічною складністю [10].

– Розширювана архітектура: передбачає розмежування шарів доменної логіки, прикладного сервісного рівня та зовнішніх залежностей додатку та забезпечує інверсію контролю й мінімізацію зв'язності між компонентами; унаслідок застосування цього патерну і підтримуваність коду значно підвищуються, оскільки зміни в інфраструктурних модулях (наприклад, доступі до даних або зовнішніх API) не потребують модифікації внутрішніх доменних сервісів, а також створюється можливість поступового розширення функціоналу без порушення існуючих контрактів між шарами. Тим не менш, розширюваний патерн також призводить до ускладнення

початкової конфігурації проєкту та призводить до значного збільшення кількості шаблонного коду й технічного боргу через необхідність явного визначення інтерфейсів і залежностей між шарами [11].

У ході опису функціональності програмного забезпечення, опосередковано формуються окремі спостереження. Так, можна помітити, що домейна модель природним чином розкладається на низку скutih контекстів: наприклад, модулю записів споживання немає потреби у детальному розумінні того, що саме було спожито та яка його внутрішня структура – достатньо лише вказати його ідентифікатор; аналогічно, при роботі з рецептами зберігання лише ідентифікаторів інгредієнтів виявляється достатнім. Також можна зафіксувати й певну неоднорідність у характері очікуваного обчислювального навантаження між складовими системи: модуль рекомендацій притаманно має більше алгоритмічне навантаження та характеризується значно вищими вимогами до обчислювальних ресурсів порівняно, наприклад, із модулем, основна функціональність якого зводиться до взаємодії з БД. Окрім того, перевантаження або відмова модуля рекомендацій може спричинити перешкоди в роботі модулів, що забезпечують першочергову функціональність застосунку, тоді як даний є лише другорядним. Це у свою чергу також підіймає інтерес в можливості застосування різних підходів до масштабування окремих складових компонентів системи. І врешті-решт, даний застосунок надає вкрай різноманітну функціональність, а простір для додавання нових можливостей є фактично необмеженим, з чого постає потреба в належному проєктуванні структурної організації додатку.

На основі даних спостережень можна схилитися до того, що найбільш вдалим рішенням буде застосування мікросервісної архітектури для серверної складової даного застосунку. Крім цього, також можна й застосовувати архітектурні патерни до самих сервісів. Чудовим вибором може слугувати розширована архітектура, оскільки вона забезпечує чітке розмежування

доменної логіки, інфраструктурних компонентів і зовнішніх залежностей, що сприяє підвищенню гнучкості системи.

Окрім цього даною системою також передбачає реалізацію алгоритму підбору такого раціону, що найкраще відповідатиме заданим нутріційним цілям, а включені страви узгоджуватимуться із вказаним критерієм. Оскільки це є задача підбору раціону з дискретними змінними й цілочисельними параметрами, котра до того ж потребує належної швидкодії у зв'язку з великим обсягом потенційних кандидатів для оцінювання, метаевристичний підхід видається найбільш доречним.

Насьогодні існує широкий спектр метаевристичних алгоритмів, які могли б підійти для вирішення даної задачі. Серед найбільш поширених можна виділити:

- Генетичний алгоритм: ітеративний метаевристичний алгоритм, у якому на початку формується множина кандидатних рішень – популяція – а кожному елементу обчислюється значення функції оцінки якості [12]. Після відбувається відбір за турнірною чи рулетковою селекцією, у результаті чого здійснюються операції кросовера та мутації для отримання нового покоління. Ітераційний процес триває до виконання критерію зупинки, після чого результатом вважається найкращий наявний індивід у популяції.

- Мурашиний алгоритм: даний алгоритм імітує пошук найкоротших маршрутів через групу мурах, що рухаються графом за ймовірнісною функцією, обчисленою на основі довжини шляху та концентрації феромонів [13]. У кінці кожної ітерації на пройдених ребрах відбувається додаткове відкладення даних феромонів, тоді як процес випаровування зменшує їхню концентрацію на менш використовуваних ділянках. Ітерації проводяться допоки не буде виконано критерію виходу; після чого результатом визнається найкраще значення відстані.

Оскільки генетичний алгоритм характеризується відносною простотою реалізації, не вимагає побудови графової моделі, що у разі застосування мурашиного алгоритму ускладнює процес оптимізації та потребує додаткових

обчислювальних ресурсів, а також демонструє високу масштабованість при роботі з великими обсягами дискретних комбінацій, було вирішено віддати перевагу саме цьому підходу.

Отож, результуюча архітектура додатку складатиметься з клієнтської частини та серверної, котра у свою чергу також використовуватиме мікросервісну архітектуру, а самі мікросервіси дотримуватимуться розширеної організації. Механізм надання рекомендованого раціону для поставлених нутріційних цілей буде реалізовано на основі генетичного алгоритму.

1.4 Аналіз та моделювання бізнес-процесів

Як було описано в розділі 1.2, основною складовою більшості бізнес-процесів виступатиме збереження записів споживання їжі, отримання їхньої нутріційної характеристики, або ж конфігурація нутрієнтів, відомості щодо яких будуть надаватися. Окрім цього, також можуть фігурувати й додаткові можливості, що підвищують якість користування. Розгляньмо таку комбінацію та змоделюємо її бізнес-процеси. Нею може бути процес планування розкладу харчування. Даний бізнес-процес представлений BPMN-моделлю, наведеною на кресленні «Схема структурна діяльності». Опис даного бізнес-процесу:

- користувач переходить на сторінку планувальника раціону;
- користувач вводить нутріційні цілі, яких він хоче досягти з планом. У той же час клієнтська частина застосунку оновлює внутрішнє представлення моделі запиту на планування у відповідності до введених користувачем значень;
- користувач натискає кнопку запиту на планування раціону;
- клієнтська частина надсилає запит на серверну частину, котра у свою чергу проводить пошук кандидатів у відповідності до специфікацій,

указаних у запиті, отримує дані рецептів з локального кешу, проводить пошук найбільш відповідної комбінації та повертає результат.

- клієнтська частина отримує відповідь та відображає її користувачу. Якщо користувач не задоволений результатом, він надсилає повторний запит на планування;

- якщо користувач вважає результат задовільним, він натискає кнопку збереження плану;

- клієнтська частина надсилає запит збереження на серверну частину, котра додає обрані до колекції приватних рецептів користувача, додає вказані кількості рецептів до щоденника харчування на вказану дату та повертає результат додавання;

- клієнтська частина відображає результат збереження, який у свою чергу бачить користувач. Процес планування завершений.

Висновки до розділу

У даному розділі було розглянути теоретичні положення, а також сформульовано на їх основі модель розуміння предметної. Проаналізовано стан сучасних рішень для відслідковування макро- та мікронутрієнтів у раціоні на сучасному ринку, розглянуто основний функціонал, ключові моменти, а також проведено детальний аналіз прикладів таких систем – MyNetDiary, EatThisMuch та Cronometer – із подальшим порівняльним аналізом.

Проаналізовано та обрано архітектурні паттерни для застосунку та його частин – ними визначено клієнт-серверну архітектуру з використанням мікросервісної архітектури для серверної частини з подальшою організацією сервісів за розшарованим архітектурним патерном. Розглянуто можливі алгоритмічні рішення задачі підбору раціону й встановлено, що у даному контексті доцільно використовувати генетичний алгоритм.

У рамках аналізу вимог до ПЗ було створено діаграму варіантів використання, деталізовано низку варіантів використання, сформовано

загальну модель вимог; сформульовано функціональні вимоги та побудовано матрицю трасування співвідношення варіантів використання та вимог. Сформульовано нефункціональні вимоги.

Здійснено формалізоване моделювання основного бізнес-процесу системи за допомогою BPMN-діаграми, в якій кожен етап — від ініціації до отримання результату — представлено як окремий елемент моделі та описано у текстовому варіанті. Сформовано модель розуміння предметної області.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Діаграма варіантів використання з ключовими акторами та основними функціями показана на кресленні «Схема структурна варіантів використання». В додатку Б наведено детальний опис варіантів використання програмного забезпечення.

2.2 Розроблення функціональних вимог

Даний застосунок розділений на окремі компоненти. Кожен компоненти відповідає за окремий набір функцій. В таблиці 2.1 наведено загальну модель вимог, а в таблиці 2.2 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог продемонстровано на рисунку 2.2.

Таблиця 2.1 – Загальна модель вимог

№	Назва	ID Вимоги	Пріоритети	Ризики
1	Система управління користувачами	FR-1	Високий	Високий
1.1	Реєстрація користувача.	FR-2	Високий	Високий
1.2	Підтвердження пошти	FR-3	Середній	Низький
1.3	Автентифікація користувача.	FR-4	Середній	Середній
1.4	Вихід із акаунту.	FR-5	Низький	Низький
1.5	Редагування особистих даних	FR-6	Середній	Середній
1.6	Налаштування відслідковуваних нутрієнтів	FR-7	Низький	Низький
2	Перегляд каталогу їжі	FR-8	Середній	Середній
2.1	Пошук їжі за назвою	FR-9	Високий	Низький
2.2	Пошук їжі за штрих-кодом	FR-10	Високий	Низький
3	Перегляд рецептів	FR-11	Середній	Низький
3.1	Перегляд публічних рецептів	FR-12	Низький	Низький

Продовження таблиці 2.1

№	Назва	ID Вимоги	Пріоритети	Ризики
3.1.1	Пошук публічних рецептів за назвою	FR-13	Низький	Низький
3.2	Перегляд власних рецептів	FR-14	Середній	Низький
3.2.1	Пошук власних рецептів за назвою	FR-15	Низький	Низький
4	Додавання записів споживання	FR-16	Високий	Середній
4.1	Додавання записів споживання продуктів	FR-17	Високий	Низький
4.2	Додавання записів споживання рецептів	FR-18	Високий	Низький
4.3	Додавання планування записів споживання продуктів	FR-19	Середній	Середній
5	Зміна стану записів споживання	FR-20	Середній	Середній
5.1	Редагування записів споживання	FR-21	Високий	Середній
5.2	Видалення записів споживання	FR-22	Середній	Середній
6	Перегляд записів споживання	FR-23	Високий	Середній
7	Перегляд статистики споживання нутрієнтів	FR-24	Високий	Низький
8	Формування раціону сервісом рекомендації	FR-25	Високий	Середній
8.1	Збереження рекомендації	FR-26	Середній	Середній
9	Створення рецептів	FR-27	Середній	Середній
9.1	Створення власних рецептів	FR-28	Середній	Низький
9.2	Створення публічних рецептів	FR-29	Середній	Середній
10	Зміна стану рецептів	FR-30	Середній	Середній
10.1	Оновлення власного рецепту	FR-31	Середній	Середній

Продовження таблиці 2.1

№	Назва	ID Вимоги	Пріоритети	Ризики
10.2	Видалення власного рецепту	FR-32	Середній	Середній
10.3	Оновлення публічного рецепту	FR-33	Середній	Середній
10.4	Видалення публічного рецепту	FR-34	Середній	Середній
11	Створення їжі	FR-35	Високий	Середній

Таблиця 2.2 – Перелік функціональних вимог

Назва	Опис
FR-1	Система управління користувачами. Додаток зобов'язаний надавати повну систему для управління користувачами, що включає в себе авторизацію, автентифікацію та управління даними користувачів.
FR-2	Реєстрація користувача. Неавтентифікований користувач може створити новий обліковий запис, ввівши ім'я користувача, електронну пошту та пароль. Після реєстрації користувач отримує повідомлення про успішну реєстрацію та заклик підтвердити електронну пошту.
FR-3	Підтвердження пошти. Користувачу необхідно перейти за посиланням, надісланим на введену електронну пошту, для підтвердження електронної адреси. Без підтвердження електронної пошти, основний функціонал застосунку залишається недоступним.
FR-4	Автентифікація користувача. Неавтентифікований користувач може увійти в систему, ввівши електронну пошту та пароль.
FR-5	Вихід із акаунту. Автентифікований користувач може вийти з облікового запису, завершуючи сеанс роботи в системі.

Продовження таблиці 2.2

Назва	Опис
FR-6	Редагування особистих даних. Автентифікований користувач може редагувати особисті дані, такі як ім'я, прізвище, контактна інформація тощо.
FR-7	Налаштування відслідковуваних нутрієнтів. Автентифікований користувач може вибирати нутрієнти, які бажає відслідковувати.
FR-8	Перегляд каталогу їжі. Автентифікований користувач може переглядати доступні продукти харчування в каталозі та отримувати детальну інформацію про кожен продукт.
FR-9	Пошук їжі за назвою. Автентифікований користувач може шукати продукти харчування в каталозі за назвою, вводячи пошуковий запит. Результати пошуку відображаються відповідно до введенного запиту.
FR-10	Пошук їжі за штрих-кодом. Автентифікований користувач може шукати продукти харчування за штрих-кодом, скануючи штрих-код за допомогою камери пристрою.
FR-11	Перегляд рецептів. Автентифікований користувач може переглядати доступні рецепти.
FR-12	Перегляд публічних рецептів. Автентифікований користувач може переглядати доступні публічні рецепти.
FR-13	Пошук публічних рецептів за назвою. Автентифікований користувач може проводити пошук публічних рецептів за назвою.
FR-14	Перегляд власних рецептів. Автентифікований користувач може переглядати свої рецепти.
FR-15	Пошук власних рецептів за назвою. Автентифікований користувач може проводити пошук власних рецептів за назвою.
FR-16	Додавання записів споживання. Автентифікований користувач може додавати нові записи про споживання продуктів або рецептів.

Продовження таблиці 2.2

Назва	Опис
FR-17	Додавання записів споживання продуктів. Автентифікований користувач може додавати нові записи про споживання продуктів, вказуючи продукт, одиницю виміру та кількість спожитих одиниць.
FR-18	Додавання записів споживання рецептів. Автентифікований користувач може додавати нові записи про споживання рецептів, вказуючи рецепт та кількість спожитих порцій рецепту.
FR-19	Додавання планування записів споживання продуктів. Автентифікований користувач може налаштовувати автоматичне додавання майбутніх записів про споживання, вказуючи продукт, кількість та час споживання.
FR-20	Зміна стану записів споживання. Автентифікований користувач може змінювати стан записів споживання.
FR-21	Редагування записів споживання. Автентифікований користувач може редагувати наявні записи про споживання, оновлюючи інформацію про продукт, одиницю виміру та кількість спожитих одиниць.
FR-22	Видалення записів споживання. Автентифікований користувач може видаляти наявні записи про споживання.
FR-23	Перегляд записів споживання. Автентифікований користувач може переглядати записи про споживання за обраний день, обираючи дату в календарі.
FR-24	Перегляд статистики споживання нутрієнтів. Автентифікований користувач може переглядати статистику споживання нутрієнтів за заданий період, обираючи дати в календарі.

Продовження таблиці 2.2

Назва	Опис
FR-25	Формування раціону сервісом рекомендації. Автентифікований користувач може отримувати персоналізований раціон на основі встановлених нутріційних цілей, обираючи кількість страв та їхні характеристики.
FR-26	Збереження рекомендації. Автентифікований користувач може зберігати рекомендований раціон у своєму профілі, додаючи запропоновані рецепти в колекцію та записи про споживання в щоденник харчування.
FR-27	Створення рецептів. Система зобов'язана надати засоби створення нових рецепти, а саме особистих та публічних
FR-28	Створення власних рецептів. Автентифікований користувач може створювати власні рецепти, вводячи назву, опис, перелік інгредієнтів та кроки приготування.
FR-29	Створення публічних рецептів. Адміністратор може створювати нові публічні рецепти, вводячи назву, опис, перелік інгредієнтів та кроки приготування.
FR-30	Зміна стану рецептів. Система зобов'язана надати засоби зміни стану рецептів, а саме редагування та видалення.
FR-31	Оновлення власного рецепту. Автентифікований користувач може оновлювати інформацію свого рецепту, таку як назва, перелік інгредієнтів, кроки приготування тощо.
FR-32	Видалення власного рецепту. Автентифікований користувач може видаляти свої рецепти з системи.
FR-33	Оновлення публічного рецепту. Адміністратор може оновлювати інформацію публічного рецепту, таку як назва, опис, перелік інгредієнтів та кроки приготування.

Продовження таблиці 2.2

Назва	Опис
FR-34	Видалення публічного рецепту. Адміністратор може видаляти публічні рецепти з системи.
FR-35	Створення їжі. Адміністратор може додавати нові продукти в каталог, вводячи назву, харчову цінність, одиниці виміру та зображення.

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15	FR-16	FR-17	FR-18	FR-19	FR-20	FR-21	FR-22	FR-23	FR-24	FR-25	FR-26	FR-27	FR-28	FR-29	FR-30	FR-31	FR-32	FR-33	FR-34	FR-35	
UC-1	+	+	+																																	
UC-2	+			+																																
UC-3	+					+																														
UC-4	+						+																													
UC-5								+																												
UC-6																+	+																			
UC-7																							+	+												
UC-8																			+																	
UC-9								+												+																
UC-10											+	+	+																							
UC-11											+				+	+																				
UC-12																																+	+			
UC-13										+																										
UC-14																																				
UC-15																																				
UC-16																																				
UC-17																																				
UC-18																																				
UC-19																																				
UC-20																																				
UC-21																+			+																	
UC-22					+																															+
UC-23																												+	+							

Рисунок 2.2 – Матриця трасування вимог

2.3 Розроблення нефункціональних вимог

Під час проектування та реалізації даного програмного забезпечення необхідно урахувати наступні нефункціональні вимоги:

- передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних.
- розробку серверної частини виконати на мові програмування C#. Реалізацію контролерів буде проведено за допомогою технологічного середовища ASP.Net Core. Взаємодія із базою даних провдити за підтримки Entity Framework Core. Автентифікацію та авторизацію користувачів здійснювати з використанням Identity ASP.Net core. У якості БД використовувати PostgreSQL, Redis та Qdrant.

- клієнтську частину системи реалізовувати за допомогою технологічного середовища Angular та мови програмування TypeScript.
- розробку виконати за допомогою платформ Visual Studio 2022 версії 17.8.22 та WebStorm 2023.2.5 версії 232.10227.9.
- вихідний код програми має бути представлений у вигляді файлів з розширеннями .html, .css, .ts, .cs, .csproj, .sln, .json, тощо.
- паролі мають зберігатися в захешованому вигляді, а сам алгоритм хешування повинен бути однонапрямленим.
- комунікація між клієнтською та серверною частинами мають здійснюватися за допомогою протоколу HTTPS.
- комунікація між компонентами серверної частини має здійснюватися асинхронно, з використанням подійних шин.

2.4 Аналіз системних вимог

Програмне забезпечення серверної частини повинно функціонувати на серверах з операційною системою Ubuntu.

Вебзастосунок передбачає розгортання щонайменше 4 мікросервісів та одного клієнтського застосунку, отож, доцільно використовувати щонайменше 4 ядра. Кожен із сервісів потребує щонайменше 500 мегабайтів оперативної пам'яті самі по собі; крім цього, передбачається також і застосування сховища кешування, яке буде займати щонайменше 150 МБ ОП, та однієї векторної бази даних, яка потребуватиме близько 500 МБ. Крім цього, потребуватиме пам'яті й брокер повідомлень – він займатиме близько 300 МБ. API-шлюз потребуватиме близько 400 МБ пам'яті. База даних також займатиме близько 400 МБ ОП.

Таким чином мінімальна конфігурація технічних засобів для серверної частини орієнтовано становить:

- Кількість ядер: 4;
- об'єм ОЗП: 8Гб.

Рекомендована конфігурація технічних засобів для серверної частини:

- Кількість ядер: 12;
- об'єм ОЗП: 16 Гб.

2.5 Аналіз економічних показників програмного забезпечення

У рамках оцінки економічних показників даного проекту буде обчислено час, необхідний на виконання даного проекту однією особою на основі оцінки Use-Case одиниць.

Розпочнімо з оцінки неузгодженої ваги акторів UAW – її наведено в таблиці 2.3.

Таблиця 2.3 – Обчислення UAW

Складність	Кількість	Коефіцієнт	
Простий	0	1	0
Середній	0	2	0
Складний	3	3	9
Обчислення UAW			9

Наступним кроком є проведення оцінки неузгодженої ваги юз-кейсів UUCW – результати наведено в таблиці 2.4.

Таблиця 2.4 – Обчислення UUCW

Складність	Кількість	Коефіцієнт	
Простий	13	5	45
Середній	9	10	90
Складний	2	15	30
Обчислення UUCW			165

Відповідно, неузгоджені Use-Case одиниці становлять:

$$UUCP = UAW + UUCW = 174 \quad (2.1)$$

Наступним кроком є обрахування факторів узгодження. Розпочнімо з фактору технічної складності, що обчислюється за формулою:

$$TCF = 0.6 + 0.01 \sum F_i \cdot W_i, \quad (2.2)$$

Де F_i – значення оцінки окремого фактора складності, а W_i – його вага. Усього є 13 факторів – їхня оцінка наведена в таблиці 2.5.

Таблиця 2.5 – Оцінка факторів технічної складності

Назва	Значення	Вага	$F_i \cdot W_i$
Розподіленість системи	0	2	0
Вимоги до продуктивності	3	1	3
Ефективність кінцевого користувача	1	1	1
Складність внутрішньої обробки	2	1	2
Повторне використання	0	1	0
Простота інсталяції	0	0.5	0
Простота користування	2	0.5	1
Портативність	0	2	0
Змінюваність	0	1	0
Паралелізм роботи	3	1	3
Особливі вимоги до безпеки	2	1	2

Продовження таблиці 2.5

Надання прямого доступу для зовнішніх сутностей	0	1	0
Тренування кінцевих користувачів	1	1	1
Обчислення $\sum F_i \cdot W_i$			13

Відповідно, фактор технічної складності становитиме:

$$TCF = 0.6 + 0.01 \sum F_i \cdot W_i = 0.73$$

Далі необхідно встановити фактор середовища, що аналогічно обчислюється за формулою:

$$EF = 1.4 - 0.03 \sum F_i \cdot W_i, \quad (2.3)$$

Де F_i – значення оцінки окремого фактора середовища, а W_i – його вага. Оцінки таких факторів наведені в таблиці 2.6.

Таблиця 2.6 – Оцінка факторів середовища

Назва	Значення	Вага	$F_i \cdot W_i$
Знайомість з процесом розробки	4	1.5	6
Наявність працівників на півставки	0	-1	0
Аналітичні здібності	3	0.5	1.5
Знайомість з типом застосування	5	0.5	2.5
Досвід ООП	4	1	4
Мотивація	5	1	5

Продовження таблиці 2.6

Назва	Значення	Вага	$F_i \cdot W_i$
Складність мови програмування	1	-1	-1
Стабільність вимог	5	2	10
Обчислення $\sum F_i \cdot W_i$			28

Відповідно,

$$EF = 1.4 - 0.03 \sum F_i \cdot W_i = 0.56$$

Урешті-решт, проведемо обчислення узгоджених Use-Case одиниць:

$$UCP = UUCP \cdot TCF \cdot EF = 71.13 \quad (2.4)$$

Для обчислення необхідного часу для виконання проекту, помножимо їх на фактор продуктивності, рівний 20 [14]:

$$E = UCP \cdot 20 = 1422.6 \quad (2.5)$$

Отож, для виконання даної розробки знадобиться 1422.6 робочих годин.

2.6 Постановка завдання на розробку програмного забезпечення

Отож, у рамках виконання даного дипломного проєкту необхідно буде реалізувати застосунок, що надаватиме можливість повноцінного управління користувачами, налаштування відстежуваних нутрієнтів, перегляду каталогу харчових продуктів із пошуковою підтримкою як за назвою, так і за штрих-кодом, створення та управління рецептами, ведення та планування записів споживання, автоматичного формування персоналізованого раціону на основі поставлених нутріційних цілей, а також перегляду звітності й статистики споживання за обрані періоди.

Метою розробки є розширення можливостей персоналізації відслідковування та спрощення мануальної послідовності дій, пов'язаних із

введенням, редагуванням і аналізом даних харчового щоденника за рахунок цифровізації та часткової автоматизації.

Програмне забезпечення повинне виконувати усі поставлені функціональні та нефункціональні вимоги

Висновки до розділу

Отож, у рамках даного розділу було проведено всебічний аналіз вимог до програмного забезпечення, що передбачав побудову діаграми варіантів використання з чіткою ідентифікацією акторів і сценаріїв взаємодії, а також деталізацію кожного з ключових варіантів використання з урахуванням передумов, перетіком подій та очікуваних результатів.

Було описано детальний перелік функціональних вимог, включно з управлінням користувачами, налаштуванням нутрієнтних параметрів, інтегрованим пошуковим механізмом і механізмами формування раціонів, а також описано нефункціональні вимоги до ПЗ. Опісля була сформульована матриця трасування, яка зіставила кожен варіант використання з відповідними функціональними вимогами.

Проведено визначення та обґрунтування рекомендованих конфігурацій технічних ресурсів, необхідних для забезпечення функціонування системи. У рамках аналізу економічних показників було оцінено кількість Use-Case points та на їх основі проведено передбачення часових потреб для розробки застосунку.

Було сформульовано постановку завдання на розробку програмного забезпечення.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

За основу даного застосунку було покладено клієнт-серверний тип архітектури, оскільки така модель організації забезпечує важливе розділення відповідальності формування представлення користувачеві та обробкою даних. Якщо у випадку клієнтської частини можна зупинитися на монолітному підході, то для серверної доцільно розглянути ширший спектр архітектурних рішень.

Отож, даний вебзастосунок реалізовуватиме клієнт-серверну архітектуру з подальшим розділенням серверної частини на мікросервіси з використанням шлюзу API та дотриманням розширеної архітектури для кожного з сервісів. Для більш детального опису архітектури, скористаймося C4 діаграмою [15]. Почнімо з огляду контекстного рівня, зображує загальну картину системи, її користувачів, а також зовнішні сервіси, з якими застосунок інтегрується. На рисунку 3.1 наведено відповідний рівень діаграми.

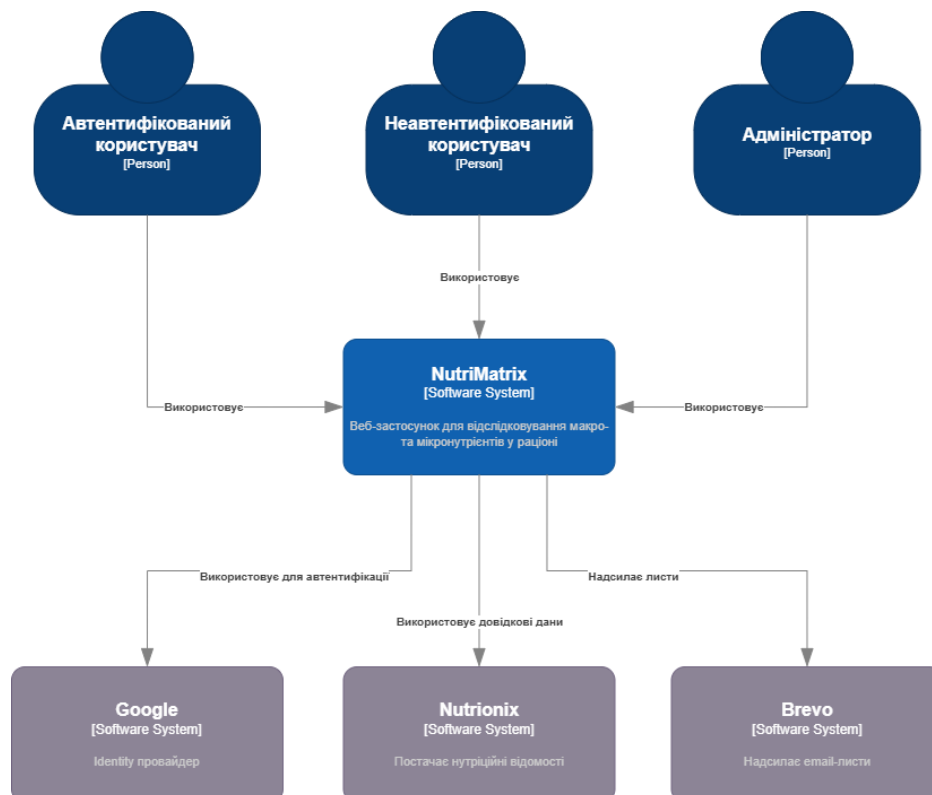


Рисунок 3.1 – Контекстний рівень діаграми C4

Як бачимо, застосунок передбачає взаємодію з трьома типами користувачів – адміністратором, автентифікованим та неавтентифікованим користувачами, а також інтеграцію з трьома сервісами: Nutrionix для забезпечення нутріційних даних, Brevo для надсилання електронних листів, а також Google у якості альтернативного способу автентифікації. Перейдімо до контейнерного рівня – його наведено на кресленні «Схема структурна компонентів програмного забезпечення».

Кожен мікросервіс має свою окрему базу даних. Сервіси, де швидкість отримання даних є надзвичайно важливим, а їхній об'єм є відносно обмеженим, також мають бази даних Redis з використанням Redis.OM для забезпечення належної швидкодії. Сервіс рецептів, що зокрема відповідає за рекомендації, також має векторну БД на базі Qdrant для швидкого отримання кандидатів перед формуванням раціону. Мікросервіси здійснюють між собою асинхронну комунікацію, що здійснюється за допомогою комунікації через шину подій на основі RabbitMQ по протоколу AMQP. API-шлюз відповідає за ізоляцію мікросервісів від клієнта. Клієнт та шлюз комунікують за допомогою протоколу HTTPS.

3.2 Архітектурні рішення та обґрунтування вибору засобів розробки

Як було описано риснками 2.1 та 2.2, застосунок проводить інтеграцію з трьома зовнішніми сервісами:

- Nutrionix. Даний сервіс володіє великим каталогом їжі та екстенсивним переліком нутрієнтів до них. Саме цей сервіс слугує першоджерелом даних. Крім цього, він також надає можливості природньої обробки мови, що дозволяє з тексту обирати нутрієнти, їхні одиниці виміру та кількості. Він надає зручний API-інтерфейс та має як безкоштовний план, що має обмеження за кількістю запитів на день, так і платний, до якого можна буде розширитися у випадку комерціалізації застосунку [16].

- Brevo. Даний сервіс надає послуги SMTP-серверу для розсилки електронних листів. Головною перевагою даного сервісу є наявність

безкоштовного плану, що має обмеження в 300 листів на добу, який, тим не менш, також можна буде згодом замінити платним [17].

- Google Identity Service. Даний сервіс забезпечує автентифікацію користувачів за допомогою облікових записів Google, що спрощує вхід і підвищує рівень безпеки. Працює на основі протоколів OpenID Connect та OAuth 2.0, підтримує делегування доступу, автоматичне оновлення токенів і багатофакторну автентифікацію. Інтеграція здійснюється через зручну клієнтську бібліотеку для вебзастосунків [18].

Застосунок використовує як SQL, так і NoSQL бази даних, використовуючи кожен тип відповідно до специфіки поставлених задач. SQL бази даних використовуються для зберігання структурованих даних великого обсягу, що вимагають дотримання цілісності. У свою чергу NoSQL бд використовуються для двох цілей:

- Для реалізації локального кешування у сервісах, які потребують швидкодії, а дані для них не є занадто великими.
- Для проведення пошуку найближчих сусідів у високовимірних векторних просторах рецептів. Для цього завдання використовується БД окремої категорії – векторна.

Перейдемо до аналізу набору технологічних засобів для реалізації даного проєкту. Задача розробки вебзастосунків є доволі поширеним завданням, і в більшості мов наявні технологічні середовища для створення відповідних доробок. Прикладами є: C# та ASP.NET Core, Java та Spring Framework, Python та Django, тощо. Серед наявних варіантів для серверної частини було обрано саме C#, адже ASP.NET Core забезпечує крос-платформність, високу продуктивність, підтримку багатьох сучасних концепцій, а також має у своїй екосистемі низку корисних модулів, що значно спростять реалізацію додатку [19], як-от ASP.NET Core Identity, що дозволяє швидко налаштовувати автентифікацію користувачів та керувати їхньою інформацією із застосуванням складних алгоритмів хешування чутливих даних [20], EntityFramework Core, що ефективно забезпечує взаємодію з базою

даних [21], та низка інших. Крім того, наявність досвіду роботи з технологією також підвищує привабливість даного варіанту.

Окрім цього дане програмне забезпечення передбачає використання низки сторонніх пакетів:

- MediatR – надає технологічне середовище для реалізації патерна медіатора, який забезпечує слабе зв'язування між компонентами системи. MediatR є основним вибором розробників для реалізації патерну CQRS [22].

- Redis.OM – це бібліотека об'єктно-орієнтованого моделювання, що забезпечує зручну роботу з ключовими структурами NoSQL-сховища у вигляді об'єктів із суворою типізацією. Завдяки інтеграції індексації та підтримки LINQ-запитів, він істотно спрощує реалізацію пошуку та кешування у частинах застосунку, що вимагають належної швидкодії [23].

- Ardalis.Specifications – надає інфраструктурний механізм для реалізації патерну специфікації [24].

- MassTransit – забезпечує високорівневу абстракцію для реалізації асинхронної міжсервісної комунікації через шину подій. Представлений як зручний засіб для інтеграції з брокерами повідомлень, він спрощує реалізацію шаблонів надійної доставки та обробки повідомлень [25].

- Quartz.NET –потужна бібліотека для планування завдань у .NET-додатках, яка дозволяє запускати задачі у визначений час або з певною періодичністю. Наявні потужні можливості для гнучкої конфігурації, що включає в себе підтримку CRON-виразів, відкладених запусків та повторень [26].

У якості IDE було обрано Visual Studio 2022, адже вона має безкоштовну версію та пропонує потужні інструменти та розширення, що значно спрощують процес розробки. Дане програмне забезпечення має зручний інтерфейс, технологію IntelliSense, що допомагає завершувати код, вбудовану сумісність з Git і зручний налагоджувач, що зокрема дозволяє легко випробовувати контейнеризовані мікросервісні додатки [27].

Для клієнтської частини було обрано саме TypeScript, адже він є основною мовою технологічного середовища Angular. Angular у свою чергу чудово підходить для швидкої розробки невеликих SPA, а також має низку вкрай-необхідних інструментів, що значно спрощують розробку, але не вимагають складної конфігурації, як-от HTTP-перехоплювачі, авторизація доступу до окремих адрес, тощо [28].

У якості IDE для розробки клієнтської частини було обрано WebStorm 2023.2.5, оскільки дане програмне забезпечення володіє вбудованою підтримкою для проектів із використанням Angular, що значно підвищує продуктивність розробника, та включає інтелектуальне доповнення коду, наприклад, під час імпорту модулів, навігацію та інструменти рефакторингу, що зокрема дозволяють безпроблемно перейменовувати класи та змінювати шляхи до файлів. Наявна й вбудована підтримка TypeScript, що також значно спрощує процес кодування [29].

3.3 Конструювання програмного забезпечення

3.3.1 Розробка алгоритму рекомендації плану харчування

Отож, завдання полягає у відборі такої комбінації рецептів і відповідних їм кількостей порцій, яка б відповідала заданим користувачем критеріям та забезпечувала максимальне наближення до визначених нутріційних цілей.

Першим кроком даного алгоритму, вочевидь, є фільтрація доступних рецептів згідно з критеріями, встановленими користувачем для кожної позиції страви. До таких критеріїв, зокрема, належать обмеження на вміст або відсутність певних інгредієнтів, а також відповідність рецепту визначеній категорії страв.

Після даної фільтрації список кандидатів для кожної позиції, природньо, звужиться. Тим не менш, завдання відповідних обмежень не є обов'язковим, а отже нерідко до розгляду може потрапляти повна сукупність публічних рецептів, обсяг якої може потенційно сягати 100 тисяч записів. Використання

такої значної вибірки кандидатів під час пошуку найкращої комбінації значно знизить швидкодію алгоритму, що може бути неприпустимим.

Тому перед проведенням безпосереднього пошуку було вирішено впровадити етап попереднього формування підмножини рецептів-кандидатів, які з найбільшою ймовірністю відповідатимуть поставленим нутріційним цілям. Для цього було вирішено використовувати метод найближчих сусідів, що полягає у виборі таких записів, для яких метрика відстані від заданого вектору буде найменшою.

Тоді як дана концепція доволі проста, її реалізація має свою специфіку. Річ у тім, що мануальний пошук таких сусідів займає занадто багато часу й може доходити до секунд, тоді як відповідь користувачу повинна поступати швидко. Тому варто використовувати технологічні засоби, що спеціалізуються на цьому – одним із них є векторні бази даних, що дозволяють зберігати вектори та індексують їх у відповідності до заданих метрик відстані: стандартно присутніми в них є Манхеттенська, Евклідова та косинусна. Але виникає інша проблема – вбудована можливість обирати які саме виміри векторів (тобто які саме нутрієнти) будуть враховуватися для обчислення відстані відсутня: доведеться або будувати індекси для всіх можливих комбінацій нутрієнтів, що при 161 нутрієнту не є можливим, або розглянути інший шлях досягнення цього. Для подолання даної проблеми було прийнято рішення застосовувати косинусу відстань, а вектори нутрієнтів перед внесенням до бази даних піддавати одиничній нормалізації:

$$\hat{v} = \frac{v}{\|v\|} \quad (3.1)$$

Відповідно, якщо у векторі пошуку встановити 0 на місці нутрієнтів, які не становлять інтерес, єдиними факторами, що впливатимуть на косинусну відстань будуть лише ті, що цікавлять користувача, а саме значення косинусної подібності набуде вигляду:

$$d_{cos}(u, v) = 1 - \frac{u \cdot v}{\|u\| \cdot \|v\|} = 1 - u \cdot v \quad (3.2)$$

Таким чином після відбору кандидатів до кожної позиції можна починати пошук найкращої комбінації – це задача мінімізації функції похибки, що визначається як нормалізована евклідова відстань нутрієнтів нинішньої комбінації до цільового вектора нутрієнтів:

$$d(x, y) = \sqrt{\sum_{i=1}^n \frac{(x_i - y_i)^2}{y_i^2}} \quad (3.3)$$

В основу реалізованого механізму пошуку такої комбінації було покладено генетичний алгоритм. Псевдокод результуючого алгоритму пошуку рекомендації представлено на рисунках 3.2 – 3.7.

Algorithm 2 CreateRandomIndividual

Input: candidatesList (list of lists), numSlots (int)
Output: Individual

```

1: indiv.Genes ← array of size numSlots
2: for i = 0 to numSlots - 1 do
3:   slotCandidates ← candidatesList[i]
4:   recipe ←  $\overset{R}{\leftarrow}$  {r ∈ slotCandidates}
5:   amount ←  $\overset{R}{\leftarrow}$  [1..2 × recipe.Servings]
6:   indiv.Genes[i] ← Gene(Recipe = recipe, Amount = amount)
7: end for
8: return indiv

```

Рисунок 3.2 – Псевдокод методу CreateRandomIndividual

Algorithm 3 EvaluateFitness

Input: indiv (Individual), expectations (dictionary)
Output: fitness (float)

```

1: currentNutrientValues ← empty dictionary
2: for all g in indiv.Genes do
3:   recipe ← g.Recipe
4:   amount ← g.Amount
5:   if recipe == null then continue
6:   end if
7:   for all (nutrientId, nutrientAmt) in recipe.NutrientAmounts do
8:     contrib ← nutrientAmt × amount / recipe.Servings
9:     current[nutrientId] ← current.get(nutrientId, 0) + contrib
10:  end for
11: end for
12: sumSq ← 0
13: for all k in expectations.Keys do
14:   expVal ← expectations[k]
15:   actual ← current.get(k, 0)
16:   diff ← expVal - actual
17:   sumSq ← sumSq + (diff2 / (expVal2))
18: end for
19: return  $\sqrt{\text{sumSq}}$ 

```

Рисунок 3.3 – Псевдокод методу EvaluateFitness

Algorithm 4 TournamentSelect**Input:** population (list of Individuals), tournoiSize (int)**Output:** selected (Individual)

```

1: best  $\leftarrow$  null
2: for  $i = 1$  to tournoiSize do
3:   candidate  $\xleftarrow{R} \{x \in \text{population}\}$ 
4:   if best == null or candidate.Fitness < best.Fitness then
5:     best  $\leftarrow$  candidate
6:   end if
7: end for
8: return best

```

Рисунок 3.4 – Псевдокод методу TournamentSelect

Algorithm 5 Crossover**Input:** parent1 (Individual), parent2 (Individual), numSlots (int)**Output:** child (Individual)

```

1: child.Genes  $\leftarrow$  array of size numSlots
2: crossPoint  $\xleftarrow{R} [1..numSlots - 1]$ 
3: for  $i = 0$  to numSlots - 1 do
4:   if  $i < \text{crossPoint}$  then
5:     source  $\leftarrow$  parent1
6:   else
7:     source  $\leftarrow$  parent2
8:   end if
9:   child.Genes[i]  $\leftarrow$  Gene(Recipe = source.Genes[i].Recipe, Amount =
    source.Genes[i].Amount)
10: end for
11: return child

```

Рисунок 3.5 – Псевдокод методу Crossover

Algorithm 6 Mutate**Input:** indiv (Individual), candidatesList (list of lists)**Output:** void

```

1: recipeMutProb  $\leftarrow$  0.10
2: amountMutProb  $\leftarrow$  0.20
3: for  $i = 0$  to indiv.Genes.Length - 1 do
4:   if with probability recipeMutProb then
5:     slotCandidates  $\leftarrow$  candidatesList[i]
6:     newRecipe  $\xleftarrow{R} \{r \in \text{slotCandidates}\}$ 
7:     indiv.Genes[i].Recipe  $\leftarrow$  newRecipe
8:     indiv.Genes[i].Amount  $\xleftarrow{R} [1..2 \times \text{newRecipe.Servings}]$ 
9:   else if with probability amountMutProb and indiv.Genes[i].Recipe  $\neq$ 
    null then
10:    maxAmt  $\leftarrow$  indiv.Genes[i].Recipe.Servings
11:    indiv.Genes[i].Amount  $\xleftarrow{R} [1..2 \times \text{maxAmt}]$ 
12:   end if
13: end for

```

Рисунок 3.6 – Псевдокод методу Mutate

Algorithm 1 GetRecommendation**Input:** RecipeRequests (list), NutritionalGoals (dictionary)**Output:** RecommendationResponseDto

```

1: timeLimit  $\leftarrow$  7seconds
2: vector  $\leftarrow$  [ NutritionalGoals[k] if  $k \in$  NutritionalGoals else 0
   for each  $k$  in possible nutrinet ids ]
3: candidatesList  $\leftarrow$  empty list of lists
4: for  $i = 0$  to RecipeRequests.Count - 1 do
5:   excludeIds  $\leftarrow$  all Ids already in candidatesList
6:   req  $\leftarrow$  RecipeRequests[i]
7:   candidateIds  $\leftarrow$  300 nearest neighbors who match the recipe criteria
   and are not in excludeIds
8:   candidates  $\leftarrow$  empty list
9:   for all id in candidateIds do
10:    item  $\leftarrow$  {  $r \in$  _recipesCollection |  $rId = id$  }
11:    if item  $\neq$  null then
12:      Add item to candidates
13:    end if
14:  end for
15:  Add candidates to candidatesList
16: end for
17: populationSize  $\leftarrow$  50
18: numSlots  $\leftarrow$  RecipeRequests.Count
19: population  $\leftarrow$  empty list of Individuals
20: bestIndividual  $\leftarrow$  null
21: for  $i = 1$  to populationSize do
22:   indiv  $\leftarrow$  CreateRandomIndividual(candidatesList, numSlots)
23:   indiv.Fitness  $\leftarrow$  EvaluateFitness(indiv, NutritionalGoals)
24:   Add indiv to population
25:   if bestIndividual == null or indiv.Fitness <
     bestIndividual.Fitness then
26:     bestIndividual  $\leftarrow$  indiv
27:   end if
28: end for
29: while timeLimit not reached do
30:   newPopulation  $\leftarrow$  empty list
31:   for  $i = 1$  to populationSize do
32:     parent1  $\leftarrow$  TournamentSelect(population, 3)
33:     parent2  $\leftarrow$  TournamentSelect(population, 3)
34:     child  $\leftarrow$  Crossover(parent1, parent2, numSlots)
35:     Mutate(child, candidatesList)
36:     child.Fitness  $\leftarrow$  EvaluateFitness(child, NutritionalGoals)
37:     Add child to newPopulation
38:     if child.Fitness < bestIndividual.Fitness then
39:       bestIndividual  $\leftarrow$  child
40:     end if
41:     if timeLimit reached then
42:       break
43:     end if
44:   end for
45:   population  $\leftarrow$  newPopulation
46: end while
47: bestCombo  $\leftarrow$  [( $g$ .Recipe,  $g$ .Amount) |  $g \in$  bestIndividual.Genes]
48: return bestCombo

```

Рисунок 3.7 – Псевдокод метода GetRecommendation

3.3.2 Опис структури бази даних

У даному застосунку фігурує 4 реляційні бази даних. В якості СУБД для сервісу управління користувачами використовується PostgreSQL. База даних сервісу призначена для зберігання користувачів, а також даних про їхні внутрішні цілі. Опис таблиць бази даних наведено у таблицях 3.1-3.9. Модель бази даних наведена на рисунку 3.5.

Таблиця 3.1 – Опис таблиці _EFMigrationHistory

Назва поля	Тип даних	Опис
MigrationId	varchar	ідентифікаційний номер міграції
ProductVersion	varchar	Версія застосунку

Таблиця 3.2 – Опис таблиціAspNetRoleClaims

Назва поля	Тип даних	Опис
Id	int	ідентифікаційний номер
RoleId	varchar	ідентифікаційний номер ролі
ClaimType	varchar	Тип клейму
ClaimValue	varchar	Значення клейму

Таблиця 3.3 – Опис таблиціAspNetRoles

Назва поля	Тип даних	Опис
Id	varchar	ідентифікаційний номер
Name	varchar	Назва ролі
NormalizedName	varchar	Нормалізована назва ролі
ConcurrencyStamp	varchar	Значення для запобігання конфліктів оновлень значень

Таблиця 3.4 – Опис таблиці AspNetUserClaims

Назва поля	Тип даних	Опис
Id	int	ідентифікаційний номер
UserId	varchar	Назва ролі
ClaimType	varchar	Тип клейму
ClaimValue	varchar	Значення клейму

Таблиця 3.5 – Опис таблиці AspNetUserLogins

Назва поля	Тип даних	Опис
LoginProvider	varchar	Назва зовнішнього провайдера входу
ProviderKey	varchar	Унікальний ідентифікатор провайдера
ProviderDisplayName	varchar	Представлення назви провайдера
UserId	varchar	Ідентифікатор користувача

Таблиця 3.6 – Опис таблиці AspNetUserRoles

Назва поля	Тип даних	Опис
UserId	varchar	Ідентифікатор користувача
RoleId	varchar	Ідентифікатор ролі

Таблиця 3.7 – Опис таблиці AspNetUserLogins

Назва поля	Тип даних	Опис
Id	varchar	Ідентифікатор користувача

Продовження таблиці 3.6

Назва поля	Тип даних	Опис
UserName	varchar	Ім'я користувача
NormalizedUser	varchar	Нормалізоване ім'я користувача
Email	varchar	Пошта корисутвача
NormalizedEmail	varchar	Нормалізована пошта користувача
EmailConfirmed	boolean	Чи підтверджено пошту
PasswordHash	varchar	Хешоване значення паролю
SecurityStamp	varchar	Значення для запобігання конфліктів оновлень значень
ConcurrencyStamp	varchar	Значення для запобігання конфліктів оновлень значень
PhoneNumber	varchar	Номер телефону
PhoneNumberCo	boolean	Чи підтверджено номер телефону
TwoFactorEnabled	boolean	Чи увімкнено двофакторну верифікацію
LockoutEnd	timestampz	Коли закінчиться блокування облікового запису користувача
LockoutEnabled	boolean	Чи заблоковано користувача
AccessFailedCount	boolean	Чи рахувати невдалі спроби увійти в акаунт

Таблиця 3.8 – Опис таблиці AspNetUserTokens

Назва поля	Тип даних	Опис
UserId	varchar	Ідентифікатор користувача

Продовження таблиці 3.8

Назва поля	Тип даних	Опис
LoginProvider	varchar	Назва зовнішнього провайдеру
Value	varchar	Значення токена

Таблиця 3.9 – Опис таблиці AspNetUserTokens

Назва поля	Тип даних	Опис
Id	bigint	Ідентифікатор запису
UserId	varchar	Ідентифікатор користувача
IsActive	boolean	Чи відслідковується нутрієнт
NutrientId	bigint	Ідентифікатор нутрієнта
Amount	real	Цільова кількість нутрієнта

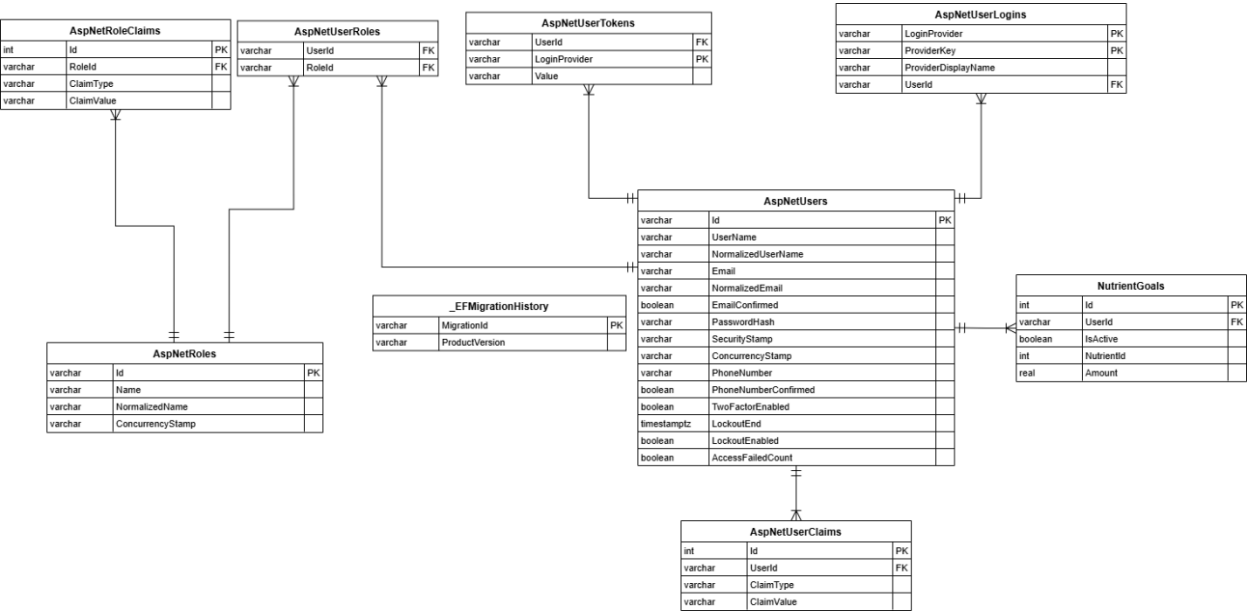


Рисунок 3.5 – ER діаграма БД сервісу управління користувачами

В якості СУБД для сервісу каталогу їжі використовується PostgreSQL. База даних сервісу призначена для зберігання їжі, приватних рецептів, а також

даних про них. Опис таблиць бази даних наведено у таблицях 3.10-3.15. Модель бази даних наведена на рисунку 3.6.

Таблиця 3.10 – Опис таблиці Food

Назва поля	Тип даних	Опис
Id	bigint	Ідентифікатор продукту
Name	varchar	Назва продукту
Barcode	varchar	Штрих-код продукту
Photo	varchar	Адреса фото продукту
IsDeleted	boolean	Чи видалено продукт

Таблиця 3.11 – Опис таблиці Measure

Назва поля	Тип даних	Опис
Id	bigint	Ідентифікатор міри
Name	varchar	Назва одиниці виміру
WeightInGrams	real	Вага в грамах міри
FoodId	bigint	Ідентифікатор асоційованого продукту
IsDeleted	boolean	Чи видалено одиницю виміру

Таблиця 3.12 – Опис таблиці Nutrient

Назва поля	Тип даних	Опис
Id	bigint	Ідентифікатор нутрієнта
Name	varchar	Назва нутрієнта

Продовження таблиці 3.12

Назва поля	Тип даних	Опис
Unit	varchar	Одиниця виміру
IsDeleted	boolean	Чи видалено нутрієнт

Таблиця 3.13 – Опис таблиці FoodNutrientIn100g

Назва поля	Тип даних	Опис
Id	bigint	Ідентифікатор запису
FoodId	bigint	Ідентифікатор продукту
NutrientId	bigint	Ідентифікатор нутрієнта
Amount	real	Кількість нутрієнта в 100 грамах продукту
IsDeleted	boolean	Чи видалено запис

Таблиця 3.14 – Опис таблиці Meal

Назва поля	Тип даних	Опис
Id	bigint	Ідентифікатор страви
Name	varchar	Назва страви
TotalServings	real	Кількість порцій страви, що готується вказаними кількостями інгредієнтів
AddedBy	varchar	Ідентифікатор користувача, який додав страву
IsDeleted	boolean	Чи видалено страву

Таблиця 3.15 – Опис таблиці FoodMeal

Назва поля	Тип даних	Опис
Id	bigint	Ідентифікатор запису
MeasureId	bigint	Ідентифікатор міри
MealId	bigint	Ідентифікатор страви
Quantity	real	Кількість одиниці виміру інгредієнта для повної кількості порцій страви
IsDeleted	boolean	Чи видалено запис

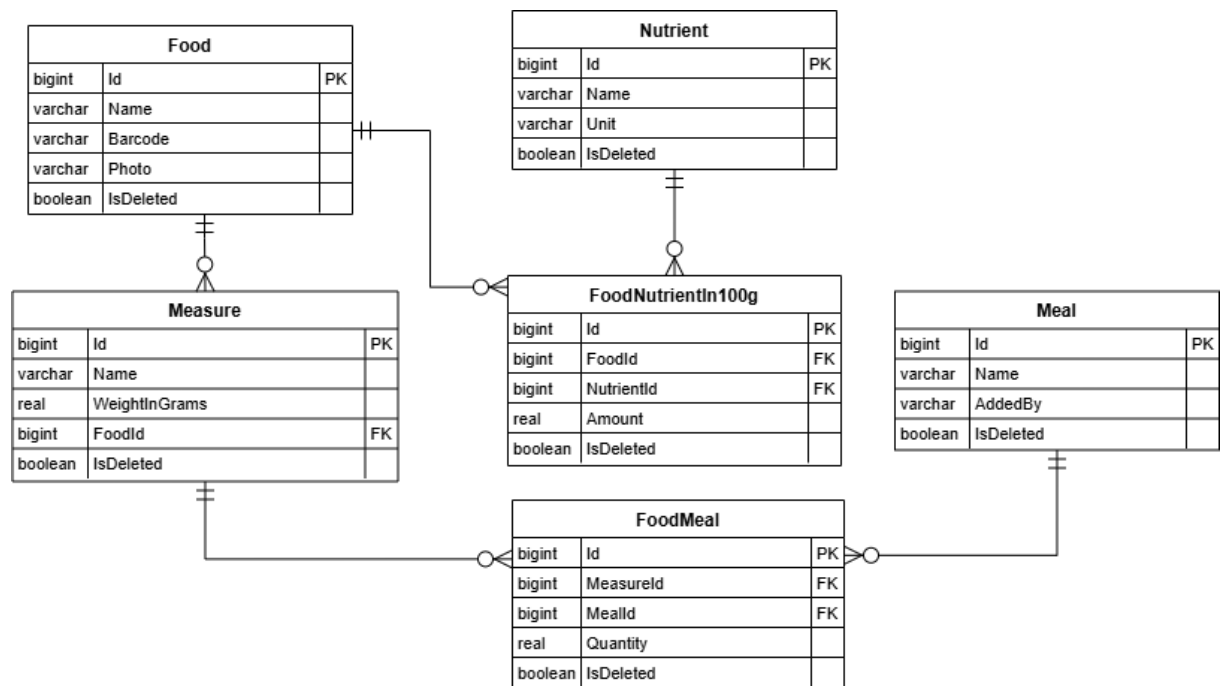


Рисунок 3.6 – ER діаграма БД сервісу каталогізації їжі

В якості СУБД для сервісу управління записами споживання використовується PostgreSQL. База даних сервісу призначена для зберігання записів споживання їжі, рецептів, а також планування додавання записів. Опис таблиць бази даних наведено у таблицях 3.16-3.19. Модель бази даних наведена на рисунку 3.7.

Таблиця 3.16 – Опис таблиці MealRecord

Назва поля	Тип даних	Опис
Id	bigint	Ідентифікатор запису рецепту
DateEaten	timestamp	Дата споживання
ServingsEaten	real	Кількість спожитих порцій страви
UserId	varchar	Ідентифікатор користувача
MealId	bigint	Ідентифікатор з'їденої страви
IsDeleted	boolean	Чи видалено запис

Таблиця 3.17 – Опис таблиці IngredientSnapshot

Назва поля	Тип даних	Опис
Id	bigint	Ідентифікатор знімку інгредієнта
FoodMeasureId	bigint	Ідентифікатор міри інгредієнта
Amount	real	Кількість міри інгредієнта
RecipeRecordId	bigint	Ідентифікатор запису споживання рецепту
IsDeleted	boolean	Чи видалено запис

Таблиця 3.18 – Опис таблиці FoodRecord

Назва поля	Тип даних	Опис
Id	bigint	Ідентифікатор запису споживання продукту
DateEaten	timestamp	Дата споживання

Продовження таблиці 3.18

Назва поля	Тип даних	Опис
UserId	varchar	Ідентифікатор користувача
FoodMeasureId	bigint	Ідентифікатор міри продукту
Amount	real	Кількість спожитого продукту
IsDeleted	boolean	Чи видалено запис

Таблиця 3.19 – Опис таблиці FoodPlan

Назва поля	Тип даних	Опис
Id	bigint	Ідентифікатор плану
UserId	bigint	Ідентифікатор користувача
Name	varchar	Назва плану
ConsummableId	bigint	Ідентифікатор міри продукту
Amount	real	Кількість продукту
RunAtUtc	timestamp	Запланований час
CronExpression	varchar	Вираз, що специфікує час виконання повторюваної задачі
RequiresConfirmation	boolean	Чи потрібне підтвердження перед додаванням
IsRecurring	boolean	Чи повторювана задача
JobKey	varchar	Ідентифікатор асоційованої задачі в Quartz.Net
TriggerKey	varchar	Ідентифікатор тригера асоційованої задачі в Quartz.Net

Продовження таблиці 3.19

Назва поля	Тип даних	Опис
ConsumableType	int	Тип споживаної їжі (продукт чи страва)
IsDeleted	boolean	Чи видалено

Таблиця 3.20 – Опис таблиці PendingRecord

Назва поля	Тип даних	Опис
Id	bigint	Ідентифікатор непідтвердженого запису споживання
DatePending	timestamp	Дата споживання
UserId	varchar	Ідентифікатор користувача
ConsumableId	bigint	Ідентифікатор спожитої їжі
ConsumableType	int	Тип споживаної їжі (продукт чи страва)
Amount	real	Кількість спожитого продукту
IsDeleted	boolean	Чи видалено запис

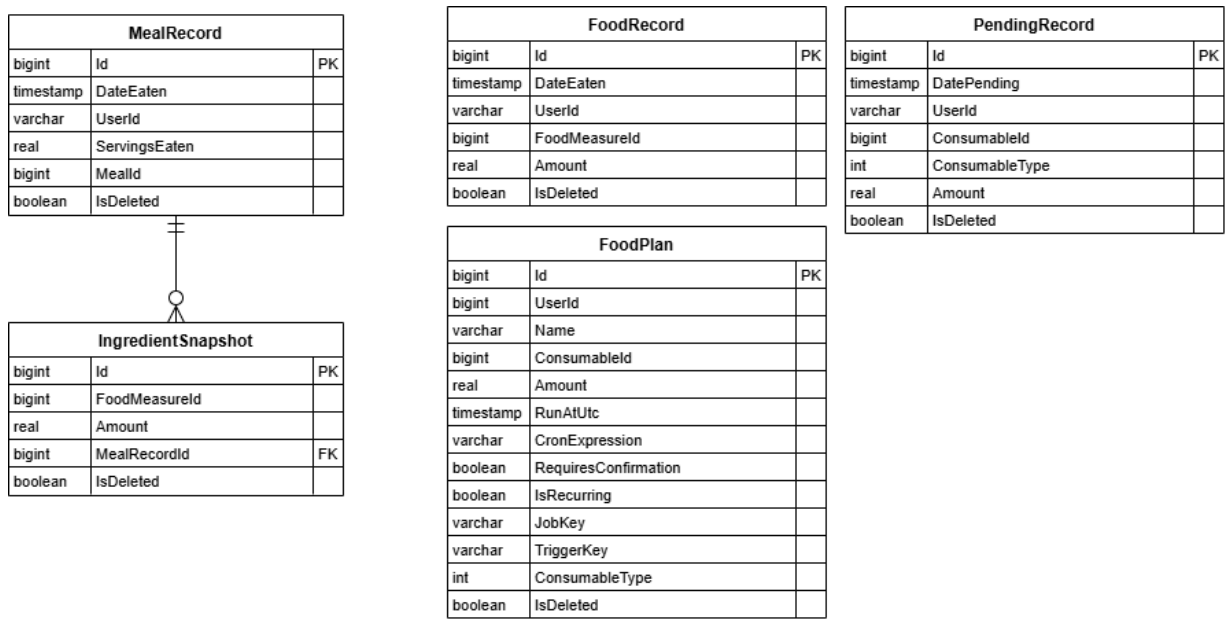


Рисунок 3.7 – ER діаграма БД сервісу управління записами споживання

В якості СУБД для сервісу рецептів та рекомендацій використовується PostgreSQL. База даних сервісу призначена для зберігання рецептів, а також інформації про них. Опис таблиць бази даних наведено у таблицях 3.21-3.23. Модель бази даних наведена на рисунку 3.8.

Таблиця 3.21 – Опис таблиці Recipe

Назва поля	Тип даних	Опис
Id	bigint	Ідентифікатор рецепту
Category	varchar	Категорія рецепту
Servings	real	Кількість порцій
Description	varchar	Опис рецепту
Directions	varchar	Інструкції приготування
IsDeleted	boolean	Чи видалено рецепт

Таблиця 3.22 – Опис таблиці RecipeMeasure

Назва поля	Тип даних	Опис
Id	bigint	Ідентифікатор міри рецепту
MeasureId	bigint	Ідентифікатор одиниці вимірювання
FoodId	bigint	Ідентифікатор продукту
Amount	real	Кількість
RecipeId	bigint	Ідентифікатор рецепту
IsDeleted	boolean	Чи видалено запис

Таблиця 3.23 – Опис таблиці NutrientAmount

Назва поля	Тип даних	Опис
Id	bigint	Ідентифікатор нутріційного показника рецепту
RecipeId	bigint	Ідентифікатор рецепту
NutrientId	int	Ідентифікатор нутрієнта
Amount	real	Кількість нутрієнта в рецепті
IsDeleted	boolean	Чи видалено запис

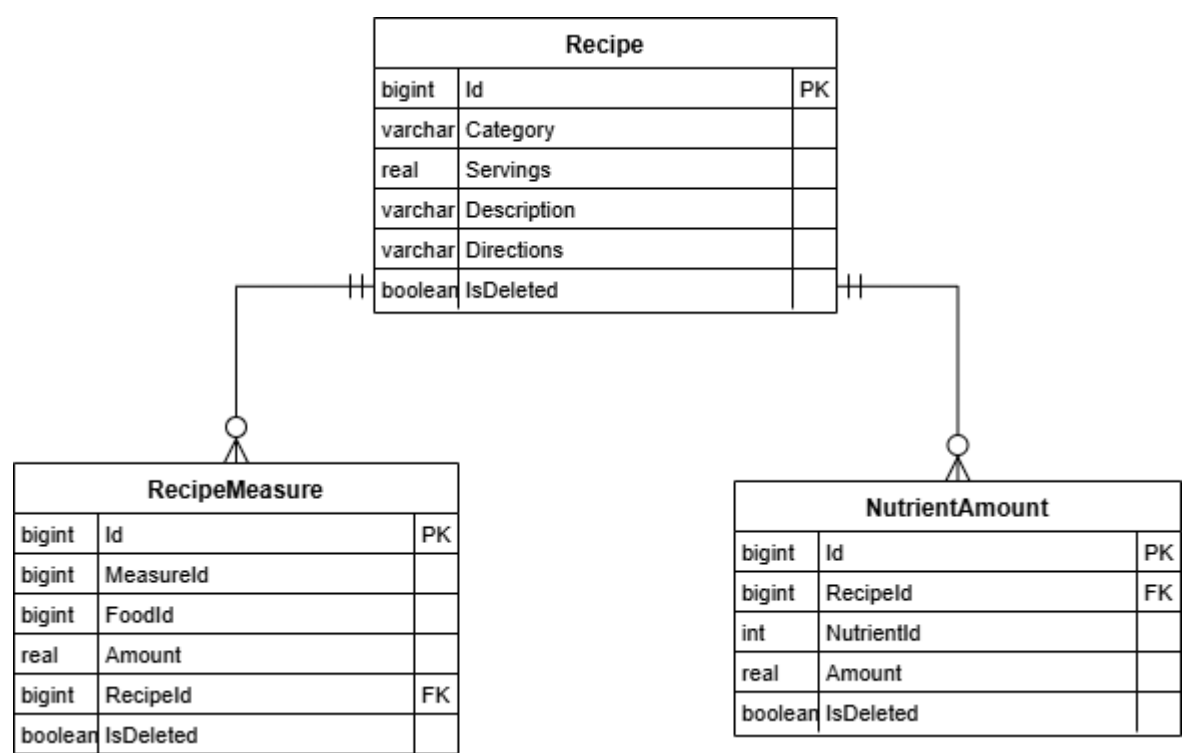


Рисунок 3.8 – ER діаграма БД сервісу рецептів та рекомендацій

Під час розробки програмного забезпечення використовувалася низка утиліт, бібліотек та стороннього програмного забезпечення – їхній опис наведено в таблиці 3.24.

Таблиця 3.24 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Visual Studio 2022	Середовище розробки програмного забезпечення серверної частини дипломного проєкту.
2	ASP.Net Core	Технологічне середовище, що становить основу для розробки API
3	ASP.NET Core Identity	Технологічне середовище, що забезпечуватиме створення облікових записів користувачів, їхнє управління та автентифікацію

Продовження таблиці 3.24

№ п/п	Назва утиліти	Опис застосування
4	EntityFramework Core	Технологічне середовище, що забезпечує взаємодію з базою даних
5	Quartz.Net	Засіб, що дозволяє створювати повторювані заплановані дії
6	Automapper	Засіб для створення методів проекції з одного класу в інший
7	MediatR	Технологічне середовище для реалізації патерну CQRS
8	Redis.OM	Бібліотека об'єктно-орієнтованого моделювання, що забезпечує зручну роботу з Redis, представляючи записи у вигляді об'єктів із суворою типізацією
9	Ardalis.Specifications	Бібліотека, що забезпечує інфраструктурний механізм для реалізації патерну специфікації
10	MassTransit	Бібліотека для реалізації асинхронної міжсервісної комунікації через шину подій та інтеграції з брокерами повідомлень
11	WebStorm 2023.2.5	Середовище розробки програмного забезпечення клієнтської частини дипломного проєкту
12	Angular	Технологічне середовище для створення односторінкових застосунків

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

3.4 Аналіз безпеки даних

Вебзастосунок передбачає підтримку авторизації для ролей звичайного користувача та адміністратора. Відповідно, база даних зобов'язана зберігати в собі дані облікових записів, у тому числі паролі. Паролі становлять чутливу інформацію, отож, для них необхідно застосовувати складні алгоритми односпрямованого хешування, що у разі витoku не дозволять зловмиснику легко отримати значення без перебору варіантів.

Для управління обліковими записами користувачів у даній курсовій роботі було використано технологічне середовище ASP.NET Core Identity, котре, крім усього, використовує вбудований алгоритм хешування паролів – HMAC-SHA256 із застосуванням PBKDF2.

PBKDF2 є однією з найпоширеніших функцій формування ключа хешування та надає значний захист від можливих вразливостей за рахунок комплексного алгоритму, що має значну вартість обчислень. Функція PBKDF2 зазвичай приймає в себе власне значення, для якого знаходитиметься ключ, криптографічну сіль – випадковий набір даних, що потім використовуватиметься в алгоритмі, псевдо-випадкову функцію, котра використовуватиметься багаторазово під час формування ключа, необхідну кількість ітерацій, а також довжину необхідного ключа [30].

HMAC-SHA256 це тип одностороннього хеш-алгоритму з використанням ключа, який побудований на основі хеш-функції SHA-256 і включає в себе комбінування секретного ключа із даними повідомлення, хешування результату за допомогою хеш-функції, повторне змішування хеш-значення з ключем, і врешті застосування хеш-функції вдруге. Вихідний хеш має довжину 256 біт, що мінімізує вірогідність колізій [31].

Окрім цього у застосунку є окремі ресурси, що асоціюються з конкретними користувачами: дії з акаунтом, записи споживання, рецепти, тощо. Отож, необхідно забезпечити механізми попередження доступу до цих ресурсів несанкціонованими користувачами – для цього було додатково впроваджено модель ресурсно-орієнтованої авторизації, що за вимоги додатково перевіряє відповідність ідентифікатора користувача ідентифікатору особи, що є власником/асоціюється з ресурсом.

Для забезпечення захисту вразливостей витіку чутливих даних котнейнерів застосунку, було також реалізовано використання конфігурації User Secrets для вразливих даних: ключів доступу до API Nutrionix, Google Identity Services та інших.

З метою запобігання потенційним вразливостям, пов'язаним із перехопленням або підміною даних у процесі обміну між клієнтом та сервером, використовується захищений протокол передачі — HTTPS, який забезпечує додаткове шифрування транспортного рівня та верифікацію автентичності сервера.

Висновки до розділу

Отож, у даному розділі було проведено моделювання та конструювання програмного забезпечення, що включало в себе опис архітектури застосунку за допомогою C4 діаграми із заглибленням у контекстний та контейнерний рівні для серверної та клієнтської частин.

Надано структурований опис ключових сутностей для кожної із баз даних мікросервісів з використанням ER-діаграми та зазначенням атрибутів та зв'язків кожної з них. Сформульовано докладний опис семантики полів та типів даних для кожної сутності. Описано специфіку впровадження алгоритму надання рекомендацій харчування.

Проаналізовано засоби убезпечення чутливої інформації – паролів користувачів – в базі даних для обраної технологічного середовища ASP.Net Core Identity та деталізовано принцип роботи алгоритму одностороннього

хешування, що застосовується. Специфіковано впровадження ресурсно-орієнтованого способу авторизації та описано протокол комунікації між клієнтом та сервером.

Розроблено програмне забезпечення на основі вказаних специфікацій.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Як засіб перевірки якості ПЗ було обрано статичний аналізатор коду Embold.io – даний сервіс широко застосовується в практиці аналізу якості програмного забезпечення для виявлення помилок, вразливостей та інших проблемних аспектів у структурі коду [32]. Дана платформа має інтеграцію з Github, що втілюється в можливості аналізу коду з репозиторію.

Метриками для оцінки якості ПЗ було обрано наступні:

- Оцінка функціональності: характеризує здатність програмного забезпечення відповідати визначеним функціональним вимогам
- Підтримуваність: визначає легкість та швидкість, з якими код може бути модифікований або розширений.
- Стійкість: характеризує здатність системи або алгоритму ефективно реагувати на непередбачені помилки під час функціонування системи.
- Зручність використання: визначає здатність змінювати чи лагодити код застосунку.
- Ефективність: відображає ефективність використання ресурсів системою.
- Надійність: характеризує частоту виникнення збоїв, здатність витримувати відмови окремих компонентів.
- Концептуальна цілісність: визначає узгодженість і послідовність загальної архітектури системи.
- Безпечність: визначає ступінь захищеності програмне забезпечення від шкідливих атак та загроз.

Результати аналізу продемонстровано на рисунку 4.1.

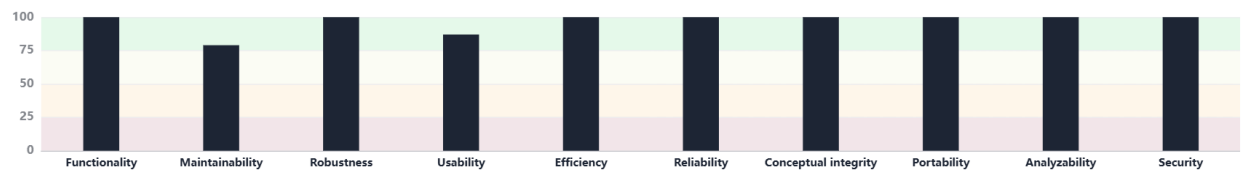


Рисунок 4.1 – Результат статичного аналізу

Як демонструє стовпчикова діаграма, за переважною більшістю метрик було здобуто задовільні показники, виняток становлять лише показники підтримуваності та зручності використання — за ними було зафіксовано значення 79 % та 86 % відповідно.

У відповідності до поставлених нефункціональних вимог, клієнтську частину було реалізовано за допомогою технологічного середовища Angular, а серверну – з використанням ASP.NET Core та його допоміжних засобів. Валідація даних як на клієнтській, так і на серверній частинах разом з низкою інших заходів забезпечують контроль введення інформації та захист від некоректних дій користувача. Заходи для дотримання встановлених вимог безпеки застосунку було наведено в розділі 3.4. Будь-яка комунікація між мікросервісами здійснювалася асинхронно.

4.2 Опис процесів тестування

Тестування результуючого програмного забезпечення виконується згідно документу «Програма та методика тестування». Випробування було здійснено шляхом мануального тестування системи в контексті низки тест-кейсів – їх наведено в таблицях 4.1 – 4.14.

Таблиця 4.1 – Тест 1.1 Реєстрація користувача

Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Електронна пошта, ім'я, пароль

Продовження таблиці 4.1

Опис проведення тесту	У відповідні поля вводяться: коректна електронна пошта, яка до цього не була зареєстрована в системі, пароль від б, який містить хоча б з одну англійську літеру й одне число, а також пароль. Опісля натискається кнопка підтвердження реєстрації.
Очікуваний результат	Реєстрація проходить успішно, користувач реєструється в системі і перенаправляється на сторінку автентифікації.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.2 – Тест 1.2 Автентифікація користувача

Початковий стан системи	Користувач автентифікований знаходиться на сторінці входження в обліковий запис
Вхідні дані	Електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться: коректна електронна пошта, яка до цього не була зареєстрована в системі, а також пароль, що попередньо був вказаний під час реєстрації. Опісля натискається кнопка входження в обліковий запис.
Очікуваний результат	Автентифікація проходить успішно, користувач входить у систему й перенаправляється на домашню сторінку.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.3 – Тест 1.3 Додавання запису споживання їжі

Початковий стан системи	Користувач автентифікований і знаходиться на домашній сторінці
Вхідні дані	Найменування їжі, найменування одиниці виміру, спожита кількість

Продовження таблиці 4.3

Опис проведення тесту	Натискається кнопка “Add Food Record”, після чого відкривається модальне вікно для вибору їжі. У поле пошукового запиту вводиться найменування спожитої їжі, після чого перелік результатів пошуку оновлюється, і серед них обирається бажаний, шляхом натискання на нього. Опісля вводиться кількість спожитих одиниць, а також самі одиниці, та натискається кнопка збереження.
Очікуваний результат	Додавання запису проходить успішно, модальне вікно закривається, а на домашній сторінці відображається новий запис.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.4 – Тест 1.4 Додавання запису споживання рецепту

Початковий стан системи	Користувач автентифікований і знаходиться на домашній сторінці
Вхідні дані	Найменування рецепту, спожита кількість
Опис проведення тесту	Натискається кнопка “Add Recipe Record”, після чого відкривається модальне вікно для вибору рецепту. У поле пошукового запиту вводиться найменування спожитого рецепту, після чого перелік результатів пошуку оновлюється. Потрібний рецепт обирається шляхом натискання на нього. Далі вводиться спожита кількість, після чого натискається кнопка збереження.
Очікуваний результат	Додавання запису проходить успішно, модальне вікно закривається, а на домашній сторінці відображається новий запис споживання рецепту.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.5 – Тест 1.5 Додавання особистого рецепту

Початковий стан системи	Користувач автентифікований і на сторінці створення особистого рецепту
Вхідні дані	Назва рецепту, загальна кількість порцій, інгредієнти з назвами, одиницями виміру та кількістю
Опис проведення тесту	У відповідні поля вводиться назва рецепту та кількість порцій. В секції додавання інгредієнтів у пошуку вводиться назва, обирається інгредієнт, одиниця виміру та кількість, після чого натискається кнопка додавання до списку. Аналогічно додаються інші інгредієнти. Урешті-решт, натискається кнопка збереження рецепту.
Очікуваний результат	Рецепт з інгредієнтами додається успішно, після чого відбувається переадресація на сторінку перегляду особистих рецептів
Фактичний результат	Збігається з очікуванням.

Таблиця 4.6 – Тест 1.6 Редагування особистого рецепту

Початковий стан системи	Користувач автентифікований і знаходиться на сторінці редагування особистого рецепту
Вхідні дані	Назва рецепту, загальна кількість порцій, інгредієнти з назвами, одиницями виміру та кількістю
Опис проведення тесту	У відповідні поля вводяться нові або змінені назва рецепту та кількість порцій. У секції додавання інгредієнтів у пошуку вводиться назва, обирається інгредієнт, задаються одиниця виміру та кількість, після чого інгредієнт додається або оновлюється у списку. Аналогічно редагуються або додаються інші інгредієнти. Наприкінці натискається кнопка збереження змін рецепту.

Продовження таблиці 4.6

Очікуваний результат	Редагований рецепт успішно зберігається, користувач переадресовується на сторінку перегляду особистих рецептів.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.7 – Тест 1.7 Редагування запису споживання їжі

Початковий стан системи	Користувач автентифікований і знаходиться на домашній сторінці з наявним записом споживання їжі, який потрібно редагувати.
Вхідні дані	Запис, який необхідно редагувати, а також нові значення назви їжі, одинці виміру та кількості спожитих одиниць.
Опис проведення тесту	Користувач обирає існуючий запис для редагування та натискає відповідну кнопку, після чого відкривається модальне вікно вибору їжі з початковими значеннями запису споживання. У поле пошуку вводиться нову назву їжі, а серед результатів обирається потрібна їжа, після чого вводиться нова кількість та одиниці виміру. Урешті-решт, натискається кнопка збереження.
Очікуваний результат	Запис споживання оновлюється відповідно до введених даних, модальне вікно закривається, а на домашній сторінці відображається оновлений запис.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.8 – Тест 1.8 Редагування запису споживання рецепту

Початковий стан системи	Користувач автентифікований і знаходиться на домашній сторінці з наявним записом споживання рецепту, який потрібно редагувати.
Вхідні дані	Користувач автентифікований і знаходиться на домашній сторінці з наявним записом споживання рецепту, який потрібно редагувати.
Опис проведення тесту	Користувач обирає існуючий запис для редагування та натискає відповідну кнопку, після чого відкривається модальне вікно вибору рецепту з початковими значеннями запису споживання. У поле пошуку вводиться нова назва рецепту, серед результатів обирається потрібний рецепт, після чого вводиться нова кількість спожитих порцій. Урешті-решт, натискається кнопка збереження.
Очікуваний результат	Запис споживання оновлюється відповідно до введених даних, модальне вікно закривається, а на домашній сторінці відображається оновлений запис споживання рецепту.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.9 – Тест 1.9 Експорт файлових звітів споживання

Початковий стан системи	Користувач автентифікований і знаходиться на сторінці генерації звіту.
Вхідні дані	Початкова дата, кінцева дата, обраний формат виводу звіту.

Продовження таблиці 4.9

Опис проведення тесту	Вводяться початкова та кінцева дати у відповідні поля, обирається формат виводу, після чого натискається кнопка експортування. Текст кнопки змінюється на “Loading”, а система генерує звіт на основі заданого періоду та обраного формату, після чого розпочинається завантаження файлу зі звітом, що містить записи споживання та їхні нутріційні відомості.
Очікуваний результат	Файл зі звітом успішно згенерований і завантажений, містить коректні дані про споживання та нутрієнти за вибраний період.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.10 – Тест 1.10 Редагування профілю користувача

Початковий стан системи	Користувач автентифікований і знаходиться на сторінці редагування профілю.
Вхідні дані	Нове ім'я користувача, оновлений перелік відслідковуваних нутрієнтів із цілями по кожному та інформацією щодо необхідності відслідковування.
Опис проведення тесту	У поле імені вводиться нове значення, після чого відбувається перехід до секції відслідковуваних нутрієнтів, де коригуються цільові значення для обраних нутрієнтів і встановлюються прапорці для включення або виключення їх зі списку відслідковування. Після внесення змін натискається кнопка збереження.

Продовження таблиці 4.10

Очікуваний результат	Зміни зберігаються успішно, відбувається переадресація користувача на сторінку профілю, нові значення і налаштування відслідковування відображаються коректно.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.11 – Тест 1.11 Копіювання публічного рецепту до особистих

Початковий стан системи	Користувач автентифікований і знаходиться на сторінці публічного рецепту, який бажає скопіювати до власної колекції.
Вхідні дані	Обраний публічний рецепт.
Опис проведення тесту	Натискається кнопка копіювання рецепту.
Очікуваний результат	Відбувається автоматичне перенаправлення на сторінку особистих рецептів, де серед списку рецептів з'являється скопійований рецепт
Фактичний результат	Збігається з очікуванням.

Таблиця 4.12 – Тест 1.12 Видалення запису споживання рецепту

Початковий стан системи	Користувач автентифікований і знаходиться на домашній сторінці з наявним записом споживання рецепту, який потрібно видалити.
Вхідні дані	Існуючий запис споживання рецепту, який підлягає видаленню.
Опис проведення тесту	Знаходиться потрібний запис серед списку споживання на домашній сторінці, після чого натискається кнопку видалення.

Продовження таблиці 4.12

Очікуваний результат	Запис споживання рецепту видаляється зі списку, інтерфейс оновлюється, і на домашній сторінці більше не відображається відповідний запис.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.13 – Тест 1.13 Видалення запису споживання їжі

Початковий стан системи	Користувач автентифікований і знаходиться на домашній сторінці з наявним записом споживання їжі, який потрібно видалити.
Вхідні дані	Існуючий запис споживання їжі, який підлягає видаленню.
Опис проведення тесту	Знаходиться потрібний запис серед списку споживання на домашній сторінці, після чого натискається кнопка видалення.
Очікуваний результат	Запис споживання їжі видаляється зі списку, інтерфейс оновлюється, і на домашній сторінці більше не відображається відповідний запис.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.14 – Тест 1.14 Створення рекомендованого раціону

Початковий стан системи	Користувач автентифікований і знаходиться на сторінці створення рекомендованого раціону.
Вхідні дані	Нутріційні цілі користувача, а також кількість комірок рецептів, кожна з яких включає в себе інформацію про бажану категорію страви, інгредієнти для включення та інгредієнти для виключення.

Продовження таблиці 4.14

Опис проведення тесту	Задається кількість комірок страв, вказуються категорії та інгредієнти для включення чи виключення, вводяться нутріційні цілі, після чого натискається кнопка створення рекомендації.
Очікуваний результат	Генерація завершується успішно, у результаті чого користувач отримує список рекомендованих рецептів із зазначенням необхідної кількості порцій кожного, при цьому підібрані страви відповідають заданим категоріям та умовам включення й виключення інгредієнтів, а також у сукупності наближаються до заданих нутріційних цілей.
Фактичний результат	Збігається з очікуванням.

4.3 Опис контрольного прикладу

Розгляньмо повноцінний цикл користування вебзастосунком користувачем. Авторизований користувач знаходиться на сторінці власного профілю, де натискає кнопку редагування (рисунок 4.2).

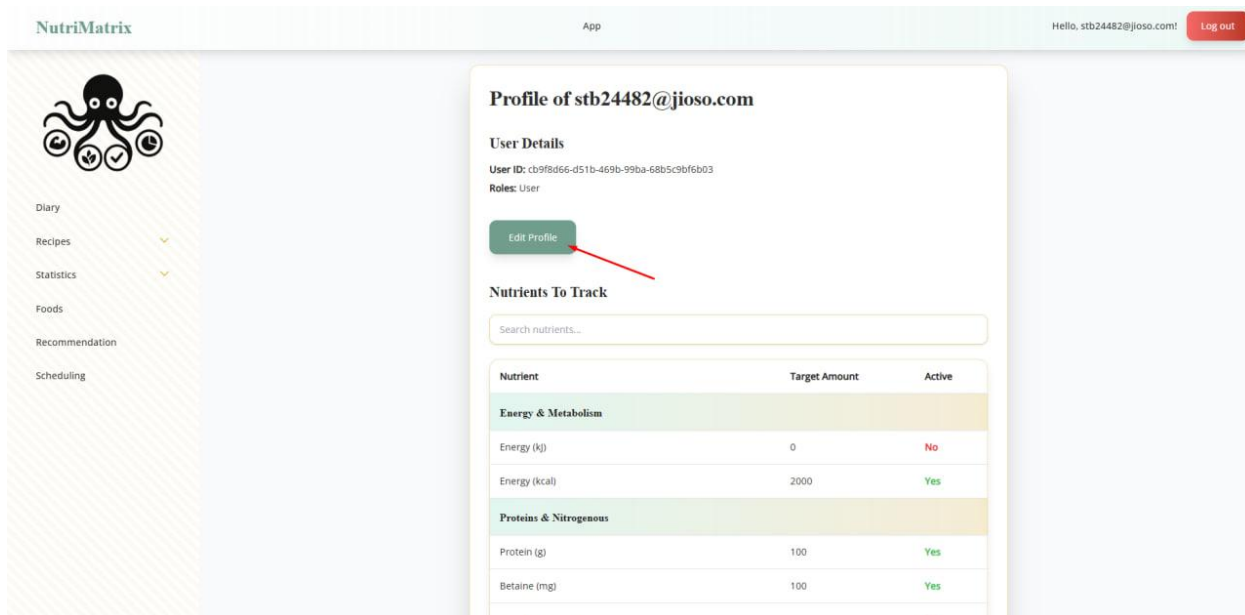


Рисунок 4.2 – Сторінка профілю користувача

Після натискання відбувається переадресація на сторінку редагування профілю, де користувач шукає нутрієнт, що його цікавить та оновлює його запис, задаючи його цільове значення, вказує чи необхідно його відслідковувати, та натискає кнопку збереження(рисунок 4.3).

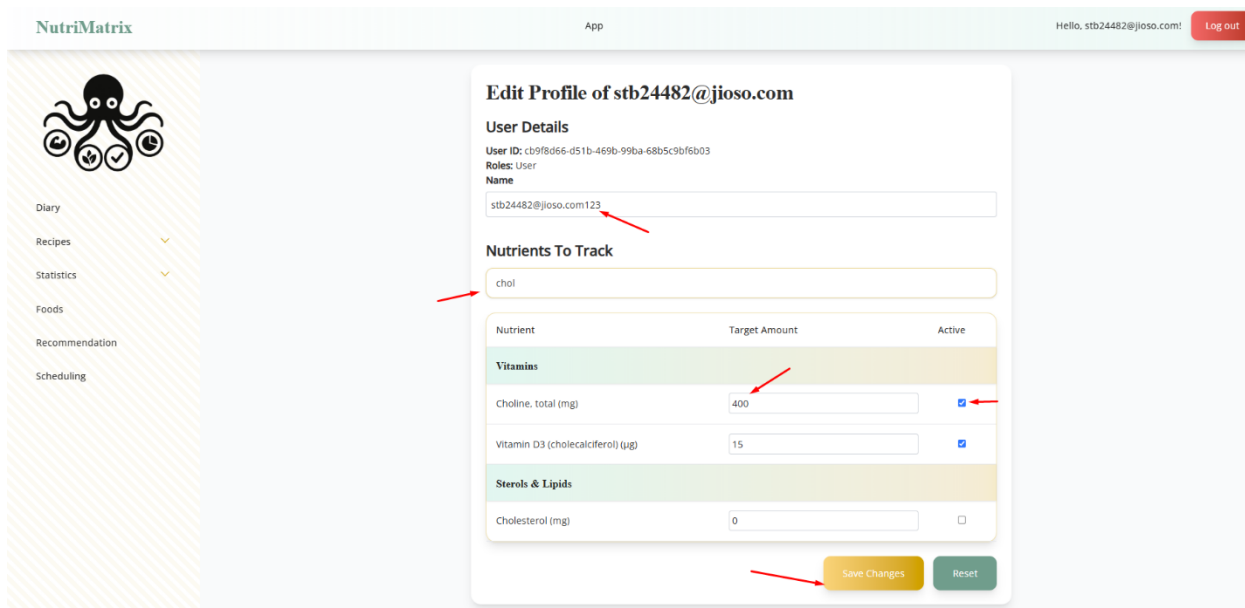


Рисунок 4.3 – Сторінка редагування профілю користувача

Якщо при збереженні змін виникла помилка, користувачеві буде виведено відповідне повідомлення, в іншому ж випадку він переходить назад на домашню сторінку та натискає кнопку додавання запису споживання їжі (рисунок 4.4).

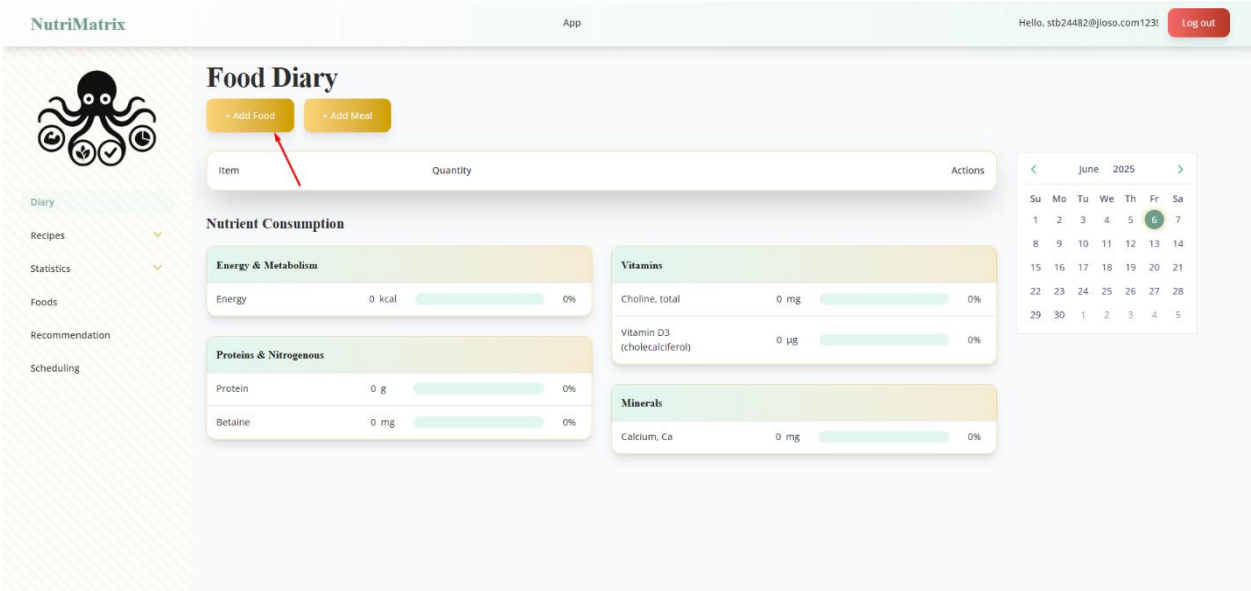


Рисунок 4.4 – Домашня сторінка та кнопка додавання запису споживання їжі

Відкривається модальне вікно, де користувач уводить пошуковий запит, обирає їжу серед результатів, одиницю виміру та кількість, після чого кнопка додавання запису споживання стає активною – він натискає на неї (рисунок 4.5).

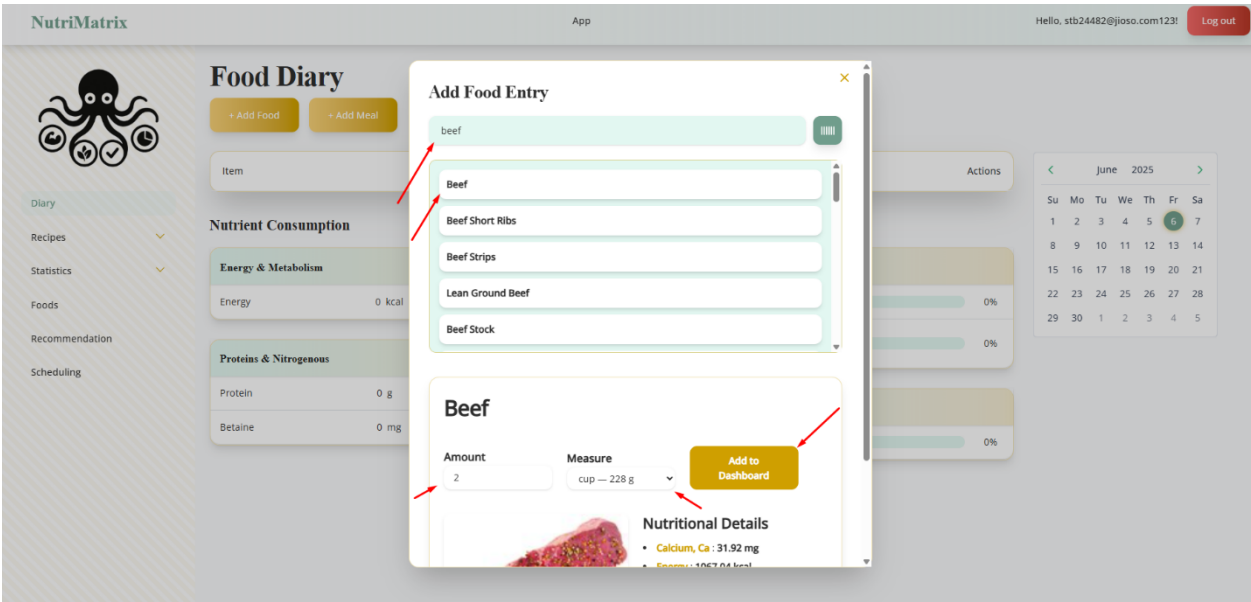


Рисунок 4.5 – Внесення даних запису споживання

У разі успішного додавання, запис зберігається в системі та відображається на домашній сторінці, а статистика споживання нутрієнтів за день оновлюється (рисунок 4.6).

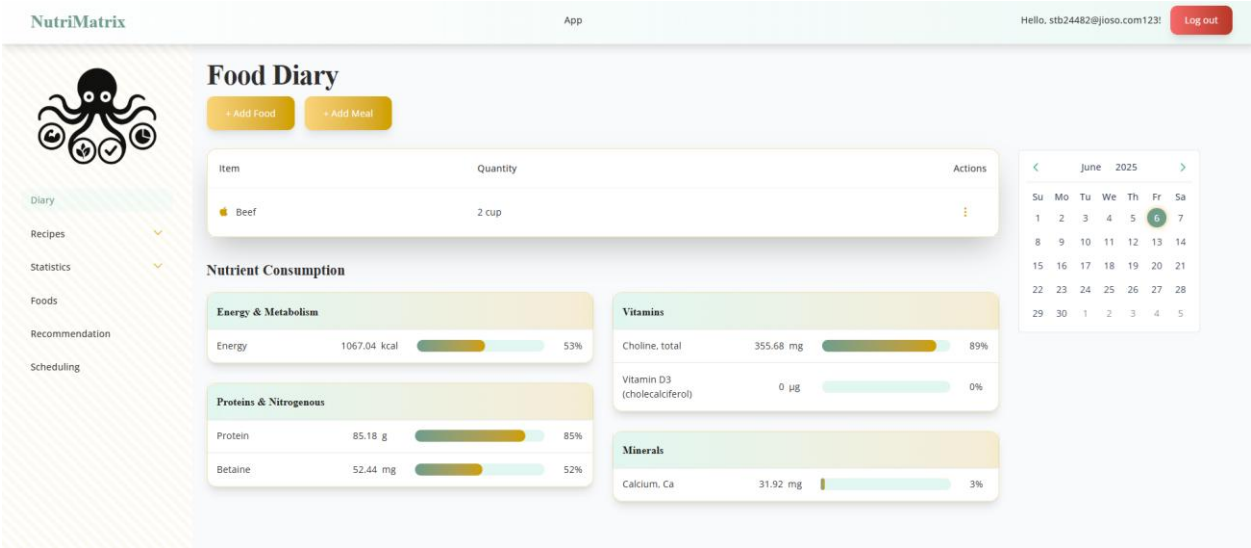


Рисунок 4.6 – Щоденник харчування оновлено

Опісля користувач переходить на сторінку формування рекомендованого раціону, де він додає позиції страв, за необхідності задає для окремих з них інгредієнти, що необхідно включити або виключити, категорію, а також нутріційні цілі (рисунок 4.7).

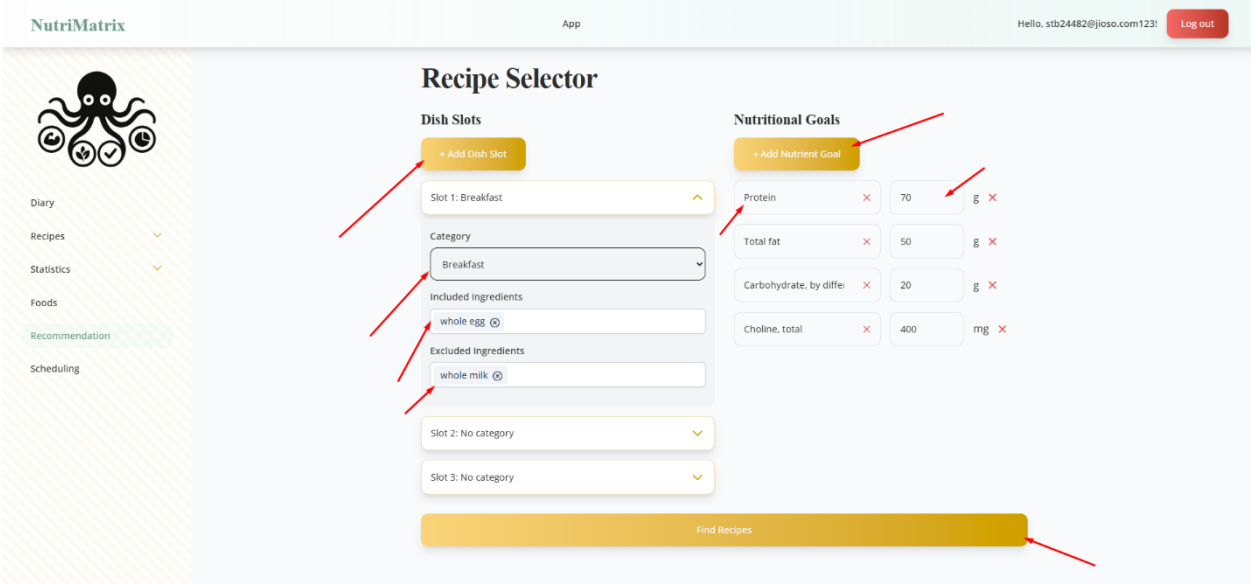


Рисунок 4.7 – Задання характеристик цільового раціону

Після зазначення всіх необхідних параметрів кнопка запиту формування рекомендацій стає активною і користувач натискає на неї, після чого, через короткий період очікування, на екран виводиться згенерований перелік рекомендованих страв; якщо користувач задоволений ними, він може натиснути кнопку збереження (рисунок 4.8).

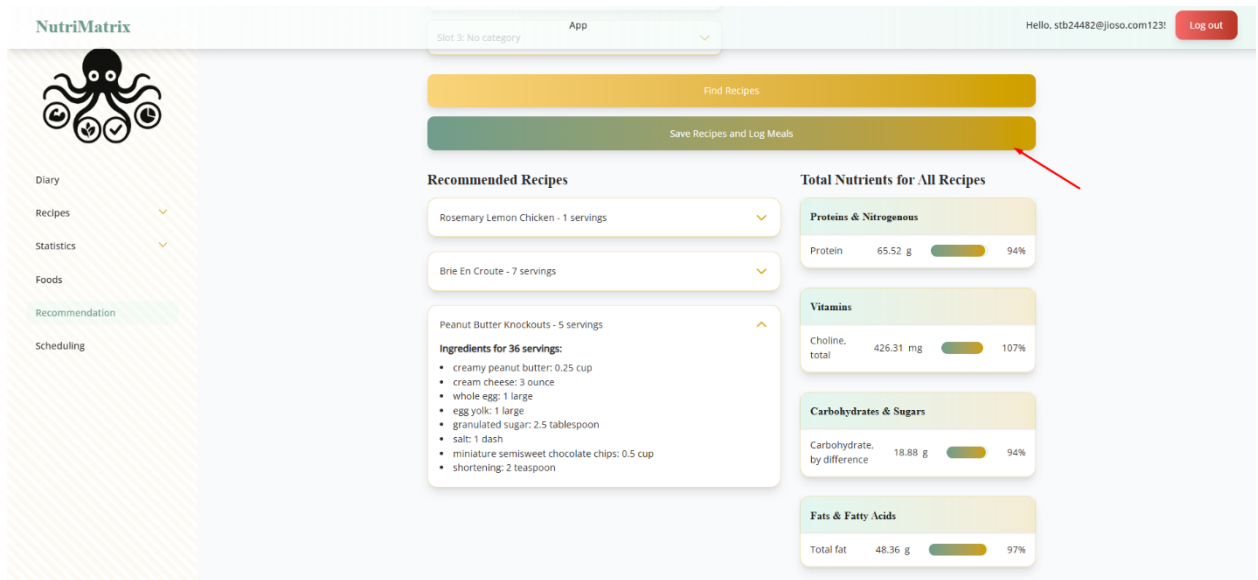


Рисунок 4.8 – Кнопка збереження раціону

Опісля вказані страви додаються в колекцію приватних рецептів, а записи споживання необхідних кількостей порцій додаються в щоденник харчування (рисунок 4.9).

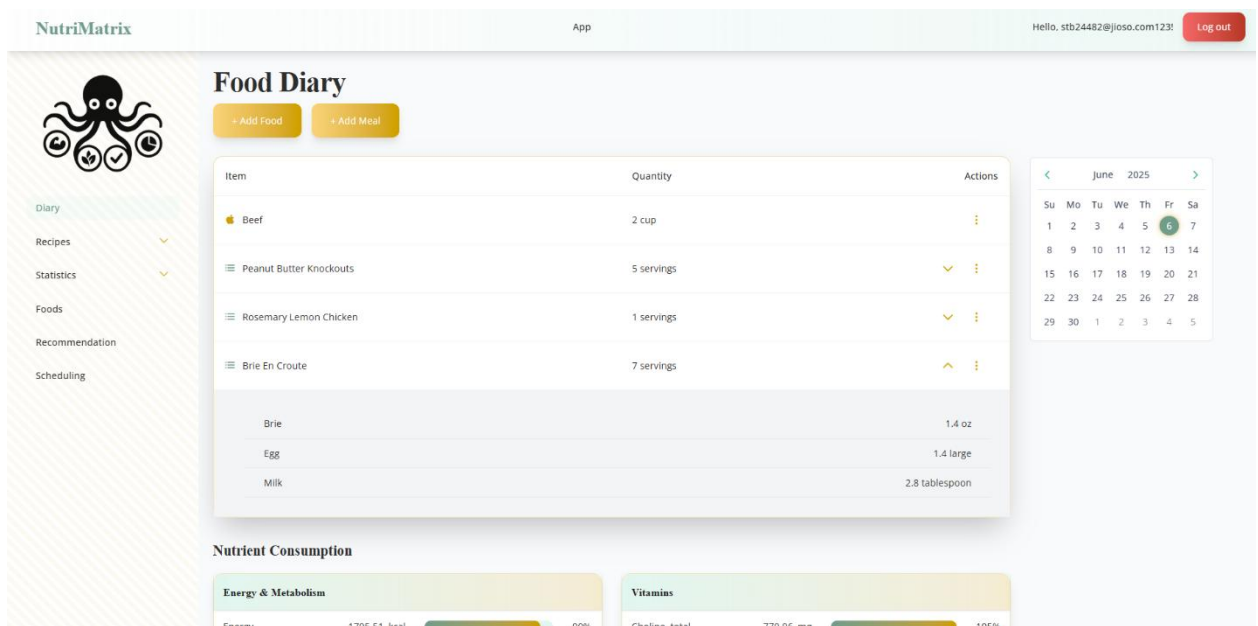


Рисунок 4.9 – Записи додано в щоденник харчування

Урешті-решт, користувач переходить на вкладку формування звітнього файлу (рисунок 4.10), де послідовно зазначає початкову та кінцеву дати вибірки, а також обирає бажаний формат експорту.

Рисунок 4.10 – Форма запиту файлового звіту споживання

Після підтвердження через кнопку формування автоматично ініціюється процес завантаження сформованого звіту у відповідному форматі (рисунок 4.11).

```
#,Date,Item Name,Units Eaten,Unit,Protein (g),"Calcium, Ca (mg)",Betaine (mg),"Choline, total (mg)",Energy (kcal),Vitamin D3 (cholecalciferol) (µg)
1,2025-06-06,Peanut Butter Knockouts → creamy peanut butter,0.034722224,cup,1.96,4.84,0.04,5.88,52.67,0
2,2025-06-06,Peanut Butter Knockouts → cream cheese,0.41666666,ounce,0.73,11.46,0.08,3.21,41.34,0
3,2025-06-06,Peanut Butter Knockouts → whole egg,0.1388889,large,0.87,3.89,0.02,20.4,9.93,0.14
4,2025-06-06,Peanut Butter Knockouts → egg yolk,0.1388889,large,0.37,3.05,0.02,19.37,7.6,0.13
5,2025-06-06,Peanut Butter Knockouts → granulated sugar,0.3472222,tablespoon,0.04,0.16,86,0
6,2025-06-06,Peanut Butter Knockouts → salt,0.1388889,dash,0.01,0,0,0
7,2025-06-06,Peanut Butter Knockouts → miniature semisweet chocolate chips,0.06944445,cup,0.12,0.93,0.064,14,0
8,2025-06-06,Peanut Butter Knockouts → shortening,0.2777778,teaspoon,0,0,0.3,10.68,0
9,2025-06-06,Rosemary Lemon Chicken → rosemary,0.5,teaspoon,0.03,7.23,0,0.1,87,0
10,2025-06-06,Rosemary Lemon Chicken → boneless skinless chicken breast,1,breast,42.53,9.45,13.23,155.17,226.8,0
11,2025-06-06,Rosemary Lemon Chicken → pepper,0.0625,teaspoon,0.01,0.57,0.01,0.01,0.32,0
12,2025-06-06,Rosemary Lemon Chicken → vegetable oil,0.75,tablespoon,0,0,0.02,90.15,0
13,2025-06-06,Rosemary Lemon Chicken → oregano,0.25,teaspoon,0.03,5.66,0.03,0.11,0.94,0
14,2025-06-06,Rosemary Lemon Chicken → lemon juice,0.5,tablespoon,0.03,0.76,0.039,1.29,0
15,2025-06-06,Brie En Croute → brie,1.4,oz,8.24,73.03,0.6,11,132.56,0.2
16,2025-06-06,Brie En Croute → egg,1.4,large,8.79,39.2,0.21,205.66,100.1,1.4
17,2025-06-06,Brie En Croute → milk,2.8,tablespoon,1.41,51.24,0.38,7,21.35,0.51
18,2025-06-06,beef,2,cup,85.18,31.92,52.44,355.68,1067.04,0
```

Рисунок 4.11 – Результуючий звіт

Висновки до розділу

У рамках даного розділу було наведено результати аналізу якості програмного забезпечення шляхом застосування інструменту статичного аналізу embold.io, у результаті чого було встановлено, що переважна більшість цільових метрик: оцінка функціональності, стійкість, ефективність, надійність, концептуальна цілісність та безпечність, – були високо оцінені даним аналізатором, тоді як підтримуваність та зручність використання отримали відносно нижчі оцінки в 79 % та 86 % відповідно.

Проведено низку мануальних тестувань, у ході яких було перевірено коректність роботи основних функцій системи. Зокрема, було протестовано процеси реєстрації та автентифікації користувача, додавання та редагування записів споживання їжі і рецептів, створення та редагування особистих рецептів, експортування файлових звітів, редагування профілю користувача, видалення записів споживання, тощо. Усі випробування пройшли успішно, а фактичні результати збіглися з очікуваними.

Урешті-решт, було описано контрольний приклад випробування застосунку, що відображав типовий цикл користування застосунком: редагування персональних даних із оновленням налаштувань відслідковування, внесення записів про споживання продуктів і їх кількості, формування рекомендацій стосовно раціону із можливістю встановлення цільових параметрів, збереження рекомендованих варіантів із подальшим автоматизованим додаванням записів їхнього споживання, а також формування і експорт звітів за обраний період у різних форматах.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Серверну частину було вирішено розгорнути на платформі Railway [33]. Передумовою розгортання є наявність облікового запису на github.com, а також репозиторію, розміщеного на ньому. Для початку розгортання необхідно авторизуватися на даній платформі через Github, після чого відбувається переадресація на домашню сторінку з проектами, де необхідно натиснути “New” (рисунок 5.1).

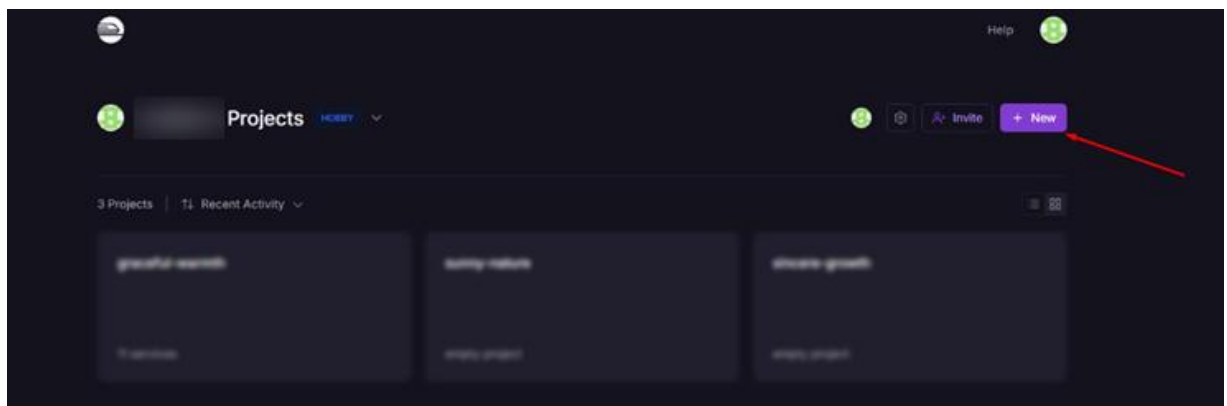


Рисунок 5.1 – Результуючий звіт

Тоді необхідно обрати “Deploy from Github”(рисунок 5.2), а опісля вказати репозиторій з вебзастосунком(рисунок 5.3).

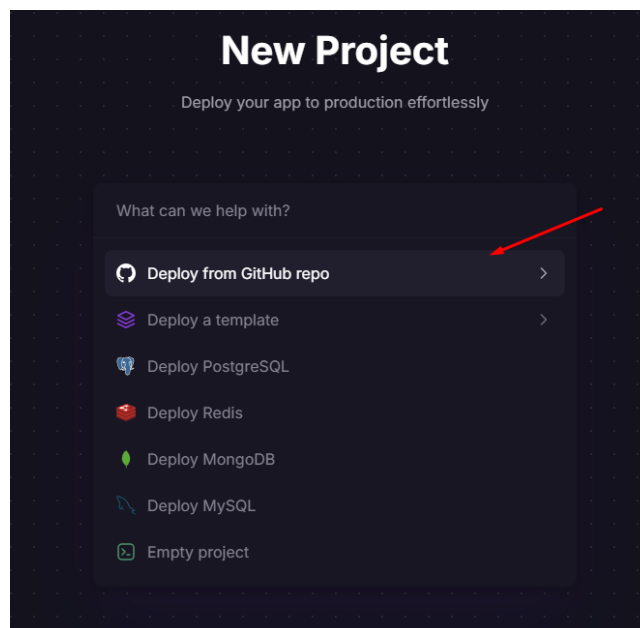


Рисунок 5.2 – Вибір опції “Deploy from Github repo”

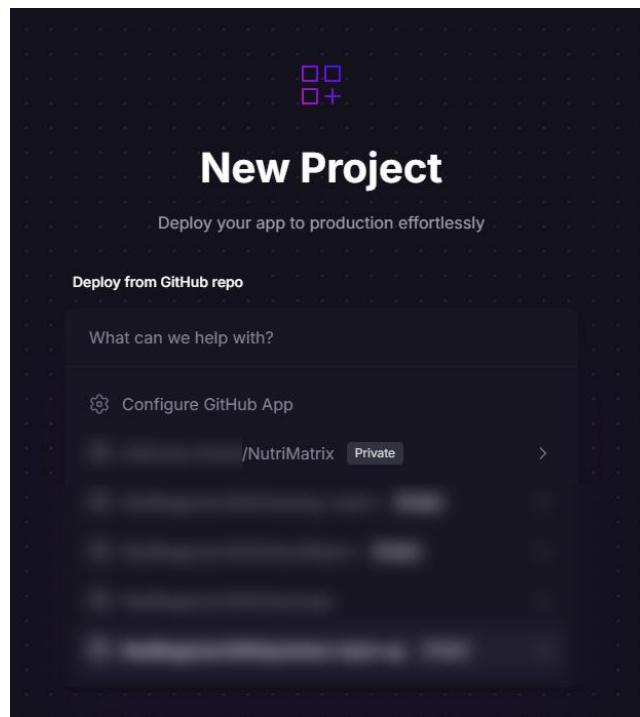


Рисунок 5.3 – Вибір репозиторію

Після цього створюється новий проєкт і відбувається перехід на його полотно. На нього необхідно перетягнути файл `docker-compose.override.yml`, у результаті чого на полотні ініціалізуються контейнери застосунку(рисунок 5.4).

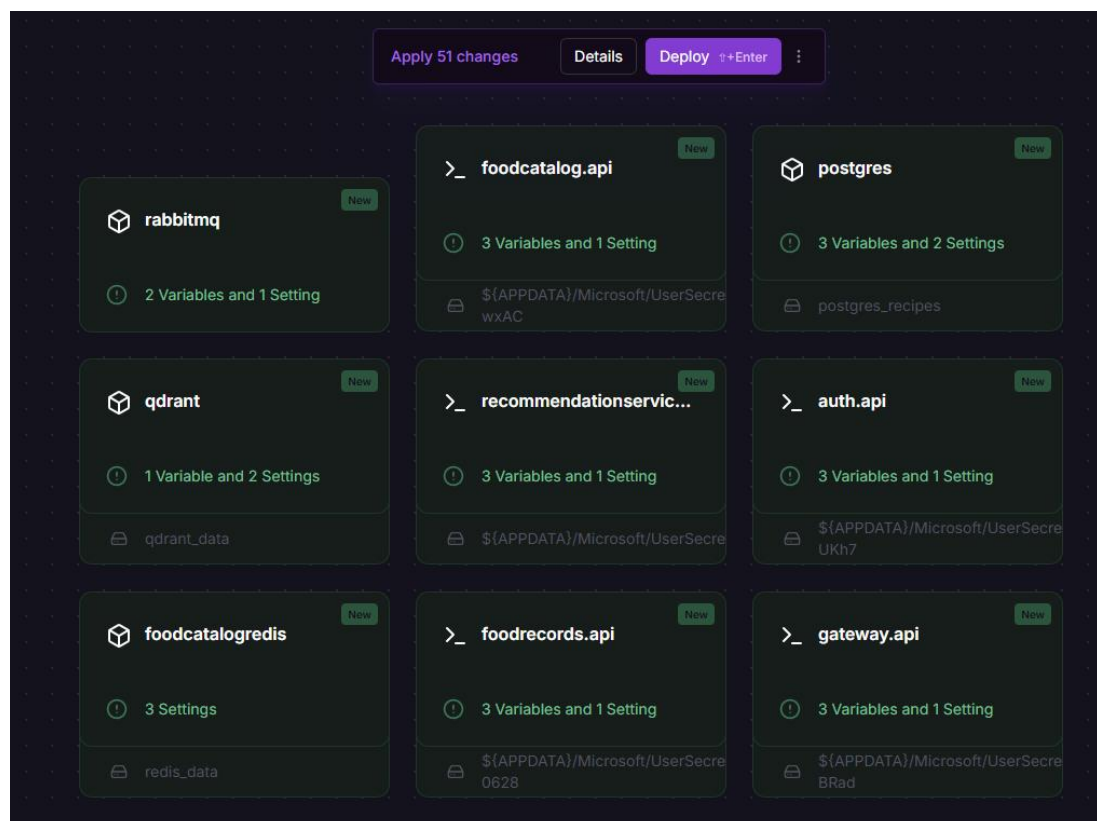


Рисунок 5.4 – Полотно проєкту з контейнерами

Наступним кроком є налаштування розгортання самих контейнерів з мікросервісами. Для цього необхідно натиснути на кожного з них, прив'язати їх до даного репозиторію з вебзастосунком та вказати кореневу директорію, у якій проводитиметься пошук докерфайлу розгортуючої одиниці, як показано на рисунку 5.5.

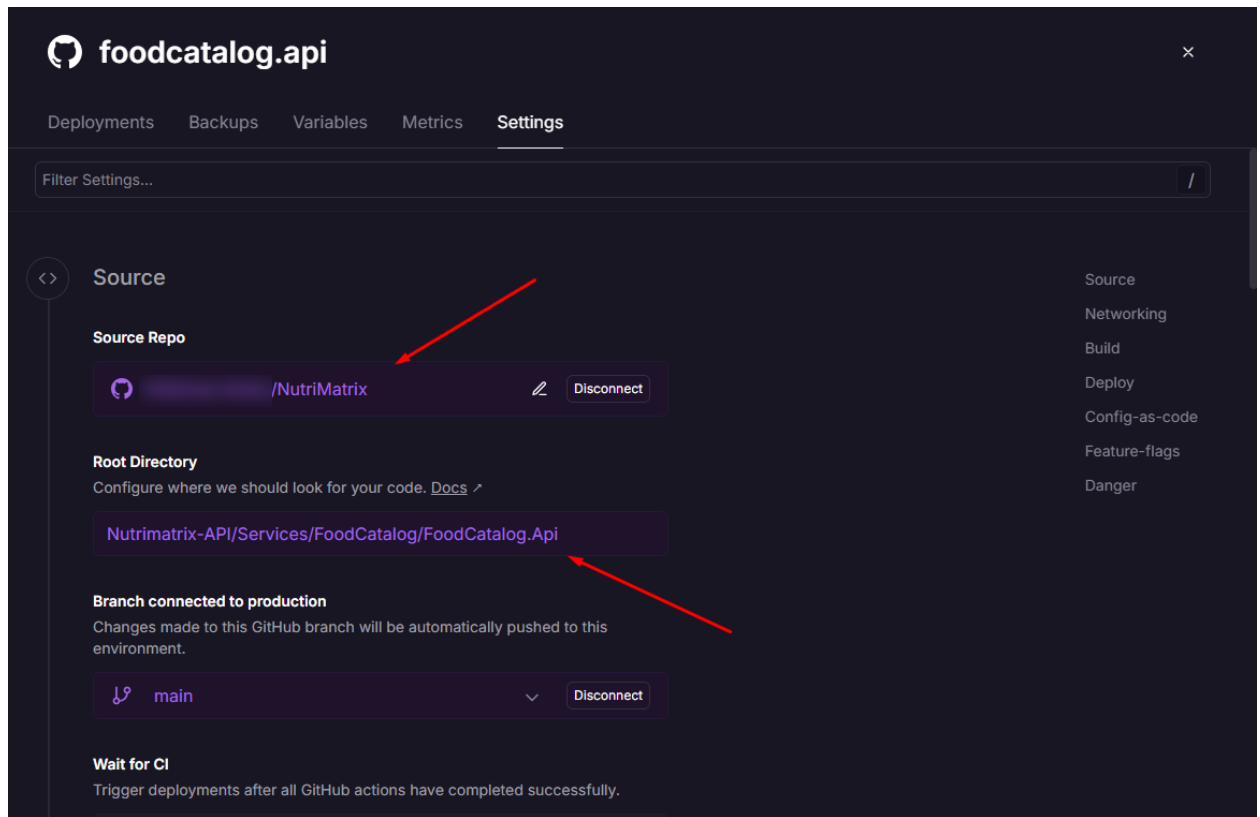


Рисунок 5.5 – Налаштування розгортання мікросервісів

Крім цього передбачається використання векторної БД Qdrant, комунікація з якою здійснюватиметься через GRPC-клієнт. Відповідно, необхідно також сконфігурувати TCP-проксі для даного контейнера. Для цього потрібно натиснути на нього та обрати “TCP proxy” (рисунок 5.6). Опісля платформа автоматично запропонує домейн та порти (рисунок 5.7).

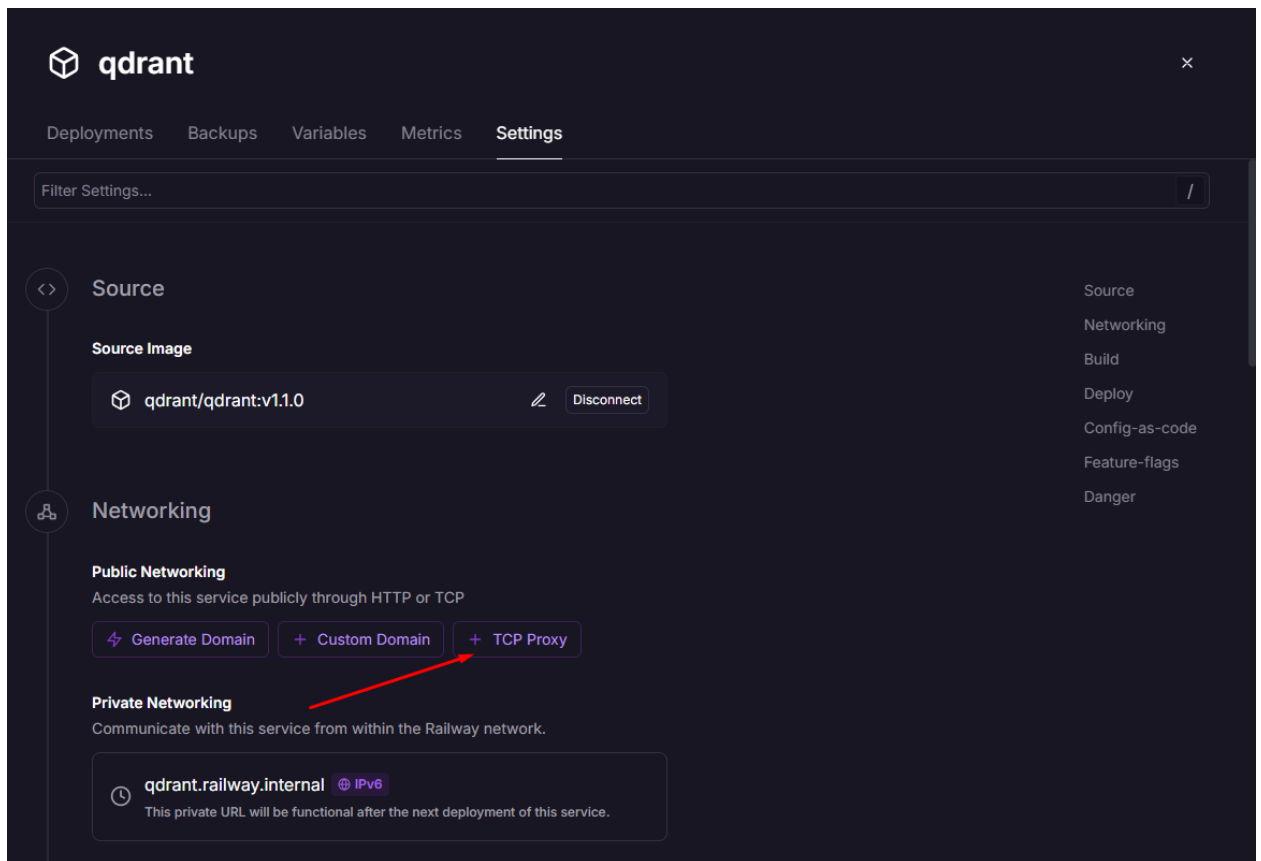


Рисунок 5.6 – Кнопка додавання ТСП-проксі

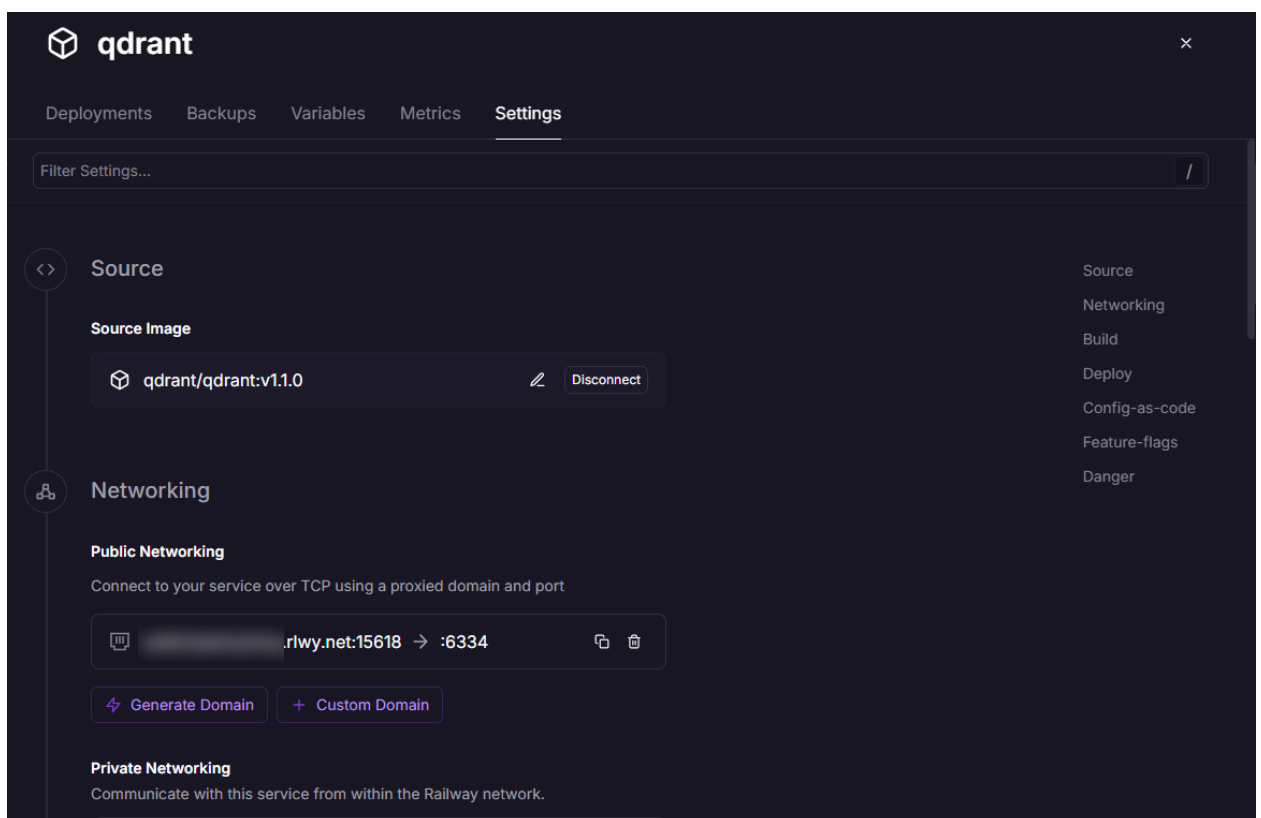


Рисунок 5.7 – ТСП-проксі створено

Далі необхідно отримати публічний домен для API-шлюзу. Для цього натискаємо на його сервіс та обираємо “Generate domain” (рисунок 5.8).

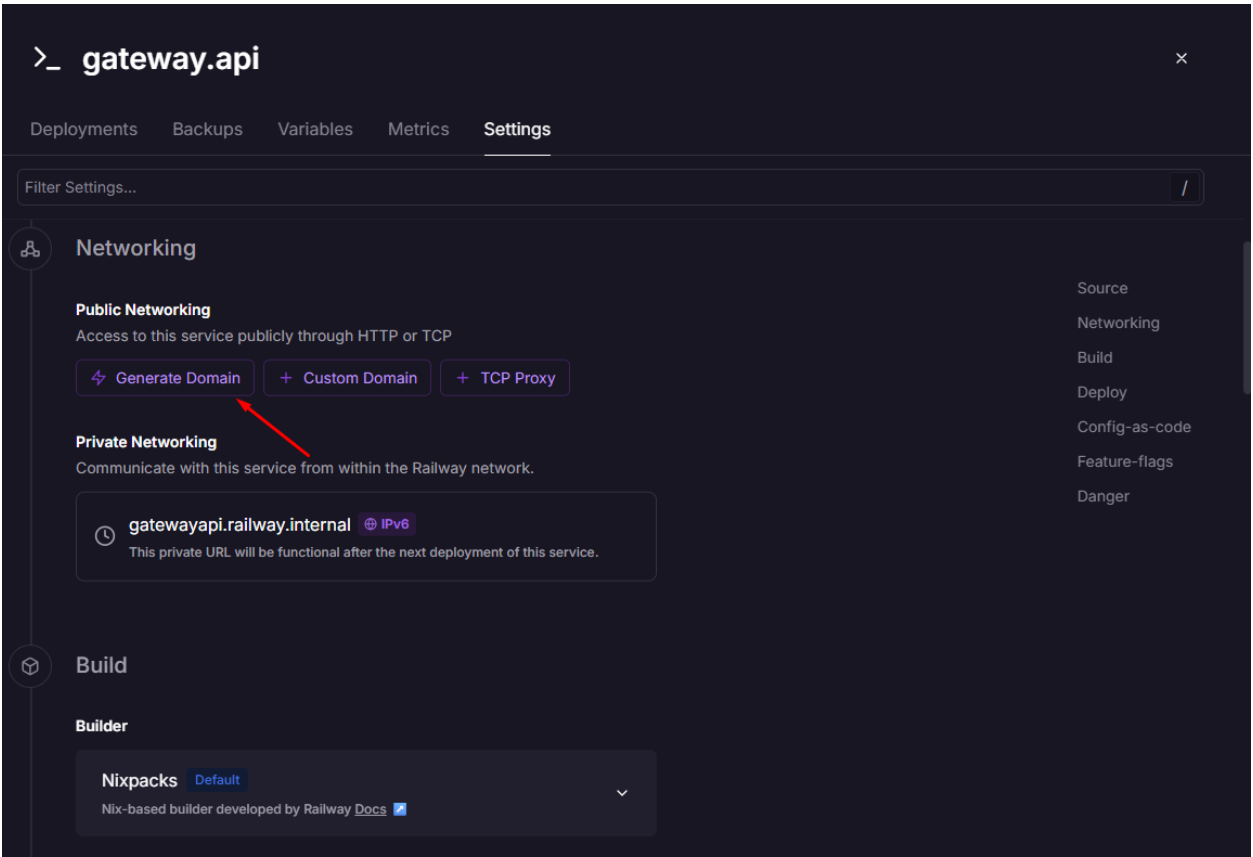


Рисунок 5.8— Кнопка додавання домейну

Для завершення розгортання необхідно натиснути кнопку “Deploy” – результат розгортання наведено на рисунку 5.9.

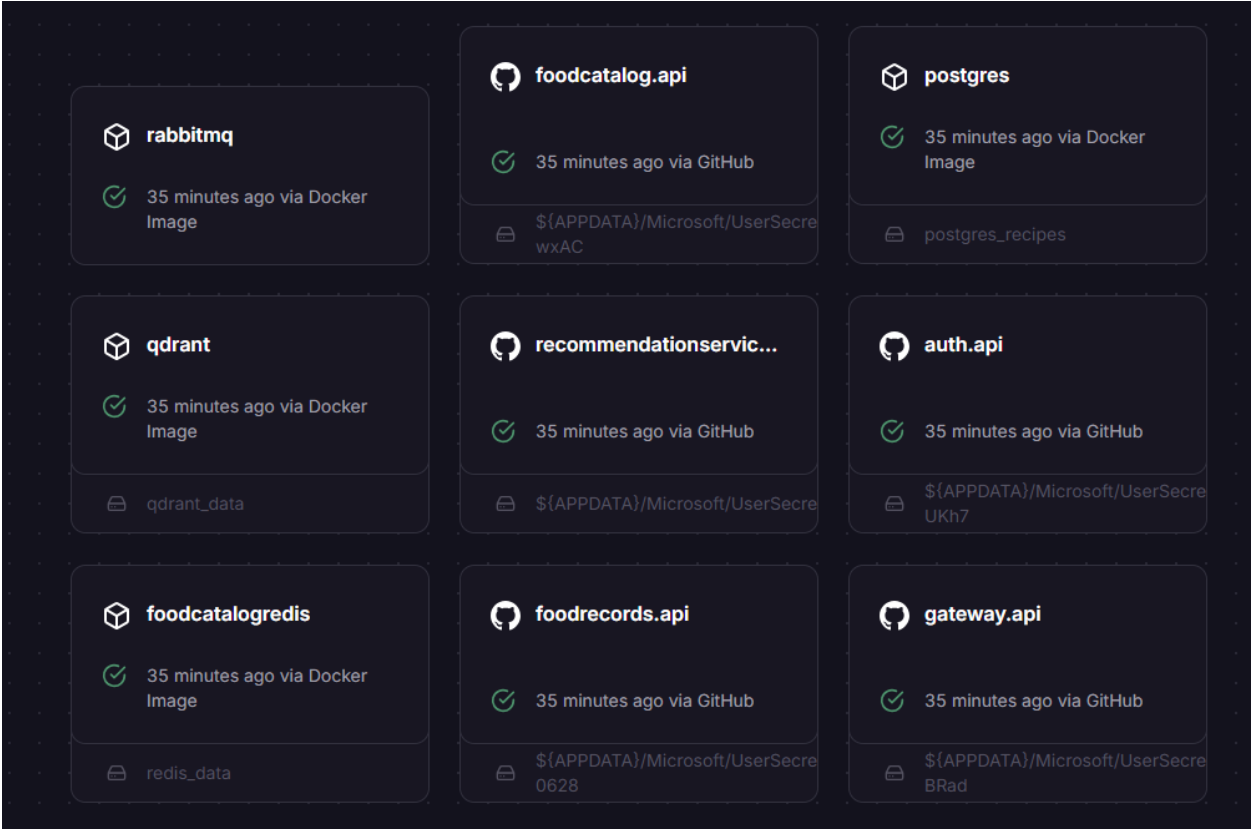


Рисунок 5.9 – Результат розгортання серверної частини

Клієнську частину було розгорнуто на платформі GoogleFirebase [34]. Для цього було перейдено на адресу `console.firebase.google.com`, після чого в профілі натиснуто кнопку “Create project” (рисунок 5.10).

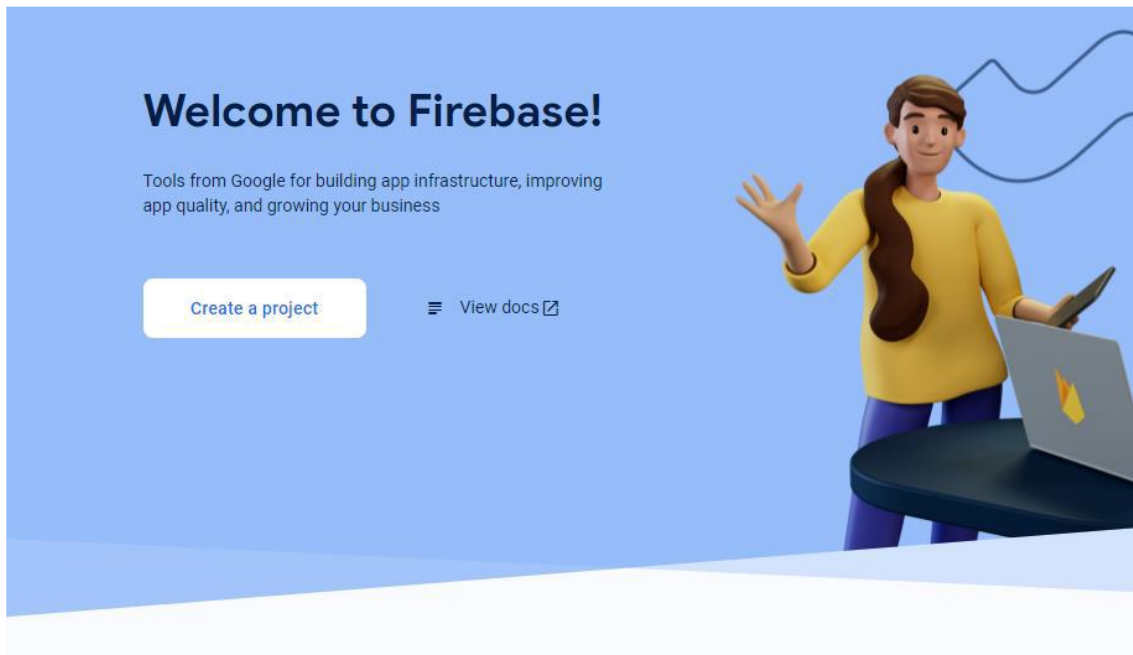


Рисунок 5.10 – Кнопка створення проєкту

Тоді необхідно дати ім'я проєкту та натиснути далі на всіх подальших формах, після чого відбувається переадресація на сторінку проєкту. Там необхідно натиснути на кнопку “Web” – рисунок 5.11.

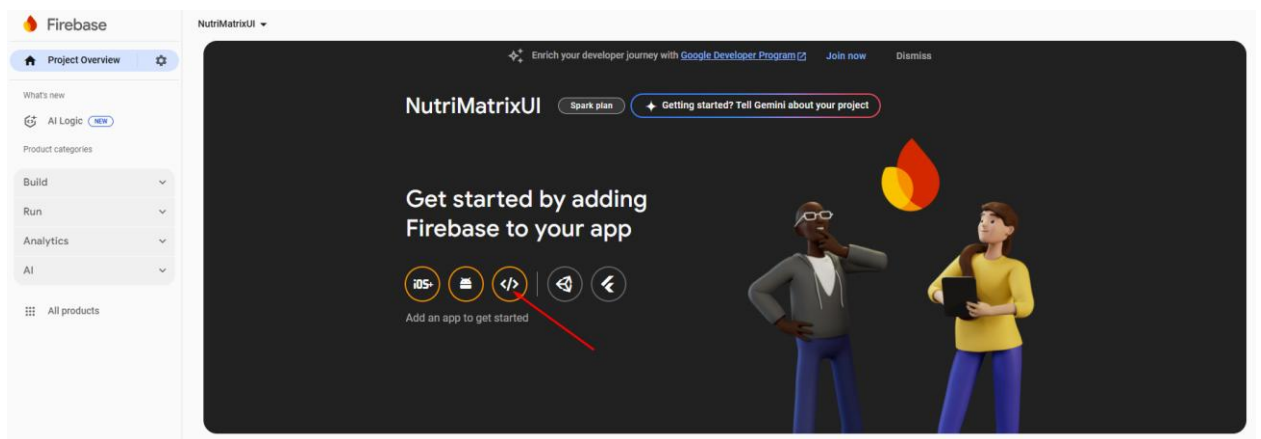


Рисунок 5.11 – Сторінка проєкту

Додається ім'я проєкту та завершується його реєстрація. Тоді необхідно перейти в локальну директорію проєкту, відкрити консоль та виконати команди:

```
npm install -g firebase-tools
```

```
firebase login
```

Опісля з'являється вікно входу в обліковий запис Google – необхідно провести автентифікацію. Після неї потрібно виконати команду `firebase init`, у результаті чого з'являється вибір кількох опцій. Необхідно обрати ту, що виділена на рисунку 5.12.

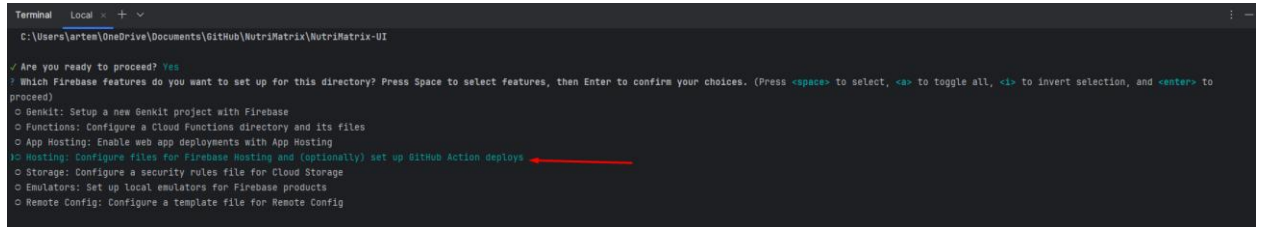


Рисунок 5.12– Необхідна опція

Тоді обирається створений Firebase-проект та встановлюється `dist` як публічна директорія. Опісля у файл `angular.json` необхідно змінити відповідні значення як на рисунку 5.13

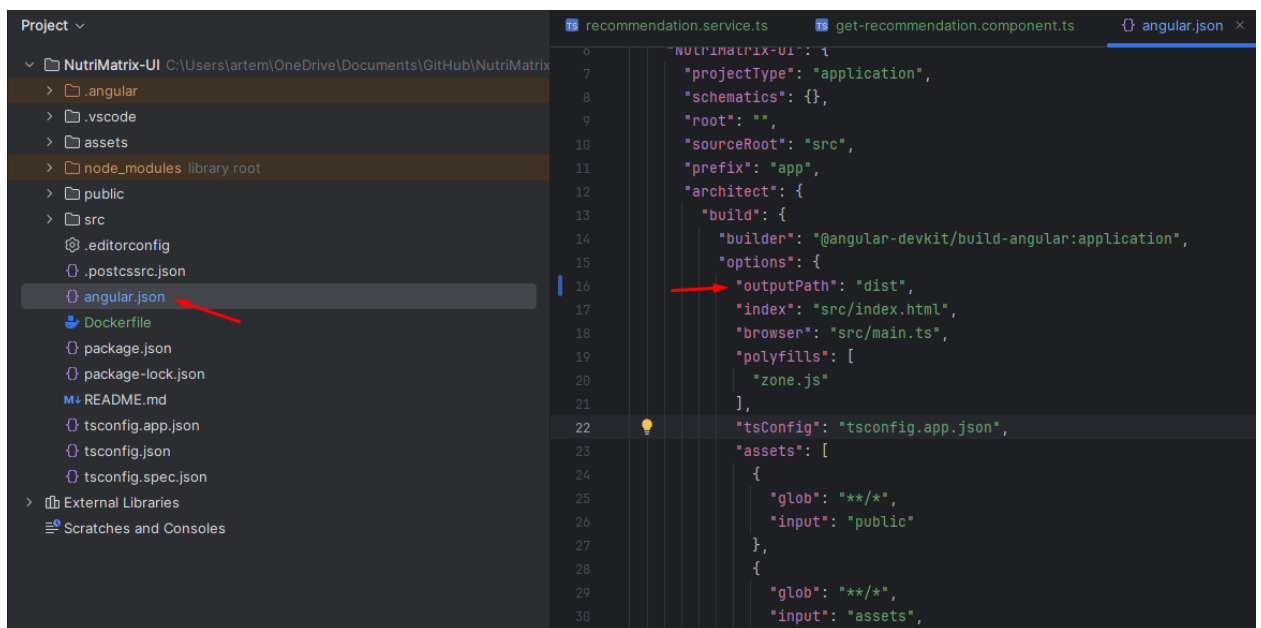


Рисунок 5.13 – Оновлення файлу Angular.json

У файлу `environment.ts` потрібно оновити базове посилання на публічний домен API-шлюзу сервера. Урешті-решт, потрібно виконати в консолі команди:

```
ng build --configuration=production
```

```
firebase deploy
```

Клієнтську частину розгорнуто.

5.2 Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі «Керівництво користувача».

Оновлення серверної частини автоматично здійснюється завдяки механізму безперервної інтеграції, побудованому на зв'язці з GitHub-репозиторієм. Достатньо лише зафіксувати зміни до відповідної гілки віддаленого репозиторію, а платформа Railway власноруч ініціює процес розгортання.

Для оновлення клієнтської частини необхідно виконати команди:

```
ng build --configuration=production  
firebase deploy
```

Висновки до розділу

У рамках даного розділу було сформульовано покрокову інструкцію із розгортання серверної та клієнтської частин додатку.

Серверна частина була розгорнута на хмарній платформі Railway, що передбачало авторизацію через GitHub, ініціалізацію проєкту, прив'язку контейнерів до відповідних мікросервісів, налаштування директорій пошуку Docker-файлів, конфігурацію TCP-проксі для взаємодії з Qdrant, а також генерацію доменного імені для API-шлюзу.

Клієнтську частину було розгорнуто на платформі GoogleFirebase. Основні кроки охорлювали створення проєкту додатку на даній платформі, виконання послідовності команд для налаштування локального середовища, зміна окремих файлів проєкту, побудова його та введення команд для фіналізації розгортання. Сформульовано кроки необхідні для підтримки ПЗ.

ВИСНОВКИ

Головною метою дипломного проєкту була розробка вебзастосунку для відслідковування макро- та мікронутрієнтів у раціоні, що надасть підвищений рівень персоналізації відстеження харчових даних та спрощення користувацьких операцій, пов'язаних із введенням, редагуванням і аналізом інформації харчового щоденника шляхом цифровізації та часткової автоматизації відповідних процесів.

Поставленої мети було досягнуто шляхом впровадження основного функціоналу, що притаманний передовим сервісам даного спрямування з розширенням можливостей окремих можливостей: налаштування аж до 161 відстежуваного нутрієнта, перегляд каталогу харчових продуктів із розширеним пошуком за назвою та штрих-кодом, створення і керування рецептами, ведення і планування записів споживання, автоматичне формування персоналізованого раціону на основі визначених макро- та мікронутрієнтних цілей, а також перегляд звітності й статистики за вибрані періоди.

У результаті виконання даного дипломного проєкту було спроектовано та реалізовано вебзастосунок для відслідковування макро- та мікронутрієнтів у раціоні із застосуванням клієнт-серверної та мікросервісної архітектури. Результуюче програмне забезпечення відповідає вимогам, сформульованим у постановці завдання, та задовольняє потреби, визначені аналізом предметної області.

В якості середовища розробки серверної частини було використано Visual Studio 2022, а також основні технологічні середовища Asp.Net Core, Entity Framework Core, Asp.Net Core Identity та інші. Основним технологічним середовищем клієнтської частини слугував Angular.

Отож, розроблений вебзастосунок становить зручний та повнофункціонально інструмент для персоналізованого відслідковування харчування індивіда. У час, коли зростає усвідомленість користувачів щодо власного раціону, потреба у таких цифрових рішеннях стає все більш

актуальною. Реалізоване ПЗ дозволяє спростити процес ведення харчового щоденника, підвищити ефективність досягнення нутріційних та персоналізувати процес відслідковування, забезпечуючи відповідність потребам найбільшого кола користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Stipanuk M.H., Caudill M.A. Biochemical, Physiological, and Molecular Aspects of Human Nutrition / M. H. Stipanuk, M. A. Caudill. – 3rd ed. – Saunders, 2013. – 990 с. – ISBN 978-1437709599
- 2) Franco R.Z. et al. Popular Nutrition-Related Mobile Apps: A Feature Assessment / R. Z. Franco, R. Fallaize, J. A. Lovegrove, F. Hwang // JMIR Mhealth Uhealth. – 2016. – Vol. 4, No. 3. – DOI:10.2196/mhealth.5846. – Режим доступу: <https://mhealth.jmir.org/2016/3/e85/> — Дата звернення: 15.05.2025.
- 3) Vasiloglou M.F. et al. Perspectives and Preferences of Adult Smartphone Users Regarding Nutrition and Diet Apps: Web-Based Survey Study / M. F. Vasiloglou, S. Christodoulidis, E. Reber, T. Stathopoulou, Ya Lu, Z. Stanga, S. Mougiakakou // JMIR Mhealth Uhealth. – 2021. – Vol. 9, No. 7. – DOI:10.2196/27885. – Режим доступу: <https://mhealth.jmir.org/2021/7/e27885/PDF> — Дата звернення: 15.05.2025.
- 4) MyNetDiary [Електронний ресурс]. – Режим доступу: <https://www.mynetdiary.com/> — Дата звернення: 15.05.2025.
- 5) Eat This Much [Електронний ресурс]. – Режим доступу: <https://www.eatthismuch.com/> — Дата звернення: 15.05.2025.
- 6) Cronometer [Електронний ресурс]. – Режим доступу: <https://cronometer.com/> — Дата звернення: 15.05.2025.
- 7) Distributed Application Architecture [Електронний ресурс] / Sun Microsystems. – Режим доступу: https://web.archive.org/web/20110406000000/http://java.sun.com/blueprints/guidelines/designing_enterprise_applications/ — Назва з екрана. – Дата останнього звернення: 15.05.2025.
- 8) Mishra M., Kunde S., Nambiar M. Cracking the monolith: Challenges in data transitioning to cloud native architectures / M. Mishra, S. Kunde, M. Nambiar // Proceedings of the 12th European Conference on Software Architecture:

- Companion Proceedings. – New York, NY, USA : Association for Computing Machinery (ACM), 2018. – С. 1–4. – DOI: 10.1145/3241403.3241440. – ISBN 9781450364836.
- 9) Richards M. Fundamentals of Software Architecture: An Engineering Approach / M. Richards. – 1st ed. – O'Reilly Media, 2020. – 448 с. – ISBN 978-1492043454.
 - 10) Wolff E. Microservices: Flexible Software Architectures / E. Wolff. – Addison-Wesley, 2016. – 256 с. – ISBN 978-0134602417.
 - 11) Khalil M., Ghani K., Khalil W. Onion architecture: a new approach for XaaS (every-thing-as-a service) based virtual collaborations / M. Khalil, K. Ghani, Wajeeha Khalil // 13th Learning and Technology Conference (L&T). — 2016. — 10 квітня.
 - 12) Lucasius C. B., Kateman G. Understanding and using genetic algorithms Part 1. Concepts, properties and context / C. B. Lucasius, G. Kateman // Chemometrics and Intelligent Laboratory Systems. – 1993. – Т. 19, № 1. – С. 1–33. – DOI: 10.1016/0169-7439(93)80079-W. – ISSN 0169-7439.
 - 13) Dorigo M. Ant colony optimization / M. Dorigo. – Cambridge, Mass.: MIT Press, 2004. – ISBN 9780262256032. – OCLC 57182707
 - 14) Damodaran M., Washington A.E. Estimation Using Use Case Points / M. Damodaran, A. E. Washington // Computer Science Program, University of Houston-Victoria, Texas, USA. – Режим доступу: <https://www.cs.uh.edu/~mel/ucp.pdf> — Дата звернення: 15.05.2025.
 - 15) The C4 model for visualising software architecture [Електронний ресурс] – Режим доступу: <https://c4model.com/> — Дата звернення: 15.05.2025.
 - 16) Nutritionix [Електронний ресурс] — Режим доступу: <https://www.nutritionix.com/> — Дата звернення: 15.05.2025.
 - 17) Brevo [Електронний ресурс] — Режим доступу: <https://www.brevo.com/> — Дата звернення: 15.05.2025.

- 18) Google Cloud Identity Platform [Электронный ресурс] — Режим доступа: <https://cloud.google.com/security/products/identity-platform> — Дата звернения: 15.05.2025.
- 19) Introduction to ASP.NET Core [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnetcore?view=aspnetcore-8.0> — Дата звернения: 15.05.2025.
- 20) Authentication in ASP.NET Core Identity [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio> — Дата звернения: 15.05.2025.
- 21) Entity Framework Core Documentation [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/en-us/ef/core/> — Дата звернения: 15.05.2025.
- 22) MediatR [Электронный ресурс] — Режим доступа: <https://github.com/jbogard/MediatR> — Дата звернения: 15.05.2025.
- 23) RedisOM for .NET [Электронный ресурс] — Режим доступа: <https://redis.io/docs/latest/integrate/redisom-for-net/> — Дата звернения: 15.05.2025.
- 24) Ardalis Specification [Электронный ресурс] — Режим доступа: <https://specification.ardalis.com/> — Дата звернения: 15.05.2025.
- 25) MassTransit [Электронный ресурс] — Режим доступа: <https://masstransit.io/> — Дата звернения: 15.05.2025.
- 26) Quartz Scheduler [Электронный ресурс] — Режим доступа: <https://www.quartz-scheduler.net/> — Дата звернения: 15.05.2025.
- 27) Visual Studio IDE Overview [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022> — Дата звернения: 15.05.2025.
- 28) Understanding Angular Overview [Электронный ресурс] – Режим доступа: <https://angular.io/guide/understanding-angular-overview> — Дата звернения: 15.05.2025.

- 29) Getting started with WebStorm [Электронный ресурс] – Режим доступа: <https://www.jetbrains.com/help/webstorm/getting-started-with-webstorm.html> — Дата звернення: 15.05.2025.
- 30) Raeburn K. Advanced Encryption Standard (AES) Encryption for Kerberos 5 [Электронный ресурс] / K. Raeburn // RFC 3962. – 2005. – DOI: 10.17487/RFC3962. – Режим доступа: <https://tools.ietf.org/html/rfc3962> — Дата звернення: 15.05.2025.
- 31) Descriptions of SHA-256, SHA-384, and SHA-512 [Электронный ресурс] / National Institute of Standards and Technology (NIST). – Режим доступа: <https://web.archive.org/web/20130526224224/https://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf> — Назва з екрана. – Дата звернення: 15.05.2025.
- 32) Embold [Электронный ресурс] — Режим доступа: <https://embold.io/> — Дата звернення: 02.06.2025
- 33) Railway [Электронный ресурс] — Режим доступа: <https://railway.app/> — Дата звернення: 02.06.2025.
- 34) Firebase Console [Электронный ресурс] — Режим доступа: <https://console.firebase.google.com/> — Дата звернення: 02.06.2025.

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

StrikePlagiarism.com

Дата звіту
Дата редагування

6/8/2025
6/8/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute

Заголовок
ІП-15_Хільчук_ПЗ

Автор
Науковий керівник / Експерт
ІП-15_ХільчукМарченко О.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.

12.76%
12.76% КП 1

10
Довжина фрази для коефіцієнта подібності 2

11868
Кількість слів

93341
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навісний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв

0

Інтервали

0

Мікропробіли

4

Білі знаки

18

Парафрази (SmartMarks)

106

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Копір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз

Копір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	ІП-15_Рибалка_ПЗ 6/2/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	134 1.13 %
2	ІП-11_Боровков_ПЗ 6/2/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	48 0.40 %

ДОДАТОК Б ОПИС ВАРІАНТІВ ВИКОРИСТАННЯ

Таблиця Б.1 – Варіант використання UC-01

Use case name	Реєстрація користувача
Use case ID	UC-01
Goals	Створення нового облікового запису користувача
Actors	Неавтентифікований користувач
Trigger	Користувач бажає зареєструватися
Pre-conditions	-
Flow of Events	Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: ім'я користувача, пошта, пароль в системі. Після заповнення даних користувача натискає кнопку реєстрації. Опісля з'являється повідомлення про успішну реєстрацію та заклик підтвердити введену пошту, а на введену пошту прийде лист з посиланням для підтвердження. Користувач переходить за надісланим посиланням, з'являється повідомлення про успішне підтвердження пошти, відбувається перенаправлення на сторінку входу.
Extension	У випадку введення не коректних даних, кнопка реєстрації стає неактивною. Якщо якесь конкретне поле введено некоректно, то воно підсвічується червоним надписом.
Post-Condition	Створення облікового запису користувача, перехід на сторінку входу

Таблиця Б.2 – Варіант використання UC-02

Use case name	Вхід в обліковий запис
Use case ID	UC-02
Goals	Вхід користувача в зареєстрований обліковий запис у системі
Actors	Неавтентифікований користувач
Trigger	Користувач бажає увійти в обліковий запис

Продовження таблиці Б.2

Pre-conditions	-
Flow of Events	Користувач переходить на сторінку входу в обліковий запис. В поля вводяться відповідні дані: пошта користувача, пароль в системі. Після заповнення даних користувача натискає кнопку надсилання запиту входження в обліковий запис. Після цього з'являється повідомлення про успішне входження, і користувач перенаправляється на домашню.
Extension	У випадку введення не коректних даних, кнопка входу стає неактивною. Якщо якийсь конкретне поле введено некоректно, то воно підсвічується червоним надписом.
Post-Condition	Автентифікація користувача, перехід на домашню сторінку.

Таблиця Б.3 – Варіант використання UC-03

Use case name	Редагування профілю
Use case ID	UC-03
Goals	Оновлення інформації облікового запису користувача
Actors	Автентифікований користувач
Trigger	Користувач бажає змінити дані свого облікового запису
Pre-conditions	Користувач автентифікований у системі
Flow of Events	Користувач переходить на сторінку редагування профілю та вводить у поля, що цікавлять його, нові дані. Після заповнення оновленої інформації користувач натискає кнопку збереження змін, внаслідок чого з'являється повідомлення про успішне оновлення профілю та відбувається переадресація на сторінку перегляду даних профілю.
Extension	Якщо не всі необхідні поля було заповнено або їхні значення є незадовільними, кнопка збереження стає неактивною, а відповідні поля підсвічуються червоним.
Post-Condition	Інформація профілю користувача оновлена

Таблиця Б.4 – Варіант використання UC-04

Use case name	Налаштування відслідковуваних нутрієнтів
Use case ID	UC-04
Goals	Налаштування списку нутрієнтів, які користувач бажає відслідковувати
Actors	Автентифікований користувач
Trigger	Користувач бажає змінити перелік нутрієнтів, які необхідно відслідковувати
Pre-conditions	Користувач автентифікований у системі
Flow of Events	Користувач переходить на секцію налаштування відслідковування нутрієнтів у профілі та обирає з переліку ті, які бажає відслідковувати. Після цього він натискає кнопку збереження налаштувань, внаслідок чого з'являється повідомлення про успішне збереження та відбувається переадресація на сторінку перегляду списку нутрієнтів, які користувач бажає відслідковувати.
Extension	Якщо жодного нутрієнта не було обрано, система повідомляє про необхідність обрати хоча б один, а кнопка збереження змін стає неактивною.
Post-Condition	Список нутрієнтів, які користувач бажає відслідковувати, оновлено.

Таблиця Б.5 – Варіант використання UC-05

Use case name	Перегляд каталогу їжі
Use case ID	UC-05
Goals	Перегляд доступних продуктів харчування в каталозі
Actors	Автентифікований користувач
Trigger	Користувач бажає переглянути перелік продуктів харчування
Pre-conditions	Користувач автентифікований у системі

Продовження таблиці Б.5

Flow of Events	Користувач натискає на кнопку додавання запису споживання їжі на сторінці харчового щоденника, після чого відкривається модальне вікно з каталогом їжі. Користувач натискає на рядки з їжею, що його цікавить, після чого відбудеться відображення нутріційних деталей обраного продукту.
Extension	Якщо користувач не може знайти необхідної їжі серед усіх наведених на екрані, він натискає на поле пошуку, вводить запит, після чого відбувається оновлення результатів, що збігаються із вказаним пошуковим запитом.
Post-Condition	Користувач ознайомлюється з актуальним вмістом каталогу їжі та детальною інформацією про продукт, що його цікавить.

Таблиця Б.6 – Варіант використання UC-06

Use case name	Додавання запису споживання їжі
Use case ID	UC-06
Goals	Додавання нового запису про споживання продукту
Actors	Автентифікований користувач
Trigger	Користувач бажає додати новий запис про споживання
Pre-conditions	Користувач автентифікований у системі
Flow of Events	Користувач натискає на кнопку додавання запису споживання їжі на сторінці харчового щоденника, після чого відкривається модальне вікно з каталогом їжі. У даному вікні він обирає продукт, що його цікавить, одиницю виміру та кількість даних одиниць, що була спожита. Опісля він натискає кнопку збереження, у результаті чого з'являється повідомлення про додавання запису, а модальне вікно закривається.
Extension	Якщо не було обрано продукт, або кількість спожитих одиниць виміру, кнопка збереження стає неактивною, а відповідні поля підсвічуються червоним.

Продовження таблиці Б.6

Post-Condition	Новий запис про споживання збережено
----------------	--------------------------------------

Таблиця Б.7 – Варіант використання UC-07

Use case name	Редагування запису споживання
Use case ID	UC-07
Goals	Редагування наявного запису про споживання
Actors	Автентифікований користувач
Trigger	Користувач бажає оновити наявний запис про споживання
Pre-conditions	Користувач автентифікований у системі
Flow of Events	Користувач переходить на сторінку харчового щоденника, обирає запис, що його цікавить, та натискає кнопку редагування. Опісля відкривається модальне вікно з каталогом продуктів, де відображається зокрема обраний продукт, обрані одиниці виміру та спожита кількість. Користувач оновлює параметри, що його цікавлять, а по завершенню натискає кнопку збереження, після чого з'являється повідомлення про успішне оновлення запису та відбувається закриття модального вікна.
Extension	Якщо не було обрано продукт, або кількість спожитих одиниць виміру, кнопка збереження стає неактивною, а відповідні поля підсвічуються червоним.
Post-Condition	Оновлений запис про споживання збережено

Таблиця Б.8 – Варіант використання UC-08

Use case name	Планування додавання записів споживання
Use case ID	UC-08

Продовження таблиці Б.8

Goals	Створення процесу для автоматичного логування споживання їжі
Actors	Автентифікований користувач
Trigger	Користувач бажає налаштувати автоматичне додавання майбутніх записів про споживання
Pre-conditions	Користувач автентифікований у системі
Flow of Events	Користувач переходить на сторінку планування через навігаційну панель і натискає “Schedule logging”. Відбувається передаресация на сторінку додавання записів планування, де він налаштовує хронометричну складову планувальника, після чого обирає продукт, вказує кількість та натискає кнопку збереження. Опісля система виводить повідомлення про успішне створення автоматизації, після чого відбувається переадресация на оновлений список планувальників.
Extension	Якщо не було обрано продукт, або кількість спожитих одиниць виміру, кнопка збереження стає неактивною, а відповідні поля підсвічуються червоним.
Post-Condition	Оновлений запис про споживання збережено

Таблиця Б.9 – Варіант використання UC-09

Use case name	Пошук їжі в каталозі за назвою
Use case ID	UC-09
Goals	Пошук продуктів харчування в каталозі за назвою
Actors	Користувач автентифікований у системі
Trigger	Користувач бажає налаштувати автоматичне додавання майбутніх записів про споживання
Pre-conditions	Користувач автентифікований у системі

Продовження таблиці Б.9

Flow of Events	Користувач натискає на кнопку додавання запису споживання їжі на сторінці харчового щоденника, після чого відкривається модальне вікно з каталогом їжі. Користувач натискає на поле пошуку, уводить запит, після чого відбувається оновлення результатів, що збігаються із вказаним пошуковим запитом.
Extension	-
Post-Condition	Користувач бачить результати пошуку або повідомлення про відсутність збігів

Таблиця Б.10 – Варіант використання UC-10

Use case name	Перегляд доступних публічних рецептів
Use case ID	UC-10
Goals	Ознайомлення із доступними публічними рецептами
Actors	Автентифікований користувач
Trigger	Користувач бажає переглянути публічні рецепти
Pre-conditions	Користувач автентифікований у системі
Flow of Events	Користувач натискає на кнопку перегляду рецептів на навігаційній панелі, після чого відбувається переадресація на сторінку каталогу публічних рецептів. За потреби, користувач має можливість натиснути на поле пошуку, ввести пошуковий запит та натиснути на кнопку лупи, після чого відбувається відображення результатів, що збігаються із вказаним запитом.
Extension	-
Post-Condition	Користувач ознайомлений із наявними публічними рецептами каталогу

Таблиця Б.11 – Варіант використання UC-11

Use case name	Перегляд рецептів користувача
Use case ID	UC-11
Goals	Ознайомлення із доступними рецептами користувача
Actors	Автентифікований користувач
Trigger	Користувач бажає переглянути власні рецепти
Pre-conditions	Користувач автентифікований у системі
Flow of Events	Користувач натискає на кнопку «My recipes» в навігаційній панелі, після чого відбувається переадресація на сторінку перегляду переліку власних рецептів. За потреби користувач може скористатися полем пошуку, вказати текст запиту та натиснути на кнопку лупи, після чого здійснюється фільтрація результатів за відповідністю назви.
Extension	-
Post-Condition	Користувач ознайомлений із наявними власними рецептами

Таблиця Б.12 – Варіант використання UC-12

Use case name	Редагування рецепту користувача
Use case ID	UC-12
Goals	Оновлення інформації рецепту користувача
Actors	Автентифікований користувач
Trigger	Користувач бажає оновити інформацію власного рецепту
Pre-conditions	Користувач автентифікований у системі

Продовження таблиці Б.12

Flow Events	of	Користувач натискає на «My Recipes» у навігаційній панелі, після чого відбувається переадресація на сторінку перегляду переліку власних рецептів. Опісля, він обирає рецепт для перегляду деталей шляхом натискання на нього та натискає на кнопку редагування, після чого відбувається завантаження сторінки редагування рецепта. Тоді користувач змінює інформацію, що його цікавить: назва, перелік інгредієнтів та їхні кількості, кроки приготування, тощо, – та натискає на кнопку “Save”, після чого оновлені дані рецепта зберігаються в базі даних, а користувач отримує повідомлення про успішне редагування та передаресовується на сторінку перегляду власних рецептів.
Extension		Якщо не було обрано інгредієнти, назву, опис, тощо, кнопка збереження стає неактивною, а відповідні поля підсвічуються червоним.
Post-Condition		Дані рецепта оновлено

Таблиця Б.13 – Варіант використання UC-13

Use case name	Пошук їжі за штрих-кодом
Use case ID	UC-13
Goals	Ідентифікація їжі за даними штрих-коду
Actors	Автентифікований користувач
Trigger	Користувач бажає знайти продукт за допомогою сканування штрих-коду
Pre-conditions	Користувач автентифікований у системі

Продовження таблиці Б.13

Flow Events	of	Користувач натискає на кнопку додавання запису споживання їжі на сторінці харчового щоденника, після чого відкривається модальне вікно з каталогом їжі. Тоді він натискає на кнопку сканування штрих-коду, після чого відбувається активація камери пристрою користувача. Користувач підносить штрих-код до камери, а система, зчитавши відображений штрих-код, розпізнає отримане зображення, виводить повідомлення про успішне зчитування та наводить інформацію про продукт на модальному вікні.
Extension		Якщо штрих-код не було розпізнано, система інформує користувача про необхідність повторного сканування. У разі відсутності їжі, пов'язаної зі штрих-кодом, у базі даних користувач отримує повідомлення про відсутність інформації.
Post-Condition		Користувач бачить інформацію про їжу, асоційовану зі штрих-кодом

Таблиця Б.14 – Варіант використання UC-14

Use case name	Перегляд статистики споживання нутрієнтів
Use case ID	UC-14
Goals	Ознайомитися з обсягами споживання обраних нутрієнтів за заданий період
Actors	Автентифікований користувач
Trigger	Користувач бажає переглянути статистику споживання нутрієнтів
Pre-conditions	Користувач автентифікований у системі

Продовження таблиці Б.14

Flow Events	of Користувач натискає на вкладку «Nutrients Consumption» в навігаційній панелі, після чого відбувається переадресація на сторінку зі зведеною інформацією щодо споживання нутрієнтів за останній тиждень. За потреби користувач може натиснути на поле вибору періоду, вказати потрібні дати у календарі та натиснути на кнопку «Apply», після чого відбувається оновлення нутріційних відомостей у відповідності до обраного часового проміжку.
Extension	-
Post-Condition	Користувач ознайомлений із наявними даними щодо споживання нутрієнтів

Таблиця Б.15 – Варіант використання UC-15

Use case name	Перегляд записів споживання
Use case ID	UC-15
Goals	Ознайомитися з переліком записів про споживання продуктів для обраного дня
Actors	Автентифікований користувач
Trigger	Користувач бажає переглянути свої записи споживання
Pre-conditions	Користувач автентифікований у системі
Flow Events	of Користувач натискає на “Diary” в навігаційній панелі, після чого відбувається переадресація на сторінку зі зведеною інформацією щодо записів споживання, а також спожитих нутрієнтів за встановлений день. За потреби користувач може вказати потрібний день у календарі та натиснути на нього, після чого відбувається оновлення нутріційних відомостей у відповідності до обраної дати.
Extension	-

Продовження таблиці Б.15

Post-Condition	Користувач ознайомлений із наявними даними щодо споживання нутрієнтів
----------------	---

Таблиця Б.16 – Варіант використання UC-16

Use case name	Редагування публічного рецепту
Use case ID	UC-16
Goals	Оновлення інформації публічного рецепту
Actors	Адміністратор
Trigger	Користувач бажає оновити інформацію публічного рецепту
Pre-conditions	Користувач має права адміністратора
Flow of Events	Користувач натискає на кнопку перегляду рецептів на навігаційній панелі, після чого відбувається переадресація на сторінку каталогу публічних рецептів. Опісля, він обирає рецепт для перегляду деталей шляхом натискання на нього та натискає на кнопку редагування, після чого відбувається завантаження сторінки редагування рецепта. Тоді користувач змінює інформацію, що його цікавить: назва, перелік інгредієнтів та їхні кількості, кроки приготування, тощо, – та натискає на кнопку “Save”, після чого оновлені дані рецепта зберігаються в базі даних, а користувач отримує повідомлення про успішне редагування та переадресовується на сторінку перегляду рецептів.
Extension	Якщо не було обрано інгредієнти, назву, опис, тощо, кнопка збереження стає неактивною, а відповідні поля підсвічуються червоним.
Post-Condition	Дані публічного рецепта оновлено

Таблиця Б.17 – Варіант використання UC-17

Use case name	Створення публічного рецепту
Use case ID	UC-17
Goals	Додавання публічного рецепту
Actors	Адміністратор
Trigger	Користувач бажає додати новий публічний рецепт
Pre-conditions	Користувач має права адміністратора
Flow of Events	Користувач натискає на кнопку перегляду рецептів на навігаційній панелі, після чого відбувається переадресація на сторінку каталогу публічних рецептів. Опісля, натискає на кнопку “Add new”, після чого відбувається завантаження сторінки створення рецепта. Тоді користувач змінює інформацію, що його цікавить: назва, перелік інгредієнтів та їхні кількості, кроки приготування, тощо, – та натискає на кнопку “Save”, після чого дані нового рецепта зберігаються в базі даних, а користувач отримує повідомлення про успішне створення та переадресовується на сторінку перегляду рецептів.
Extension	Якщо не було обрано інгредієнти, назву, опис, тощо, кнопка збереження стає неактивною, а відповідні поля підсвічуються червоним.
Post-Condition	Створено новий публічний рецепт

Таблиця Б.18 – Варіант використання UC-18

Use case name	Формування раціону сервісом рекомендації
Use case ID	UC-18
Goals	Надання користувачу персоналізованого раціону на основі встановлених нутріційних цілей
Actors	Автентифікований користувач

Продовження таблиці Б.18

Trigger	Користувач ініціює запит на отримання рекомендованого раціону
Pre-conditions	Користувач автентифікований у системі
Flow of Events	Користувач переходить на вкладку «Meal Planner» в навігаційній панелі, після чого відбувається переадресація на сторінку формування раціону. Користувач задає цілі за макрота мікронутрієнтами, обирає кількість страв та їхні характеристики і натискає кнопку застосування, після чого на екрані користувача відображається запропонований раціон.
Extension	-
Post-Condition	Персоналізований раціон відображено на екрані користувача

Таблиця Б.19 – Варіант використання UC-19

Use case name	Збереження рекомендованого раціону
Use case ID	UC-19
Goals	Збереження сформованого рекомендованого раціону у профілі користувача
Actors	Автентифікований користувач
Trigger	Користувач бажає зберегти рекомендований раціон
Pre-conditions	Користувач автентифікований у системі та перебуває на сторінці планувальника із запропонованим раціоном
Flow of Events	Користувач переглядає сформований раціон на екрані, після чого натискає кнопку «Save Plan», після чого запропоновані рецепти додаються в колекцію користувача, а в його щоденник харчування додаються записи про споживання.
Extension	-
Post-Condition	Рекомендований раціон збережено у профілі користувача

Таблиця Б.20 – Варіант використання UC-20

Use case name	Додавання їжі
Use case ID	UC-20
Goals	Внесення нових продуктів у каталог
Actors	Адміністратор
Trigger	Адміністратор ініціює процес додавання нового продукту до каталогу
Pre-conditions	Користувач має права адміністратора
Flow of Events	Адміністратор натискає кнопку «Add Food» у навігаційній панелі, після чого відбувається переадресація на сторінку створення нового продукту, на якій відображається форма з полями для введення назви, харчової цінності на 100 г, одиниць виміру, а також область для завантаження зображення. Опісля адміністратор заповнює необхідні поля та натискає кнопку «Save», у результаті чого користувачу відображається повідомлення про успішне додавання та відбувається переадресація на домашню сторінку.
Extension	Якщо не було вказано значення з наведених полів, виводиться повідомлення про відсутність даних, а відповідні поля підсвічуються червоним.
Post-Condition	Нову їжу збережено

Таблиця Б.21 – Варіант використання UC-21

Use case name	Додавання запису споживання рецепту
Use case ID	UC-21
Goals	Створення нового запису про споживання користувачем обраного рецепту
Actors	Автентифікований користувач
Trigger	Користувач бажає додати новий запис споживання рецепту

Продовження таблиці Б.21

Pre-conditions	Користувач автентифікований у системі
Flow of Events	Користувач натискає кнопку додавання запису споживання страви на сторінці харчового щоденника, після чого відкривається модальне вікно перегляду власних рецептів. Надалі користувач обирає рецепт, зазначає кількість спожитого рецепту, вказуючи одиницю виміру та числове значення. Опісля він натискає кнопку збереження, у результаті чого з'являється повідомлення про додавання запису, а модальне вікно закривається.
Extension	Якщо не було обрано рецепт, або кількість спожитих одиниць виміру, кнопка збереження стає неактивною, а відповідні поля підсвічуються червоним.
Post-Condition	Новий запис про споживання збережено

Таблиця Б.22 – Варіант використання UC-22

Use case name	Вихід із облікового запису
Use case ID	UC-22
Goals	Деавтентифікація користувача
Actors	Автентифікований користувач
Trigger	Користувач бажає вийти з облікового запису
Pre-conditions	Користувач автентифікований у системі
Flow of Events	Користувач натискає кнопку на кнопку виходу з облікового запису, після чого відбувається переадресація на домашню сторінку та вихід із акаунта.
Extension	-
Post-Condition	Користувач вийшов із облікового запису

Таблиця Б.23 – Варіант використання UC-23

Use case name	Створення власного рецепту
Use case ID	UC-23
Goals	Додавання власного рецепту
Actors	Автентифікований користувач
Trigger	Користувач бажає додати власний рецепт
Pre-conditions	Користувач має права адміністратора
Flow of Events	Користувач натискає на кнопку перегляду власних рецептів на навігаційній панелі, після чого відбувається переадресація на сторінку каталогу публічних рецептів. Опісля, натискає на кнопку “Add new”, після чого відбувається завантаження сторінки створення рецепта. Тоді користувач змінює інформацію, що його цікавить: назва, перелік інгредієнтів та їхні кількості – та натискає на кнопку “Save”, після чого інформація про новий рецепт зберігаються в базі даних, а користувач отримує повідомлення про успішне створення та переадресовується на сторінку перегляду рецептів.
Extension	Якщо не було обрано інгредієнти, назву, опис, тощо, кнопка збереження стає неактивною, а відповідні поля підсвічуються червоним.
Post-Condition	Створено новий власний рецепт

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ВЕБЗАСТОСУНОК ДЛЯ ВІДСЛІДКОВУВАННЯ МАКРО- ТА
МІКРОНУТРІЄНТІВ У РАЦІОНІ**

Текст програми

КПІ.ПІ-1527.045440.03.12

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена МАРЧЕНКО

Нормоконтроль:

_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:

_____ Артем ХІЛЬЧУК

Київ – 2025

Посилання на репозиторій з повним текстом програмного коду
<https://github.com/KhilchukArtemIP-14/NutriMatrix>

RecipeRecommendationService.cs

Реалізація функціональної задачі формування рекомендованого раціону

```
using Microsoft.EntityFrameworkCore;
using Qdrant.Client.Grpc;
using RecommendationService.Api.Data;
using RecommendationService.Api.Models;
using RecommendationService.Api.Models.Dto;
using RecommendationService.Api.Models.Redis;
using RecommendationService.Api.Services.NutrientsAnalysisService;
using RecommendationService.Api.Services.Qdrant;
using Redis.OM;
using Redis.OM.Searching;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;

namespace RecommendationService.Api.Services.RecommendationService
{
    public partial class RecipeRecommendationService :
    IRecipeRecommendationService
    {
        private readonly RedisCollection<RecipeShortcutRedis> _recipesCollection;
        private readonly IQdrantService _qdrant;
        private readonly RecipeDbContext _db;

        private static readonly TimeSpan _timeLimit = TimeSpan.FromSeconds(7);
        private Stopwatch _searchStopwatch;

        private readonly Random _rng = new Random();

        public RecipeRecommendationService(
            RedisCollection<RecipeShortcutRedis> recipesCollection,
            IQdrantService qdrant,
            RecipeDbContext db)
        {
            _recipesCollection = recipesCollection;
            _qdrant = qdrant;
        }
    }
}
```

[illegible]

```

        .Select(m => m.FoodId)
        .Distinct()
        .ToList(),
        NutrientAmounts = fromDb.NutrientsPerTotalServings
        .Where(n => !n.IsDeleted)
        .ToDictionary(n => n.NutrientId, n => n.Amount)
    };
}
}
if (item != null && item.Servings > 0)
    candidates.Add(item);
}
candidatesList.Add(candidates);
}

int populationSize = 50;
int numSlots = dto.RecipeRequests.Count();
var population = new List<Individual>(populationSize);
Individual bestIndividual = null;

for (int i = 0; i < populationSize; i++)
{
    var indiv = CreateRandomIndividual(candidatesList, numSlots);
    indiv.Fitness = EvaluateFitness(indiv, dto.NutritionalGoals);
    population.Add(indiv);
    if (bestIndividual == null || indiv.Fitness < bestIndividual.Fitness)
        bestIndividual = indiv;
}

while (_searchStopwatch.Elapsed < _timeLimit)
{
    var newPopulation = new List<Individual>(populationSize);
    for (int i = 0; i < populationSize; i++)
    {
        var parent1 = TournamentSelect(population, 3);
        var parent2 = TournamentSelect(population, 3);

        var child = Crossover(parent1, parent2, numSlots);

        Mutate(child, candidatesList);
        child.Fitness = EvaluateFitness(child, dto.NutritionalGoals);

        newPopulation.Add(child);
        if (child.Fitness < bestIndividual.Fitness) bestIndividual = child;
    }
}

```

```

        if (_searchStopwatch.Elapsed >= _timeLimit) break;
    }

    population = newPopulation;
}

var bestCombo = bestIndividual.Genes
    .Select(g => (g.Recipe, g.Amount))
    .Reverse()
    .ToList();
sw.Stop();

var totalDistance = (float)Math.Sqrt(dto.NutritionalGoals.Keys
    .Sum(k =>
    {
        var diff = dto.NutritionalGoals[k] - bestCombo
            .Sum(rc => rc.Recipe.NutrientAmounts.GetValueOrDefault(k, 0f) *
rc.Amount / rc.Recipe.Servings); ;
        return diff * diff / (dto.NutritionalGoals[k] * dto.NutritionalGoals[k]);
    }));

var aggregated = bestCombo
    .SelectMany(rc => rc.Recipe.NutrientAmounts
        .Where(n => dto.NutritionalGoals.ContainsKey(n.Key))
        .ToDictionary(n => n.Key, n => n.Value * rc.Amount /
rc.Recipe.Servings))
    .GroupBy(p => p.Key)
    .ToDictionary(g => g.Key, g => g.Sum(x => x.Value));

return new RecommendationResponseDto
{
    RecipesAndAmounts = bestCombo
        .Select(rc => new RecipeWithAmountDto
        {
            Recipe = new RecipeShortcutDto { Id = rc.Recipe.Id, TotalServings
= rc.Recipe.Servings, Name = rc.Recipe.Title },
            Amount = rc.Amount
        })
        ,
    TimeToRespondInMs = sw.ElapsedMilliseconds,
    TotalDistance = totalDistance,
    Nutrients = aggregated
};
}

```

```

private Individual CreateRandomIndividual(
    List<List<RecipeShortcutRedis>> candidatesList,
    int numSlots)
{
    var indiv = new Individual
    {
        Genes = new Gene[numSlots]
    };
    for (int i = 0; i < numSlots; i++)
    {
        var slotCandidates = candidatesList[i];
        if (slotCandidates.Count == 0)
        {
            indiv.Genes[i] = new Gene { Recipe = null, Amount = 0 };
            continue;
        }
        var recipe = slotCandidates[_rng.Next(slotCandidates.Count)];
        int amount = _rng.Next(1, Convert.ToInt32(recipe.Servings * 2) + 1);
        indiv.Genes[i] = new Gene { Recipe = recipe, Amount = amount };
    }
    return indiv;
}

private float EvaluateFitness(Individual indiv, Dictionary<int, float>
expectations)
{
    float sumSq = 0f;
    var current = new Dictionary<int, float>();
    foreach (var gene in indiv.Genes)
    {
        var recipe = gene.Recipe;
        var amount = gene.Amount;
        if (recipe == null) continue;
        foreach (var kvp in recipe.NutrientAmounts)
        {
            float contrib = kvp.Value * amount / recipe.Servings;
            current[kvp.Key] = current.GetValueOrDefault(kvp.Key, 0f) + contrib;
        }
    }
    foreach (var k in expectations.Keys)
    {
        float expVal = expectations[k];
        float actual = current.GetValueOrDefault(k, 0f);
        float diff = expVal - actual;
        sumSq += (diff * diff) / (expVal * expVal);
    }
}

```

```

    }
    return (float)Math.Sqrt(sumSq);
}

private Individual TournamentSelect(List<Individual> population, int
tournoiSize)
{
    Individual best = null;
    for (int i = 0; i < tournoiSize; i++)
    {
        var candidate = population[_rng.Next(population.Count)];
        if (best == null || candidate.Fitness < best.Fitness) best = candidate;
    }
    return best;
}

private Individual Crossover(Individual parent1, Individual parent2, int
numSlots)
{
    var child = new Individual { Genes = new Gene[numSlots] };
    int crossPoint = _rng.Next(1, numSlots);
    for (int i = 0; i < numSlots; i++)
    {
        var source = (i < crossPoint) ? parent1 : parent2;
        child.Genes[i] = new Gene
        {
            Recipe = source.Genes[i].Recipe,
            Amount = source.Genes[i]?.Amount ?? 0
        };
    }
    return child;
}

private void Mutate(Individual indiv, List<List<RecipeShortcutRedis>>
candidatesList)
{
    const float recipeMutProb = 0.1f;
    const float amountMutProb = 0.2f;

    for (int i = 0; i < indiv.Genes.Length; i++)
    {
        if (_rng.NextDouble() < recipeMutProb)
        {
            var slotCandidates = candidatesList[i];
            if (slotCandidates.Count > 0)

```

```

        {
            var newRecipe = slotCandidates[_rng.Next(slotCandidates.Count)];
            indiv.Genes[i].Recipe = newRecipe;

            indiv.Genes[i].Amount = _rng.Next(1,
Convert.ToInt32(newRecipe.Servings*2) + 1);
        }
    }
    else if (_rng.NextDouble() < amountMutProb && indiv.Genes[i].Recipe
!= null)
    {
        float maxAmt = indiv.Genes[i].Recipe.Servings;
        indiv.Genes[i].Amount = _rng.Next(1, Convert.ToInt32(maxAmt*2) +
1);
    }
}
}
}
}
}

```

```

internal static class DictionaryExtensions
{
    public static TValue GetValueOrDefault<TKey, TValue>(
        this IDictionary<TKey, TValue> dict, TKey key, TValue defaultValue =
default)
    {
        return dict.TryGetValue(key, out var val) ? val : defaultValue;
    }
}

```

Файл QdrantService.cs

Реалізація функціональної задачі взаємодії з QdrantDb

```
using Google.Protobuf.Collections;
```

```
using Qdrant.Client;
```

```
using Qdrant.Client.Grpc;
```

```
using System.Net.NetworkInformation;
```

```
using static Qdrant.Client.Grpc.Conditions;
```

```
namespace RecommendationService.Api.Services.Qdrant
```

```

{
    public class QdrantService : IQdrantService
    {

```

```

private readonly QdrantClient _qdrantClient;
private readonly string _collectionName = "recipe_nutrients";
public QdrantService(QdrantClient qdrantClient)
{
    _qdrantClient = qdrantClient;
}
public async Task CreateCollectionAsync()
{
    await _qdrantClient.CreateCollectionAsync(_collectionName, new
VectorParams
    {
        Size = 161,
        Distance = Distance.Cosine,
        HnswConfig = new HnswConfigDiff
        {
            M = 16,
            EfConstruct = 100
        }
    });
}

public async Task<UpdateStatus> InsertRecipeVectorAsync(int id,
IEnumerable<float> rawVector, string category, List<string> ingredientIds)
{
    if (rawVector.Count() != 161) throw new ArgumentException("Incorrect
vector size");

    var point = new PointStruct()
    {
        Id = (ulong)id,

```

```

    Vectors = PerformUnitNormalization(rawVector).ToArray(),
    Payload = {
        ["recipeId"] = id,
        ["category"] = category,
        ["ingredients"] = ingredientIds.ToArray()
    }
};

```

```

    var updateResult = await _qudrantClient.UpsertAsync(_collectionName,
new List<PointStruct> { point });

```

```

    return updateResult.Status;
}

```

```

public async Task<List<int>> FindKNearestNeighborsAsync(
    int k,
    IEnumerable<float> rawSearchVector,
    IEnumerable<int>? excludeIds = null,
    string? category = null,
    List<string>? includeIngredientIds = null,
    List<string>? excludeIngredientIds = null)
{
    Filter filterOptions = null;

    if (category != null)
    {
        filterOptions = MatchKeyword("category", category);
    }

    if (excludeIds != null)

```

```

    {
        foreach (var id in excludeIds)
        {
            if (filterOptions == null) filterOptions = !MatchKeyword("recipeId",
id.ToString());
            else filterOptions &= !MatchKeyword("recipeId", id.ToString());
        }
    }
    if (includeIngredientIds != null)
    {
        foreach (var ing in includeIngredientIds)
        {
            if (filterOptions == null) filterOptions = MatchKeyword("ingredients",
ing);
            else filterOptions &= MatchKeyword("ingredients", ing);
        }
    }
    if (excludeIngredientIds != null)
    {
        foreach (var ing in excludeIngredientIds)
        {
            if (filterOptions == null) filterOptions =
!MatchKeyword("ingredients", ing);
            else filterOptions &= !MatchKeyword("ingredients", ing);
        }
    }

    var queryVector =
PerformUnitNormalization(rawSearchVector).ToArray();

```

```

        var results = await _qudrantClient.SearchAsync(_collectionName,
queryVector, limit: (ulong)k, filter: filterOptions);

        return results.Select(r => Convert.ToInt32(r.Id.Num)).ToList();
    }

    private IEnumerable<float> PerformUnitNormalization(IEnumerable<float>
rawVector)
    {
        var norm = MathF.Sqrt(rawVector.Sum(x => x * x));

        return rawVector.Select(x => x / norm);
    }

    public async Task DeleteAllRecordsAsync()
    {
        await _qudrantClient.DeleteCollectionAsync(_collectionName);
    }
}

```

Файл GetMealByIdConsumer.cs

Реалізація функціональної задачі обробки запитів на відомості щодо приватного рецепту

```

using BuildingBlocks.Messaging.Events;
using BuildingBlocks.Messaging.Responses;
using FoodCatalog.Api.Data.Context;
using FoodCatalog.Api.Models.Dto;
using MassTransit;
using Microsoft.EntityFrameworkCore;

```

```

namespace FoodCatalog.Api.Integration
{
    public class GetMealByIdConsumer : IConsumer<GetMealByIdRequest>
    {
        private readonly FoodCatalogDbContext _context;

        public GetMealByIdConsumer(FoodCatalogDbContext context)
        {
            _context = context;
        }

        public async Task Consume(ConsumeContext<GetMealByIdRequest>
context)
        {
            var meal = await _context.Meals
                .Include(m => m.FoodMeals)
                .ThenInclude(fm => fm.Measure)
                .FirstOrDefaultAsync(m => m.Id == context.Message.Id &&
!m.IsDeleted);

            if (meal is null)
            {
                throw new InvalidOperationException($"Meal with ID
{context.Message.Id} not found");
            }

            var dto = new MealResponseDto(
                meal.Id,
                meal.Name,
                meal.AddedBy,
                meal.TotalServings,

```

```

meal.FoodMeals
    .Where(fm => !fm.IsDeleted)
    .Select(fm => new FoodMealResponseDto(
        fm.Id,
        fm.MeasureId,
        fm.Quantity,
        fm.Measure?.Name ?? "",
        fm.Measure?.WeightInGrams ?? 0f
    )).ToList()
);

await context.RespondAsync(dto);
}
}
}

```

Файл MealFetcher.cs

Реалізація функціональної задачі надсилання запитів відомостей щодо приватного рецепту

```

using BuildingBlocks.Messaging.Events;
using BuildingBlocks.Messaging.Responses;
using MassTransit;

namespace FoodRecords.Api.Services.MealFetcher
{
    public class MealFetcher : IMealFetcher
    {
        private readonly IRequestClient<GetMealByIdRequest> _requestClient;
    }
}

```

```

public MealFetcher(IRequestClient<GetMealByIdRequest> requestClient)
{
    _requestClient = requestClient;
}

public async Task<MealResponseDto?> FetchMealAsync(long mealId)
{
    var response = await _requestClient.GetResponse<MealResponseDto>(new
GetMealByIdRequest() { Id=mealId});
    return response.Message;
}
}

```

Файл RegisterCommand.cs

Реалізація функціональної задачі реєстрації користувачів

```

using Auth.API.Data;
using Auth.API.Models.DTO;
using Auth.API.Models;
using Auth.API.Services.EmailService;
using Microsoft.AspNetCore.Identity;
using System.Security.Claims;
using System.Web;
using MediatR;

public class RegisterCommand : IRequest<IdentityResult>
{
    public RegisterUserDTO RegisterUserDTO { get; set; }
}

```

```

public class RegisterCommandHandler : IRequestHandler<RegisterCommand,
IdentityResult>
{
    private readonly UserManager<IdentityUser> _userManager;
    private readonly IConfiguration _config;
    private readonly IEmailService _emailService;
    private readonly AuthDbContext _dbContext;

    public RegisterCommandHandler(UserManager<IdentityUser> userManager,
IConfiguration config, IEmailService emailService, AuthDbContext dbContext)
    {
        _userManager = userManager;
        _config = config;
        _emailService = emailService;
        _dbContext = dbContext;
    }

    public async Task<IdentityResult> Handle(RegisterCommand request,
CancellationToken cancellationToken)
    {
        var dto = request.RegisterUserDTO;
        var identityUser = new IdentityUser
        {
            UserName = dto.Name,
            Email = dto.Email
        };

        var identityResult = await _userManager.CreateAsync(identityUser,
dto.Password);
    }
}

```

```

    if (!identityResult.Succeeded)
    {
        return identityResult;
    }

    if (dto.Roles != null && dto.Roles.Any())
    {
        identityResult = await _userManager.AddToRolesAsync(identityUser,
dto.Roles);
        if (!identityResult.Succeeded)
        {
            return identityResult;
        }
    }

    var nuteintIds = new List<long> { 301, 205, 601, 208, 606, 204, 605, 303,
291, 306, 307, 203, 269, 539, 324, 299, 1001, 1006, 1002, 290, 261, 260, 1003,
1004, 1005, 513, 221, 511, 207, 514, 454, 262, 639, 322, 321, 326, 421, 334, 312,
507, 268, 325, 610, 611, 696, 612, 625, 652, 697, 613, 626, 673, 662, 653, 687,
614, 617, 674, 663, 859, 618, 670, 675, 669, 619, 851, 685, 627, 615, 628, 672,
689, 852, 853, 620, 855, 629, 857, 624, 630, 858, 631, 621, 654, 671, 607, 608,
609, 645, 646, 693, 695, 313, 417, 431, 435, 432, 212, 287, 515, 211, 516, 512,
521, 503, 213, 504, 338, 337, 505, 214, 506, 304, 428, 315, 406, 573, 578, 257,
664, 676, 856, 665, 666, 305, 410, 508, 636, 517, 319, 405, 317, 518, 641, 209,
638, 210, 263, 404, 502, 323, 341, 343, 342, 501, 509, 510, 318, 320, 418, 415,
401, 328, 430, 429, 255, 309, 344, 345, 346, 347 };

    var nutrients = nuteintIds.Select(id => new NutrientTracking
    {
        NutrientId = id,
        UserId = identityUser.Id,

```

```

        TargetAmount = 0,
        IsActive = false
    }).ToList();
    await _dbContext.NutrientTrackings.AddRangeAsync(nutrients);
    await _dbContext.SaveChangesAsync();

    var token = await
    _userManager.GenerateEmailConfirmationTokenAsync(identityUser);
    var confirmationLink = $"{_config["ClientUrl"]}/auth/confirm-
    email?token={HttpUtility.UrlEncode(token)}&email={HttpUtility.UrlEncode(dto.
    Email)}";
    await _emailService.SendEmailAsync(dto.Email, "Email confirmation",
    $"Please confirm your email by clicking <a href='{confirmationLink}'>here</a>");

    return IdentityResult.Success;
}
}

```

Файл LoginQuery.cs

Реалізація функціональної задачі автентифікації користувачів

```

using Auth.API.Data;
using Auth.API.Models.DTO;
using MediatR;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

```

```

namespace Auth.API.Features.Queries

```

```

{
    public class LoginQuery : IRequest<LoginResponseDTO>
    {
        public LoginUserDTO LoginUserDTO { get; set; }
    }

    public class LoginQueryHandler : IRequestHandler<LoginQuery,
LoginResponseDTO>
    {
        private readonly UserManager<IdentityUser> _userManager;
        private readonly AuthDbContext _dbContext;
        private readonly IConfiguration _config;

        public LoginQueryHandler(UserManager<IdentityUser> userManager,
AuthDbContext dbContext, IConfiguration config)
        {
            _userManager = userManager;
            _dbContext = dbContext;
            _config = config;
        }

        public async Task<LoginResponseDTO> Handle(LoginQuery request,
CancellationToken cancellationToken)
        {
            var user = await
_userManager.FindByEmailAsync(request.LoginUserDTO.Email);

            if (user != null)
            {

```

```

        var passwordCheck = await _userManager.CheckPasswordAsync(user,
request.LoginUserDTO.Password);
        if (passwordCheck)
        {
            var roles = await _userManager.GetRolesAsync(user);
            var n = await _dbContext.NutrientTrackings.Where(r => r.UserId ==
user.Id).ToArrayAsync();

            if (roles != null)
            {
                var response = new LoginResponseDTO()
                {
                    Name = user.UserName,
                    Email = user.Email,
                    UserId = user.Id,
                    Roles = roles.ToArray(),
                    Token = await CreateJWTToken(user, roles.ToList()),
                    NutrientTrackings = n
                };

                return response;
            }
        }
        return null;
    }

    public async Task<string> CreateJWTToken(IdentityUser user, List<string>
roles)

```

```

{
    var claims = new List<Claim>
    {
        new Claim(ClaimTypes.Name, user.UserName),
        new Claim(ClaimTypes.Email, user.Email),
        new Claim("userId", user.Id),
    };
    foreach (var role in roles)
    {
        claims.Add(new Claim(ClaimTypes.Role, role));
    }

    var key = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_config["Jwt:Key"]));
    var credentials = new SigningCredentials(key,
SecurityAlgorithms.HmacSha256);

    var token = new JwtSecurityToken(
        _config["Jwt:Issuer"],
        _config["Jwt:Audience"],
        claims,
        expires: DateTime.Now.AddHours(1),
        signingCredentials: credentials
    );
    return new JwtSecurityTokenHandler().WriteToken(token);
}
}
}

```

Файл GoogleLoginCommand.cs

Реалізація функціональної задачі автентифікації користувачів за допомогою Google

```

using Auth.API.Data;
using Auth.API.Models.DTO;
using Auth.API.Models;
using FirebaseAdmin.Auth;
using MediatR;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace Auth.API.Features.Queries
{
    public class GoogleLoginCommand : IRequest<LoginResponseDTO>
    {
        public string IdToken { get; set; }
    }

    public class GoogleLoginCommandHandler :
    IRequestHandler<GoogleLoginCommand, LoginResponseDTO>
    {
        private readonly UserManager<IdentityUser> _userManager;
        private readonly AuthDbContext _dbContext;
        private readonly IConfiguration _config;
        public GoogleLoginCommandHandler(UserManager<IdentityUser>
userManager, AuthDbContext dbContext, IConfiguration config)
        {
            _userManager = userManager;
            _dbContext = dbContext;
            _config = config;
        }
        public async Task<string> CreateJWTToken(IdentityUser user, List<string>
roles)
        {
            var claims = new List<Claim>
            {
                new Claim(ClaimTypes.Name, user.UserName),
                new Claim(ClaimTypes.Email, user.Email),
                new Claim("userId", user.Id),
            };
            foreach (var role in roles)
            {
                claims.Add(new Claim(ClaimTypes.Role, role));
            }

```

```

        var key = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_config["Jwt:Key"]));
        var credentials = new SigningCredentials(key,
SecurityAlgorithms.HmacSha256);

        var token = new JwtSecurityToken(
            _config["Jwt:Issuer"],
            _config["Jwt:Audience"],
            claims,
            expires: DateTime.Now.AddHours(1),
            signingCredentials: credentials
        );
        return new JwtSecurityTokenHandler().WriteToken(token);
    }
    public async Task<LoginResponseDTO> Handle(GoogleLoginCommand
request, CancellationToken cancellationToken)
    {
        var decodedToken = await
FirebaseAuth.DefaultInstance.VerifyIdTokenAsync(request.IdToken);
        string email = decodedToken.Claims.TryGetValue("email", out var
emailClaim) ? emailClaim.ToString() : null;
        var user = await _userManager.FindByEmailAsync(email);

        if (user == null)
        {
            user = new IdentityUser { UserName = email, Email = email,
EmailConfirmed = true };
            var result = await _userManager.CreateAsync(user);
            if (!result.Succeeded)
            {
                return null;
            }

            var nuteintIds = new List<long> { 301, 205, 601, 208, 606, 204, 605, 303,
291, 306, 307, 203, 269, 539, 324, 299, 1001, 1006, 1002, 290, 261, 260, 1003, 1004,
1005, 513, 221, 511, 207, 514, 454, 262, 639, 322, 321, 326, 421, 334, 312, 507,
268, 325, 610, 611, 696, 612, 625, 652, 697, 613, 626, 673, 662, 653, 687, 614, 617,
674, 663, 859, 618, 670, 675, 669, 619, 851, 685, 627, 615, 628, 672, 689, 852, 853,
620, 855, 629, 857, 624, 630, 858, 631, 621, 654, 671, 607, 608, 609, 645, 646, 693,
695, 313, 417, 431, 435, 432, 212, 287, 515, 211, 516, 512, 521, 503, 213, 504, 338,
337, 505, 214, 506, 304, 428, 315, 406, 573, 578, 257, 664, 676, 856, 665, 666, 305,
410, 508, 636, 517, 319, 405, 317, 518, 641, 209, 638, 210, 263, 404, 502, 323, 341,
343, 342, 501, 509, 510, 318, 320, 418, 415, 401, 328, 430, 429, 255, 309, 344, 345,
346, 347 };

```

```

var nutrients = nuteintIds.Select(id => new NutrientTracking
{
    NutrientId = id,
    UserId = user.Id,
    TargetAmount = 0,
    IsActive = false
}).ToList();
await _dbContext.NutrientTrackings.AddRangeAsync(nutrients);
await _dbContext.SaveChangesAsync();
}

var roles = await _userManager.GetRolesAsync(user);
var nutrientTrackings = await _dbContext.NutrientTrackings.Where(r =>
r.UserId == user.Id).ToArrayAsync();
var token = await CreateJWTToken(user, roles.ToList());

return new LoginResponseDTO
{
    Name = user.UserName,
    Email = user.Email,
    UserId = user.Id,
    Roles = roles.ToArray(),
    Token = token,
    NutrientTrackings = nutrientTrackings
};
}
}
}
}

```

Файл TaskSchedulerService.cs

Реалізація функціональної задачі створення та управління планувальниками

```

using FoodRecords.Api.Data;
using FoodRecords.Api.Jobs;
using FoodRecords.Api.Models.Domain;
using Quartz;

namespace FoodRecords.Api.Services.TaskSchedulerService
{
    public class TaskSchedulerService : ITaskSchedulerService
    {
        private readonly IScheduler _scheduler;
        private readonly FoodRecordsDbContext _dbContext;

        public TaskSchedulerService(IScheduler scheduler, FoodRecordsDbContext
dbContext)

```

```

{
    _scheduler = scheduler;
    _dbContext = dbContext;
}

public async Task<FoodPlan> CreateScheduleAsync(ScheduleDto dto)
{
    if (!Enum.IsDefined(typeof(ConsumableType), dto.ConsumableType))
    {
        throw new ArgumentException("ConsumableType must be 'Food' or 'Recipe'");
    }

    if (string.IsNullOrEmpty(dto.UserId))
    {
        throw new ArgumentException("UserId cannot be null or empty");
    }

    var tempId = Guid.NewGuid().ToString();

    var jobKeyStr = $"objectJob-{tempId}";
    var triggerKeyStr = $"trigger-{tempId}";

    var foodPlan = new FoodPlan
    {
        ConsumableId = dto.ConsumableId,
        Amount = dto.Amount,
        RequiresConfirmation = dto.RequiresConfirmation,
        IsRecurring = dto.IsRecurring,
        RunAtUtc = dto.IsRecurring ? null : dto.RunAtUtc,
        CronExpression = dto.IsRecurring ? dto.CronExpression : null,
        UserId = dto.UserId,
        IsDeleted = false,
        ConsumableType = dto.ConsumableType,
        JobKey = jobKeyStr,
        TriggerKey = triggerKeyStr,
        Name = dto.Name
    };

    _dbContext.FoodPlans.Add(foodPlan);
    await _dbContext.SaveChangesAsync();

    var finalJobKeyStr = $"objectJob-{foodPlan.Id}";
    var finalTriggerKeyStr = $"trigger-{foodPlan.Id}";

```

```

        foodPlan.JobKey = finalJobKeyStr;
        foodPlan.TriggerKey = finalTriggerKeyStr;
        await _dbContext.SaveChangesAsync();

        var jobKey = new JobKey(finalJobKeyStr);
        var triggerKey = new TriggerKey(finalTriggerKeyStr);

        var jobDetail = JobBuilder.Create(dto.ConsumableType ==
ConsumableType.Food ? typeof(FoodRecordCreationJob) :
typeof(RecipeRecordCreationJob))
            .WithIdentity(jobKey)
            .UsingJobData(new JobDataMap
            {
                { "ObjectId", foodPlan.ConsumableId },
                { "Amount", (float)foodPlan.Amount },
                { "RequiresConfirmation", foodPlan.RequiresConfirmation },
                { "ScheduleId", foodPlan.Id },
                { "ObjectType", foodPlan.ConsumableType }
            })
            .Build();

        ITrigger trigger = foodPlan.IsRecurring
            ? TriggerBuilder.Create()
                .WithIdentity(triggerKey)
                .WithCronSchedule(foodPlan.CronExpression, x =>
x.InTimeZone(TimeZoneInfo.Utc))
                .Build()
            : TriggerBuilder.Create()
                .WithIdentity(triggerKey)
                .StartAt(foodPlan.RunAtUtc.Value)
                .Build();

        await _scheduler.ScheduleJob(jobDetail, trigger);

        return foodPlan;
    }

    public async Task<FoodPlan> UpdateScheduleAsync(ScheduleDto dto)
    {
        if (dto.Id == null || !Enum.IsDefined(typeof(ConsumableType),
dto.ConsumableType))
        {
            throw new ArgumentException("Invalid Id or ConsumableType;
ConsumableType must be 'Food' or 'Recipe'");
        }
    }

```

```

    }

    if (string.IsNullOrEmpty(dto.UserId))
    {
        throw new ArgumentException("UserId cannot be null or empty");
    }

    var foodPlan = await _dbContext.FoodPlans.FindAsync(dto.Id);
    if (foodPlan == null || foodPlan.IsDeleted)
    {
        throw new Exception("FoodPlan not found or deleted");
    }

    foodPlan.ConsumableId = dto.ConsumableId;
    foodPlan.Amount = dto.Amount;
    foodPlan.RequiresConfirmation = dto.RequiresConfirmation;
    foodPlan.IsRecurring = dto.IsRecurring;
    foodPlan.RunAtUtc = dto.IsRecurring ? null : dto.RunAtUtc;
    foodPlan.CronExpression = dto.IsRecurring ? dto.CronExpression : null;
    foodPlan.UserId = dto.UserId;
    foodPlan.ConsumableType = dto.ConsumableType;

    var jobKey = new JobKey(foodPlan.JobKey);
    var triggerKey = new TriggerKey(foodPlan.TriggerKey);

    var newJobDetail = JobBuilder.Create(dto.ConsumableType ==
ConsumableType.Food ? typeof(FoodRecordCreationJob) :
typeof(RecipeRecordCreationJob))
        .WithIdentity(jobKey)
        .UsingJobData(new JobDataMap
        {
            { "ObjectId", foodPlan.ConsumableId.ToString() },
            { "Amount", (float)foodPlan.Amount },
            { "RequiresConfirmation", foodPlan.RequiresConfirmation },
            { "ScheduleId", foodPlan.Id },
            { "ObjectType", foodPlan.ConsumableType }
        })
        .Build();

    await _scheduler.AddJob(newJobDetail, true);

    ITrigger newTrigger = foodPlan.IsRecurring
        ? TriggerBuilder.Create()
            .WithIdentity(triggerKey)

```

```

        .WithCronSchedule(foodPlan.CronExpression,
x.InTimeZone(TimeZoneInfo.Utc))
        .Build()
        : TriggerBuilder.Create()
        .WithIdentity(triggerKey)
        .StartAt(foodPlan.RunAtUtc.Value)
        .Build();

        await _scheduler.RescheduleJob(triggerKey, new Trigger);
        await _dbContext.SaveChangesAsync();
        return foodPlan;
    }
    public async Task DeleteScheduleAsync(long scheduleId)
    {
        var plan = await _dbContext.FoodPlans.FindAsync(scheduleId);
        if (plan == null || plan.IsDeleted)
            throw new Exception("FoodPlan not found or already deleted");

        var jobKey = new JobKey(plan.JobKey);
        var triggerKey = new TriggerKey(plan.TriggerKey);

        await _scheduler.PauseTrigger(triggerKey);
        await _scheduler.UnscheduleJob(triggerKey);
        await _scheduler.DeleteJob(jobKey);

        plan.IsDeleted = true;
        await _dbContext.SaveChangesAsync();
    }
}

```

Файл FoodRecordCreationJob.cs

Реалізація функціональної задачі автоматичного додавання записів споживання їжі

```

using FoodRecords.Api.Data;
using FoodRecords.Api.Models.Domain;
using MassTransit;
using Microsoft.EntityFrameworkCore;
using Quartz;
using System;

```

```

namespace FoodRecords.Api.Jobs

```

```

{
    public class FoodRecordCreationJob : IJob
    {
        private readonly FoodRecordsDbContext _dbContext;

        public FoodRecordCreationJob(FoodRecordsDbContext dbContext)
        {
            _dbContext = dbContext;
        }

        public async Task Execute(IJobExecutionContext context)
        {
            var objectId = (long)context.MergedJobDataMap["ObjectId"];
            var amount = (float)context.MergedJobDataMap["Amount"];
            var requiresConfirmation = (bool)context.MergedJobDataMap["RequiresConfirmation"];
            var scheduleId = (long)context.MergedJobDataMap["ScheduleId"];
            var objectType = (ConsumableType)context.MergedJobDataMap["ObjectType"];

            var job = await _dbContext.FoodPlans.Where(fp => !fp.IsDeleted && fp.Id == scheduleId).FirstOrDefaultAsync();
            if (requiresConfirmation)
            {
                var pendingAddition = new PendingRecord
                {
                    ConsumableType = ConsumableType.Food,
                    Amount = amount,
                    ConsumableId = objectId,
                    UserId = job.UserId,
                    DatePending = DateTime.UtcNow,
                    IsDeleted = false
                };

                _dbContext.PendingRecords.Add(pendingAddition);
                await _dbContext.SaveChangesAsync();
            }
            else
            {
                var foodRecord = new FoodRecord()
                {
                    DateEaten = DateTime.UtcNow,
                    UserId = job.UserId,
                    FoodMeasureId = objectId,

```

```
        Amount = amount,
        IsDeleted = false
    };

    _dbContext.FoodRecords.Add(foodRecord);
    await _dbContext.SaveChangesAsync();
}

if (!job.IsRecurring)
{
    job.IsDeleted = true;
    await context.Scheduler.DeleteJob(context.JobDetail.Key);
    await _dbContext.SaveChangesAsync();
}
}
}
```

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ ВІДСЛІДКОВУВАННЯ МАКРО- ТА

МІКРОНУТРІЄНТІВ У РАЦІОНІ

Програма та методика тестування

КП.ІП-1527.045440.04.51

“ПОГОДЖЕНО”

Керівник роботи:

_____ Олена МАРЧЕНКО

Консультант:

_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:

_____ Артем ХІЛЬЧУК

Київ – 2025

ЗМІСТ

1	ОБ'ЄКТ ВИПРОБУВАНЬ.....	3
2	МЕТА ТЕСТУВАННЯ.....	4
3	МЕТОДИ ТЕСТУВАННЯ.....	5
4	ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ.....	6

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Об'єктом випробування є вебзастосунок для відслідковування макро- та мікронутрієнтів у раціоні в браузері Chrome 119.0.6045.200(1920x1080).

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка збереження даних;
- перевірка сумісності веб-додатку з останніми версіями сучасних браузерів (Chrome, Opera, Firefox, ...);
- знаходження проблем, помилок і недоліків з метою їх усунення;
- перевірка зручності та інформативності графічного інтерфейсу.

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;
- динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;
- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- системне тестування – перевіряється усе програмне забезпечення в цілому;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування програмного забезпечення здійснюється у форматі мануального наскрізного (end-to-end) тестування з метою виявлення як функціональних помилок, так і потенційних недоліків у користувацькій взаємодії. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- динамічне тестування на відповідність функціональним вимогам;
- тестування на виведення повідомлень про помилку, коли це необхідно;
- тестування інтерфейсу користувача;
- тестування зручності використання;
- тестування валідації введених даних, де це є необхідним.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ВЕБЗАСТОСУНОК ДЛЯ ВІДСЛІДКУВАННЯ МАКРО- ТА
МІКРОНУТРІЄНТІВ У РАЦІОНІ**

Керівництво користувача

КП.ІП-1527. 045440.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена МАРЧЕНКО

Нормоконтроль:

_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:

_____ Артем ХІЛЬЧУК

Київ – 2025

ЗМІСТ

1	ПРИЗНАЧЕННЯ ПРОГРАМИ	3
2	ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ....	4
2.1	Системні вимоги для коректної роботи.....	4
2.2	Завантаження застосунку	4
2.3	Перевірка коректної роботи.....	4
3	ВИКОНАННЯ ПРОГРАМИ	5

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

NutriMatrix – це вебзастосунок призначений для відслідковування нутрієнтної характеристики спожитої їжі. Система надає можливості ведення та планування записів споживання, перегляду каталогу харчових продуктів, що розширена пошуковою підтримкою як за назвою, так і за штрих-кодом, створення та управління рецептами, налаштування відстежуваних нутрієнтів, автоматичного формування персоналізованого раціону на основі поставлених нутріційних цілей, а також перегляду звітності й статистики споживання за обрані періоди часу.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Вебзастосунок був призначений для використання у середовищі Chrome 119.0.6045.200. Для успішної роботи даного браузера необхідне виконання наступних вимог:

- наявність доступу до інтернету;
- ОЗП більше 4 ГБ;
- об'єм вільної пам'яті в понад 1 ГБ.

2.2 Завантаження застосунку

Доступ до вебзастосунку здійснюється шляхом переходу за гіперпосиланням на розгорнуту клієнтську частину.

2.3 Перевірка коректної роботи

За умови, що користувач перейшов за правильним гіперпосиланням, після завершення завантаження вебзастосунку відображається сторінка, наведена на рисунку 2.1.

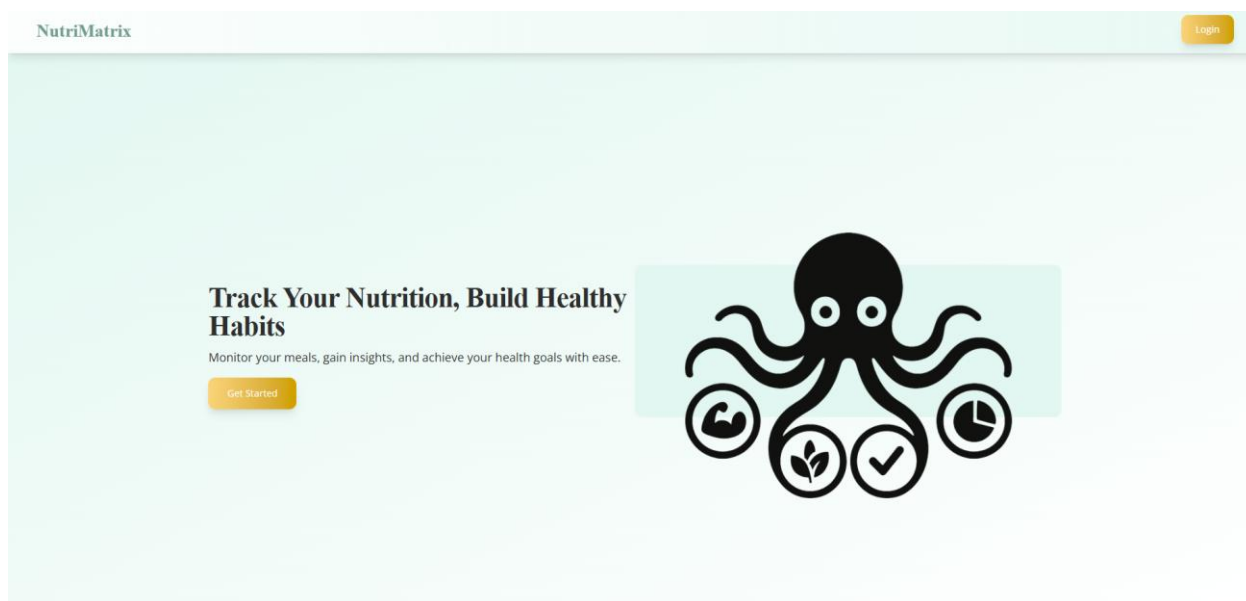


Рисунок 2.1 – Домашня сторінка вебзастосунку

3 ВИКОНАННЯ ПРОГРАМИ

При зверненні до домашньої сторінки застосунку користувачеві демонструється домашня сторінка, представлена на рисунку 3.1.

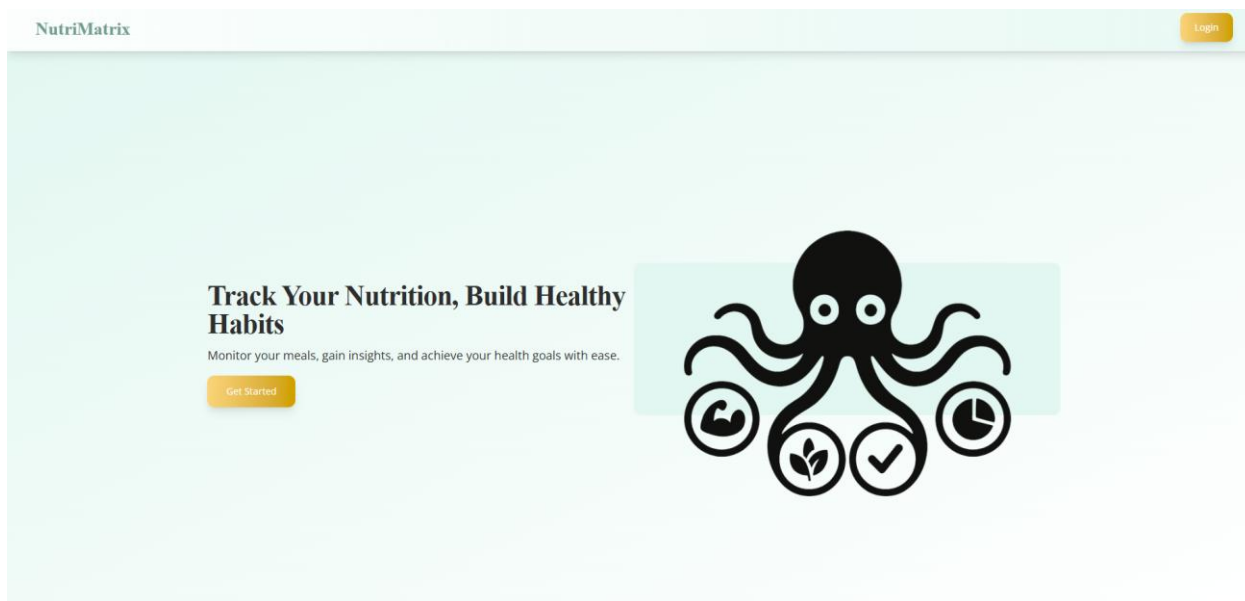


Рисунок 3.1 – Домашня сторінка вебзастосунку

Користувач може автентифікуватися, натиснувши кнопку «Login» й увівши необхідні облікові дані або обравши вхід через обліковий запис Google (рисунок 3.2); окрім того, із цієї сторінки передбачено перехід до форми реєстрації нового користувача (рисунок 3.3). Якщо користувач забув свій пароль, він може натиснути «Forgot password», після чого відбувається переадресація на сторінку відновлення паролю, де увівши пошту, асоційовану з обліковим записом, після чого на неї буде надіслано лист з подальшими інструкціями.

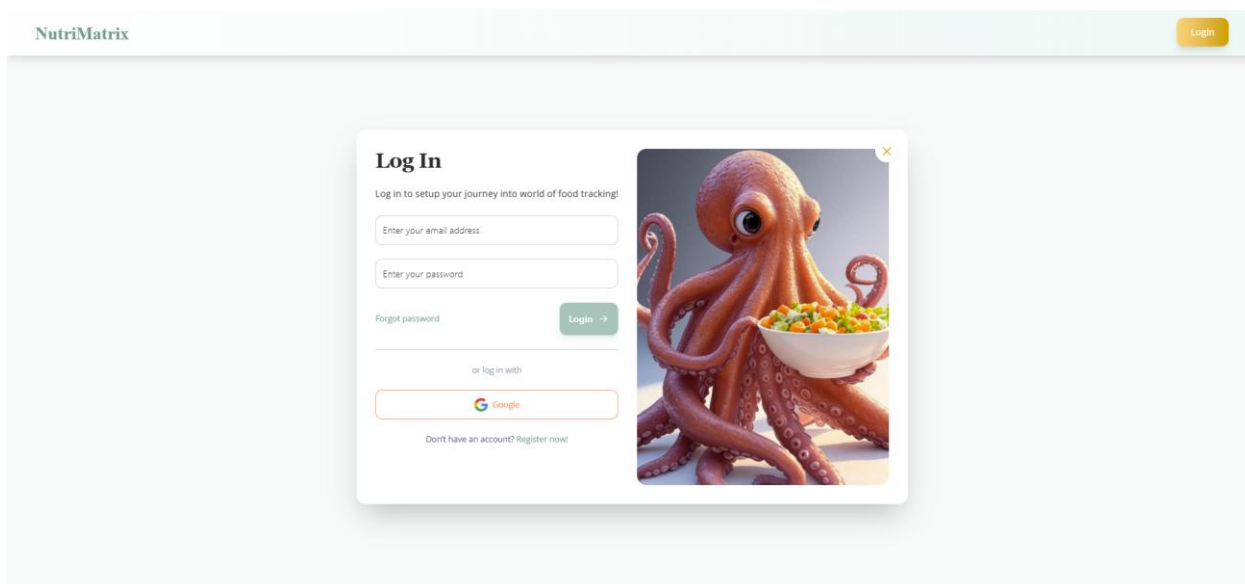


Рисунок 3.2 – Сторінка входження в обліковий запис

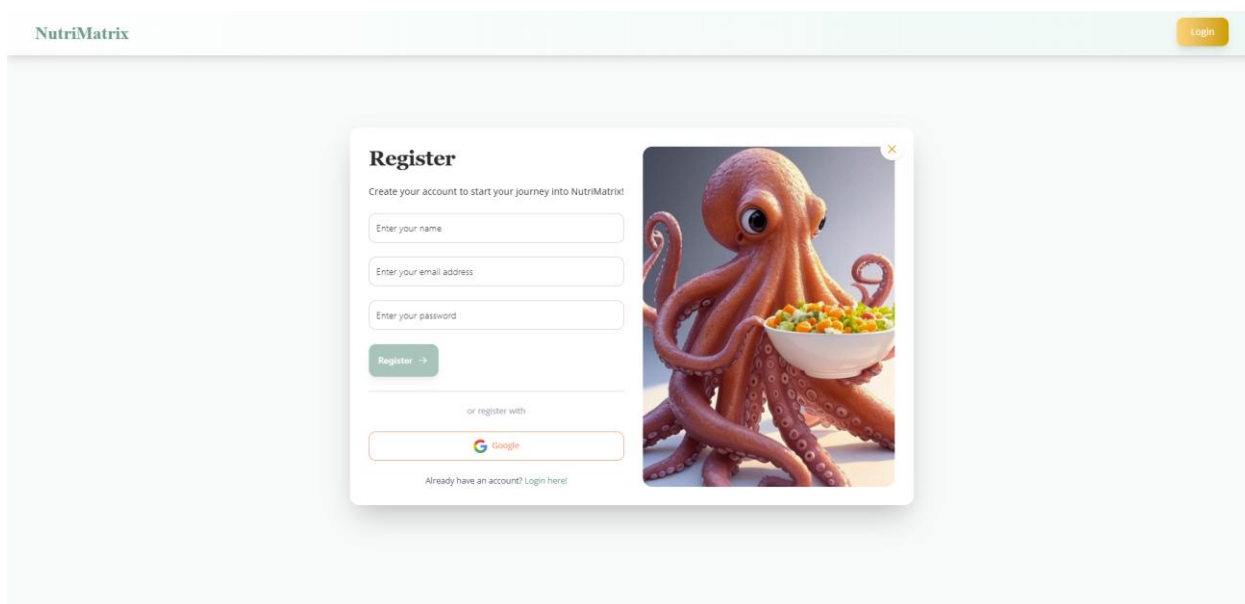


Рисунок 3.3 – Сторінка реєстрації

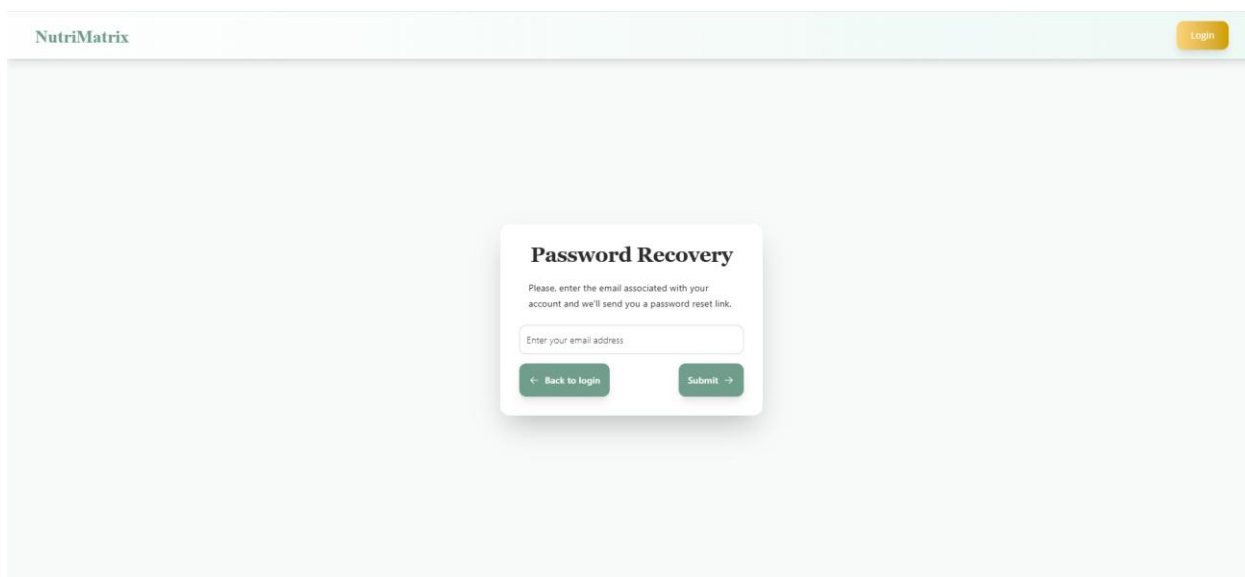


Рисунок 3.4 – Сторінка запиту скидання паролю

Після входження в обліковий запис користувач переадресовується на сторінку з харчовим щоденником додатку, а на збоку сторінки з'являється додаткова навігаційна панель (рисунок 3.5).

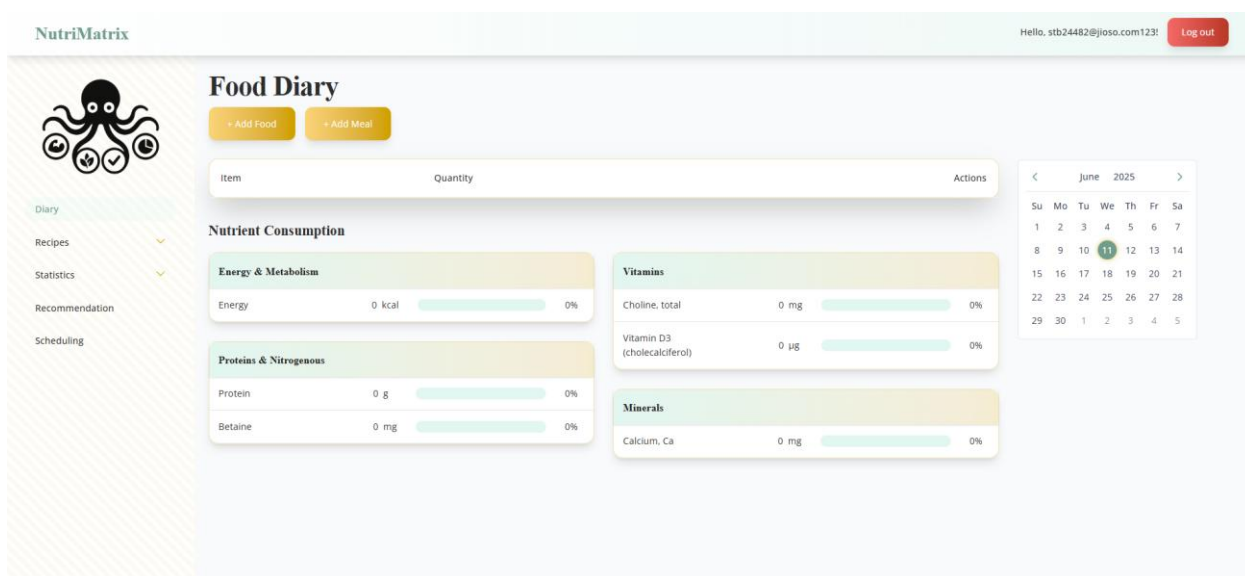


Рисунок 3.5 – Сторінка харчового щоденника

Користувач може додати новий запис споживання їжі, натиснувши на кнопку «Add Food Record», після чого відкривається модальне вікно вибору їжі; у даному вікні обирається необхідний продукт з можливістю його пошуку як за назвою через введення пошукового запиту, так і за штрих-кодом шляхом активації відповідної кнопки, після чого виконується введення одиниці виміру, кількості спожитого продукту (рисунок 3.6).

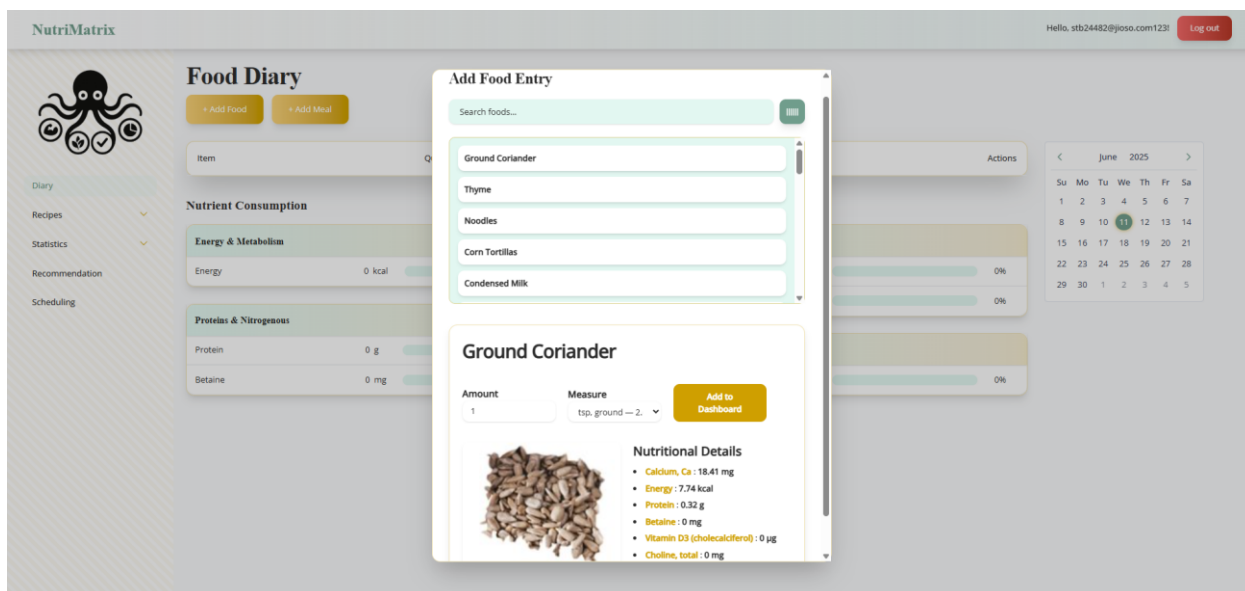


Рисунок 3.6 – Модальне вікно додавання запису споживання їжі

Аналогічним чином можна й додати запис споживання рецепту, для цього необхідно натиснути кнопку «Add Recipe Record», після чого відкривається модальне вікно вибору рецепту; у цьому вікні потрібно обрати необхідний рецепт з можливістю його пошуку як за назвою через введення пошукового запиту також ввести кількість спожитого рецепту в порціях (рисунок 3.7).

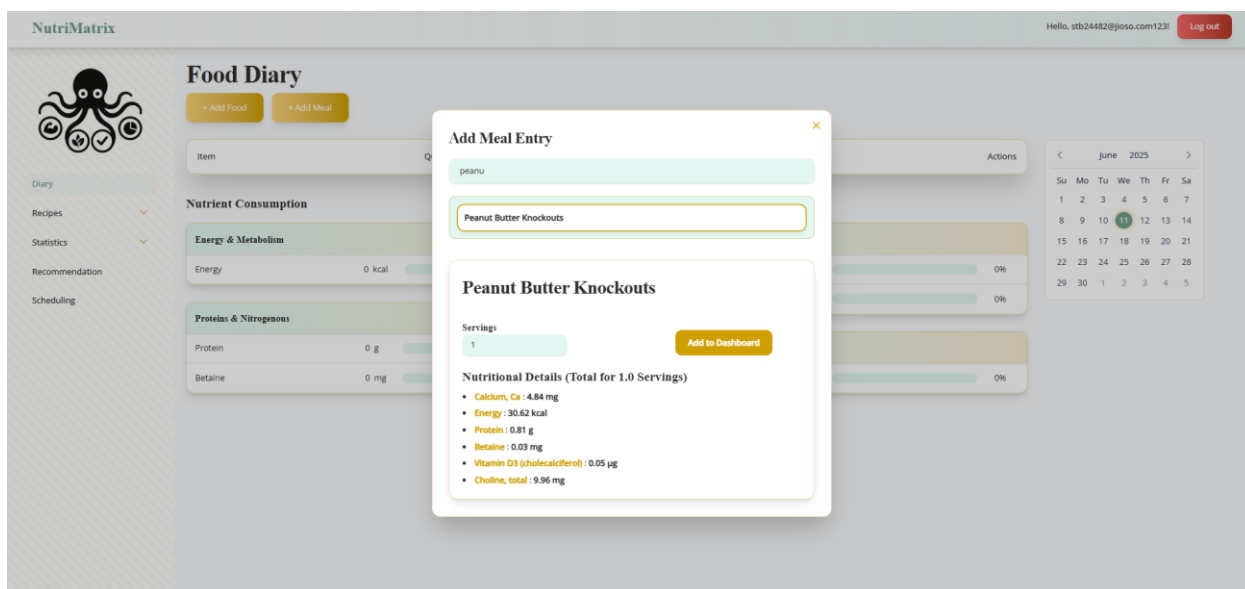


Рисунок 3.7 – Модальне вікно додавання запису споживання рецепту

Користувач може переглядати записи споживання за різні дати – для цього достатньо натискати на різні дати на календарі (рисунок 3.8).

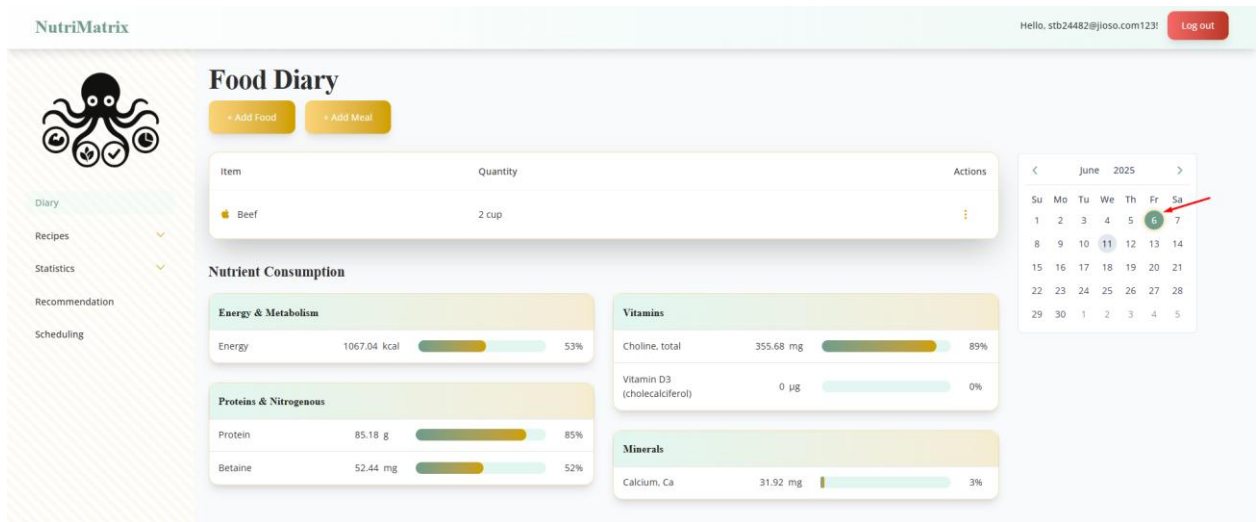


Рисунок 3.8 – Зміна обраної дати на календарі призводить до зміни відображуваних записів

Для зміни набору відслідковуваних нутрієнтів користувачу потрібно перейти на власний профіль (рисунок 3.9), натиснувши на своє ім'я у правому верхньому кутку, після чого потрібно натиснути на кнопку редагування профілю, обрати потрібні нутрієнти та натиснути зберегти (рисунок 3.10).

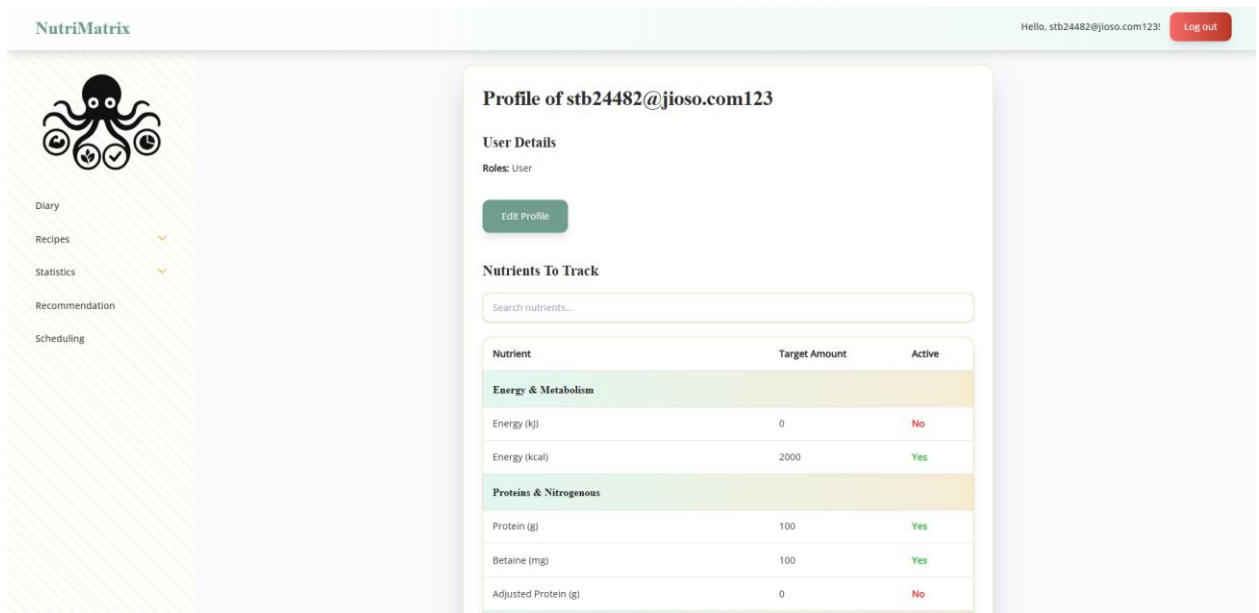
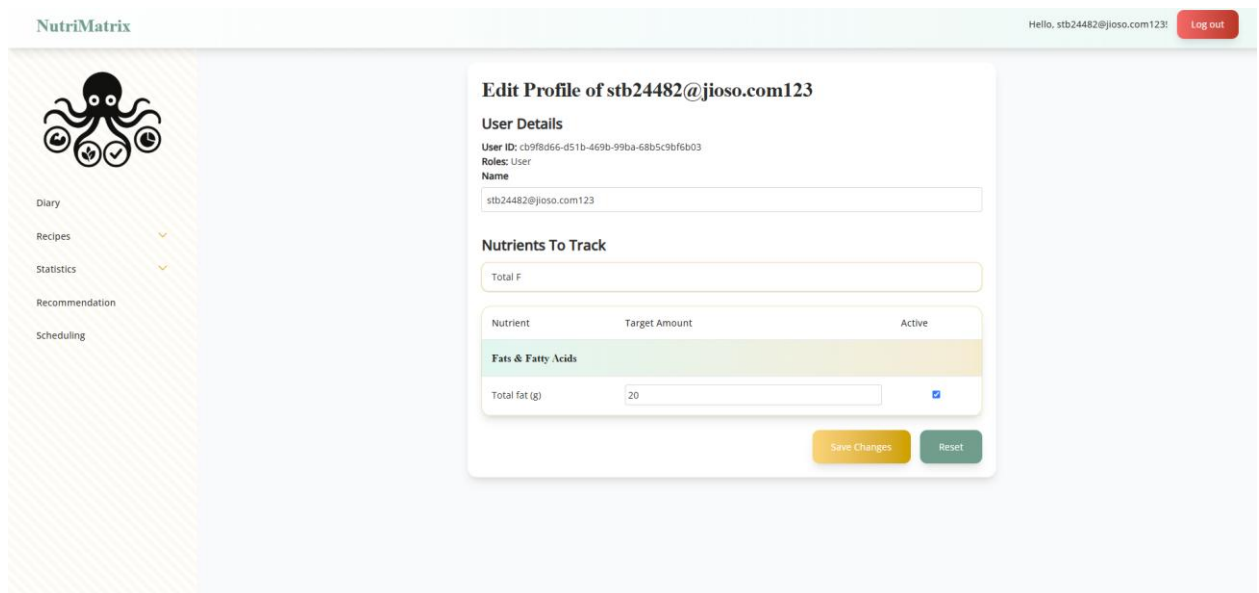


Рисунок 3.9 – Сторінка профілю користувача



NutriMatrix Hello, stb24482@jioso.com123! [Log out](#)

Edit Profile of stb24482@jioso.com123

User Details

User ID: cb9f8d66-d51b-469b-99ba-68b5c9bf6b03
 Roles: User
 Name:

Nutrients To Track

Total F:

Nutrient	Target Amount	Active
Fats & Fatty Acids		
Total fat (g)	20	☒

[Save Changes](#) [Reset](#)

Diary
Recipes
Statistics
Recommendation
Scheduling

Рисунок 3.10 – Сторінка редагування профілю користувача

Якщо користувач бажає отримати рекомендований раціон, йому необхідно натиснути на «Recommendation» на навігаційній панелі, після чого на сторінці отримання рекомендацій визначаються нутріційні цілі та по чергово натискається кнопка «Add Dish» для специфікації позиції страви та опціонально вводяться категорія, інгредієнти для включення та виключення (рисунок 3.11). Користувач натискає на кнопку отримання рекомендацій, після чого через деякий час знизу з'являється рекомендований раціон та його нутріційні характеристики (рисунок 3.12); якщо він задоволений ним, він може натиснути на кнопку для збереження рекомендації, у результаті чого наведені рецепти зберігаються в колекцію приватних користувача, а відповідні кількості додаються як записи споживання в щоденник харчування.

NutriMatrix Hello, stb24482@jioso.com123! Log out

Recipe Selector

Dish Slots

+ Add Dish Slot

Slot 1: < 15 Mins

Category: < 15 Mins

Included ingredients: whole egg

Excluded ingredients: whole milk

Slot 2: No category

Slot 3: No category

Slot 4: No category

Nutritional Goals

+ Add Nutrient Goal

Nutrient	Status	Value	Unit
Protein	X	70	g
Total fat	X	50	g
Carbohydrate, by difference	X	20	g
Vitamin K (phylloquinone)	X	40	µg
Choline, total	X	400	mg
Vitamin D3 (cholecalciferol)	X	15	µg

Find Recipes

Рисунок 3.11 – Визначення параметрів для рекомендації

NutriMatrix Hello, stb24482@jioso.com123! Log out

Recommended Recipes

Save Recipes and Log Meals

Brie En Croute - 1 servings

Round Steak and Gravy - 1 servings

miniature quiches - 7 servings

miniature quiches - 5 servings

Total Nutrients for All Recipes

Proteins & Nitrogenous

Protein 72.91 g 104%

Vitamins

Vitamin D3 (cholecalciferol) 10.51 µg 70%

Choline, total 487.03 mg 122%

Vitamin K (phylloquinone) 39.07 µg 98%

Carbohydrates & Sugars

Carbohydrate, by difference 22.44 g 112%

Fats & Fatty Acids

Total fat 39.13 g 78%

Рисунок 3.12 – Результуюча рекомендація

Для перегляду переліку власних рецептів необхідно перейти на вкладку «My Recipes», де за необхідності можна скористатися можливістю пошуку (рисунок 3.13); для перегляду ж конкретного рецепту можна натиснути на кнопку «View More», після чого відбудеться переадресація на сторінку перегляду деталей рецепта (рисунок 3.14).

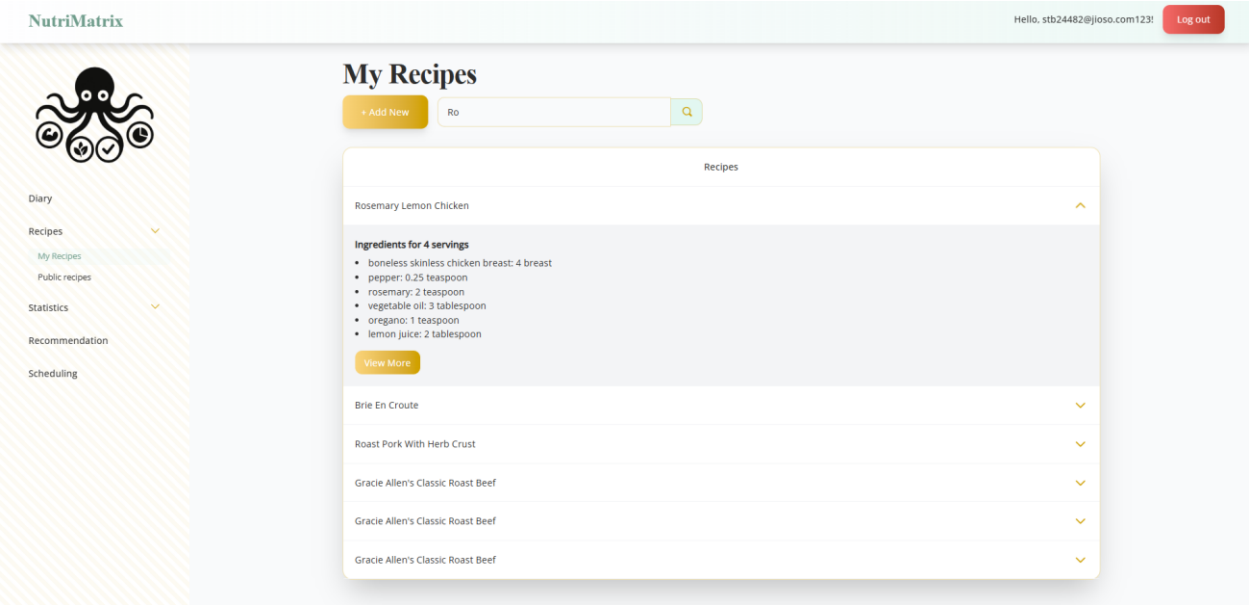


Рисунок 3.13 – Сторінка колекції власних рецептів

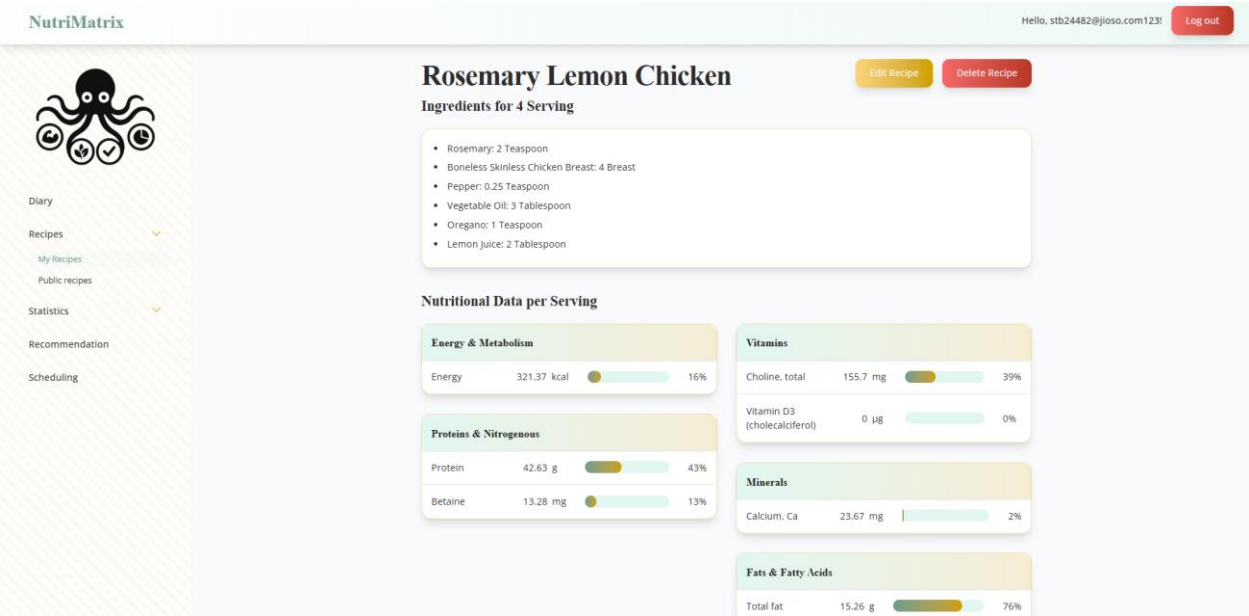


Рисунок 3.14 – Сторінка власного рецепту

На сторінці перегляду рецепту користувач може або видалити рецепт, натиснувши на «Delete Recipe», або ініціювати редагування – для цього необхідно натиснути на кнопку «Edit Recipe», після чого відбудеться переадресація на сторінку редагування рецепту (рисунок 3.15). Після цього по чергово вводяться нова назва рецепту, перелік інгредієнтів та загальна кількість порцій, а врешті – натискається кнопка збереження.

Рисунок 3.15 – Сторінка редагування власного рецепту

Для створення власного рецепту, необхідно на сторінці перегляду всіх рецептів натиснути на кнопку «Add New Recipe», після чого відбувається переадресація на сторінку створення рецепту(рисунок 3.16). На ній необхідно задати ім'я рецепту, кількість порцій та перелік інгредієнтів, їх мір та кількостей, що дадуть таку кількість порцій та врешті-решт натиснути кнопку «Submit Recipe» для збереження.

Energy & Metabolism		
Energy	102.51 kcal	5%

Vitamins		
Choline, total	0.92 mg	0%
Vitamin D3 (cholecalciferol)	0 µg	0%

Рисунок 3.16 – Сторінка додавання власного рецепту

Для перегляду ж публічних рецептів необхідно перейти на вкладку “Public Recipes”, де за необхідності можна скористатися можливістю пошуку за назвою або фільтрами за інгредієнтами(рисунок 3.17); для перегляду ж

конкретного рецепту можна натиснути на кнопку «View More», після чого відбудеться переадресація на сторінку перегляду деталей рецепта(рисунок 3.18). На даній сторінці, за бажання, можна скопіювати рецепт до своєї колекції – для цього потрібно натиснути на кнопку «Save To My Recipes».

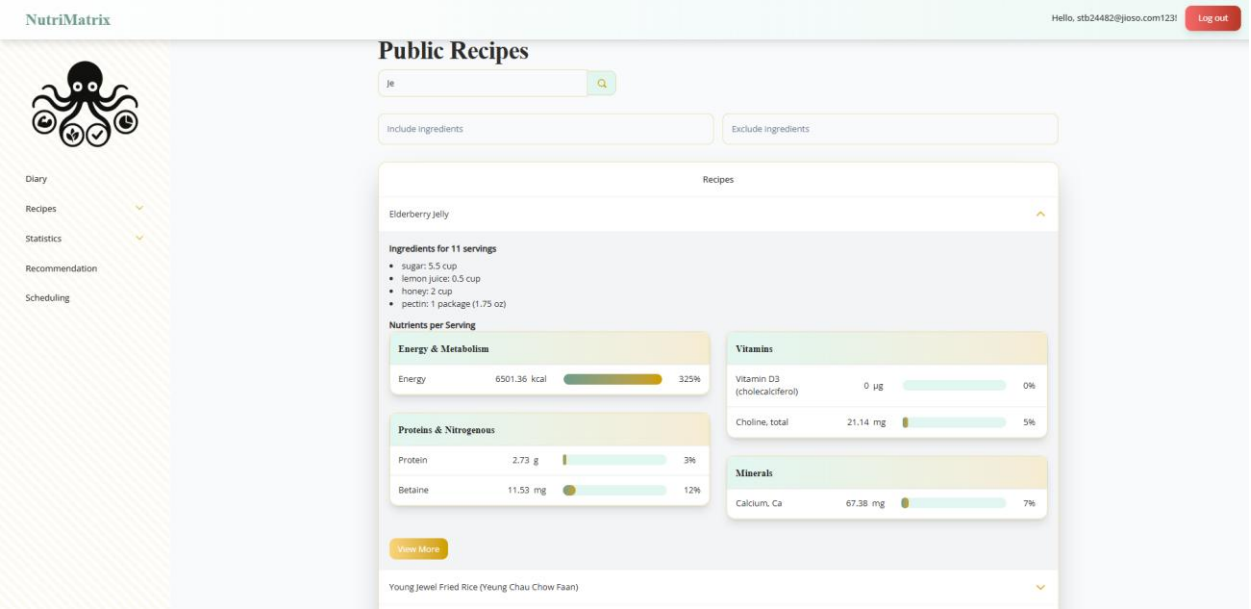


Рисунок 3.17 – Сторінка колекції публічних рецептів



Рисунок 3.18 – Сторінка публічного рецепту

Для перегляду власних планувальників записів споживання користувачу потрібно натиснути на «Scheduling» на навігаційній панелі, після чого його буде переадресовано на сторінку з переліком активних планів (рисунок 3.19). Для видалення плану, достатньо буде натиснути кнопку «Delete» для відповідного запису.

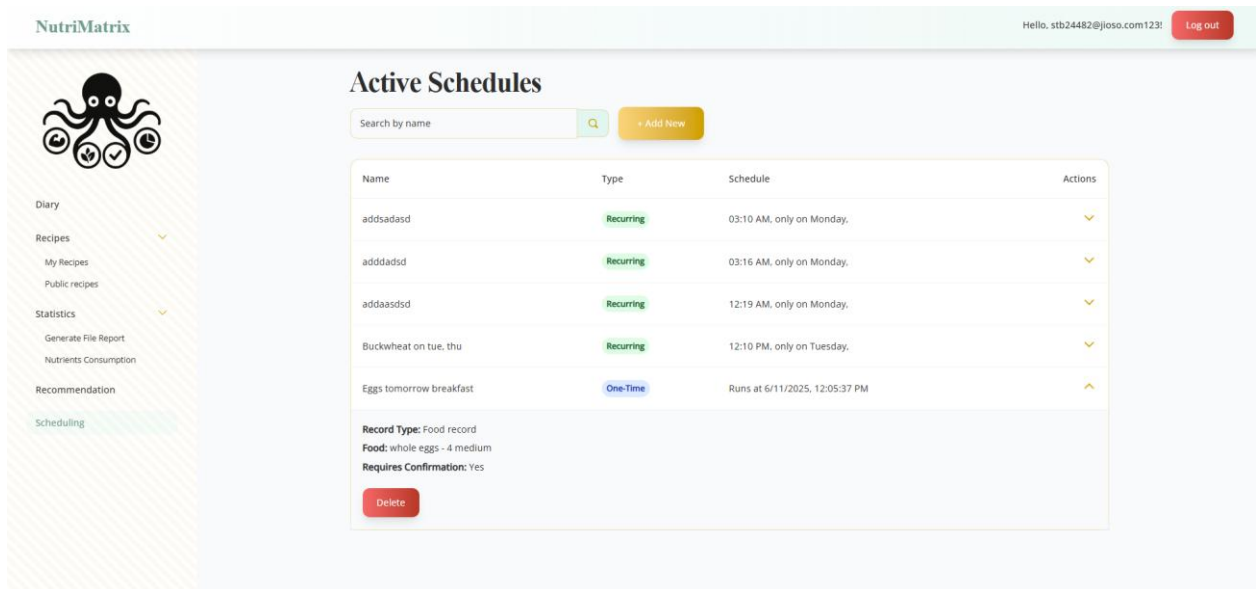


Рисунок 3.19 – Сторінка перегляду планувальників користувача

Для створення нового планувальника необхідно натиснути кнопку «Add New», після чого відбувається переадресація на сторінку створення нового планувальника (рисунок 3.20). Необхідно вказати його назву, тип споживання (їжа чи рецепт), встановити конкретний об'єкт споживання та його спожиту кількість, а також специфікувати чи потребує запис підтвердження перед доданням та чи повторюваний планувальник, чи ні; залежно від його типу необхідно увести часову складову планувальника: якщо він одноразовий, потрібно лише вказати дату й час виконання, а якщо він є повторюваний, потрібно задати час та дні тижня, коли йому слід виконуватися.

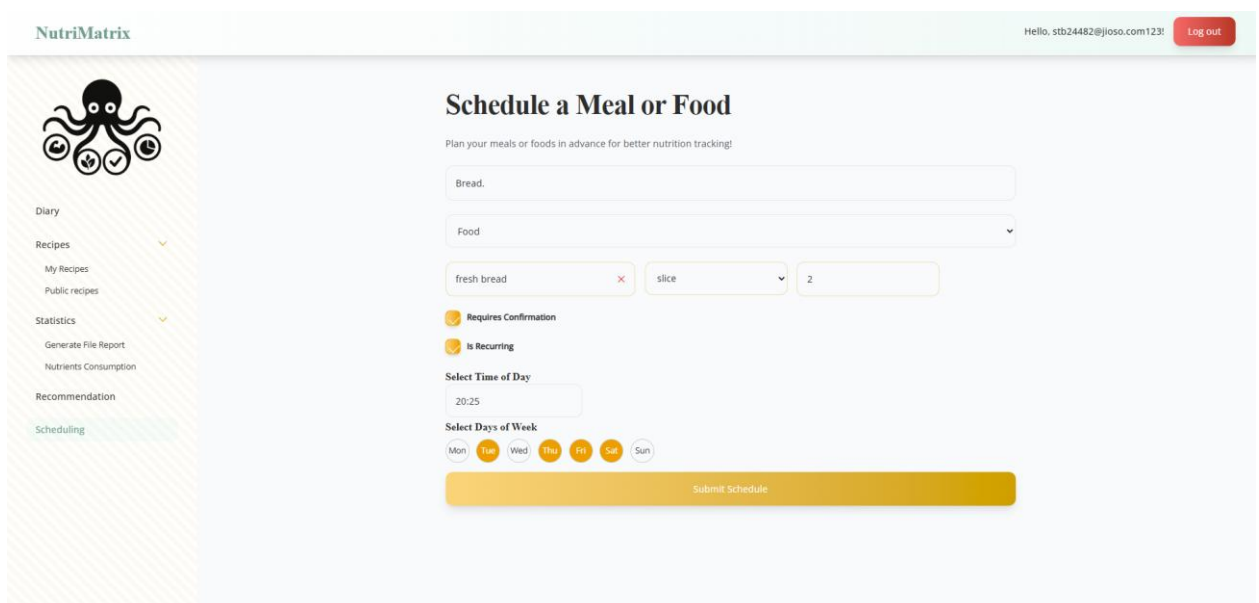


Рисунок 3.20– Сторінка створення планувальника

Перегляд статистики передбачає натискання на кнопку «Nutrients Consumption», у результаті чого відбувається переадресація на сторінку перегляду статистики споживання нутрієнтів (рисунок 3.21). На ній можна вказати часовий проміжок, за яким буде збиратися статистика, після чого буде виведено відповідну інфографіку та таблиці з інформацією щодо споживання відслідковуваних нутрієнтів.

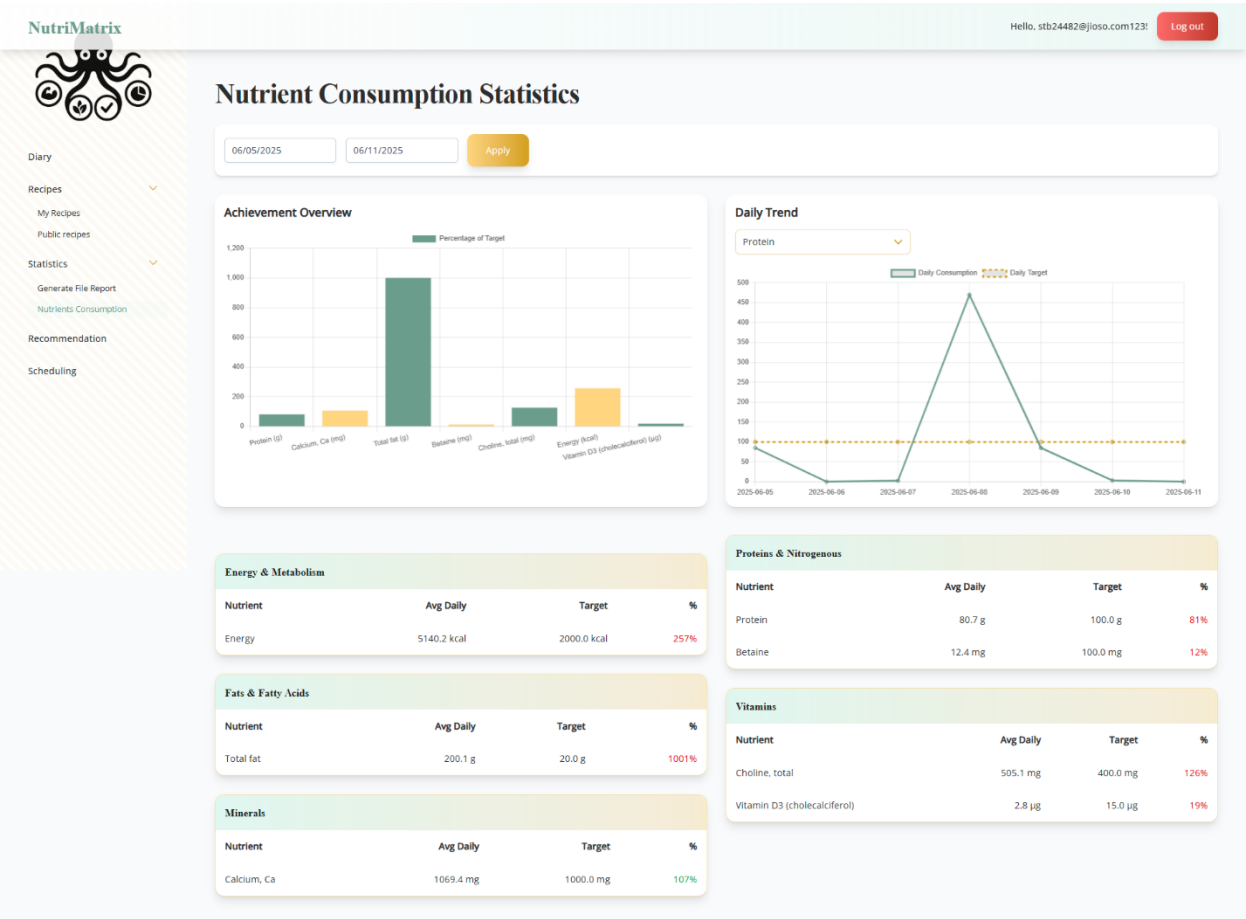


Рисунок 3.21 – Сторінка перегляду статистики споживання нутрієнтів

Для формування ж файлового звіту споживання потрібно натиснути на вкладку «Generate File Report» на навігаційній панелі, після чого на відкритій сторінці потрібно вказати часовий проміжок, а також формат вихідного файлу (рисунок 3.22). Після натискання на кнопку «Generate Report» відбувається завантаження файлу-звіту.

NutriMatrix Hello, stb24482@jioso.com123! [Log out](#)



- Diary
- Recipes
 - My Recipes
 - Public recipes
- Statistics
 - Generate File Report
 - Nutrients Consumption
- Recommendation
- Scheduling

Generate Consumption Report

Start Date
2025-06-01

End Date
2025-06-11

File Format
CSV

Generate Report

Рисунок 3.22 – Сторінка формування файлового звіту споживання

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ВЕБЗАСТОСУНОК ДЛЯ ВІДСЛІДКУВАННЯ МАКРО- ТА
МІКРОНУТРІЄНТІВ У РАЦІОНІ**

Графічний матеріал

КПІ.ПІ-1527.045440.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олена МАРЧЕНКО

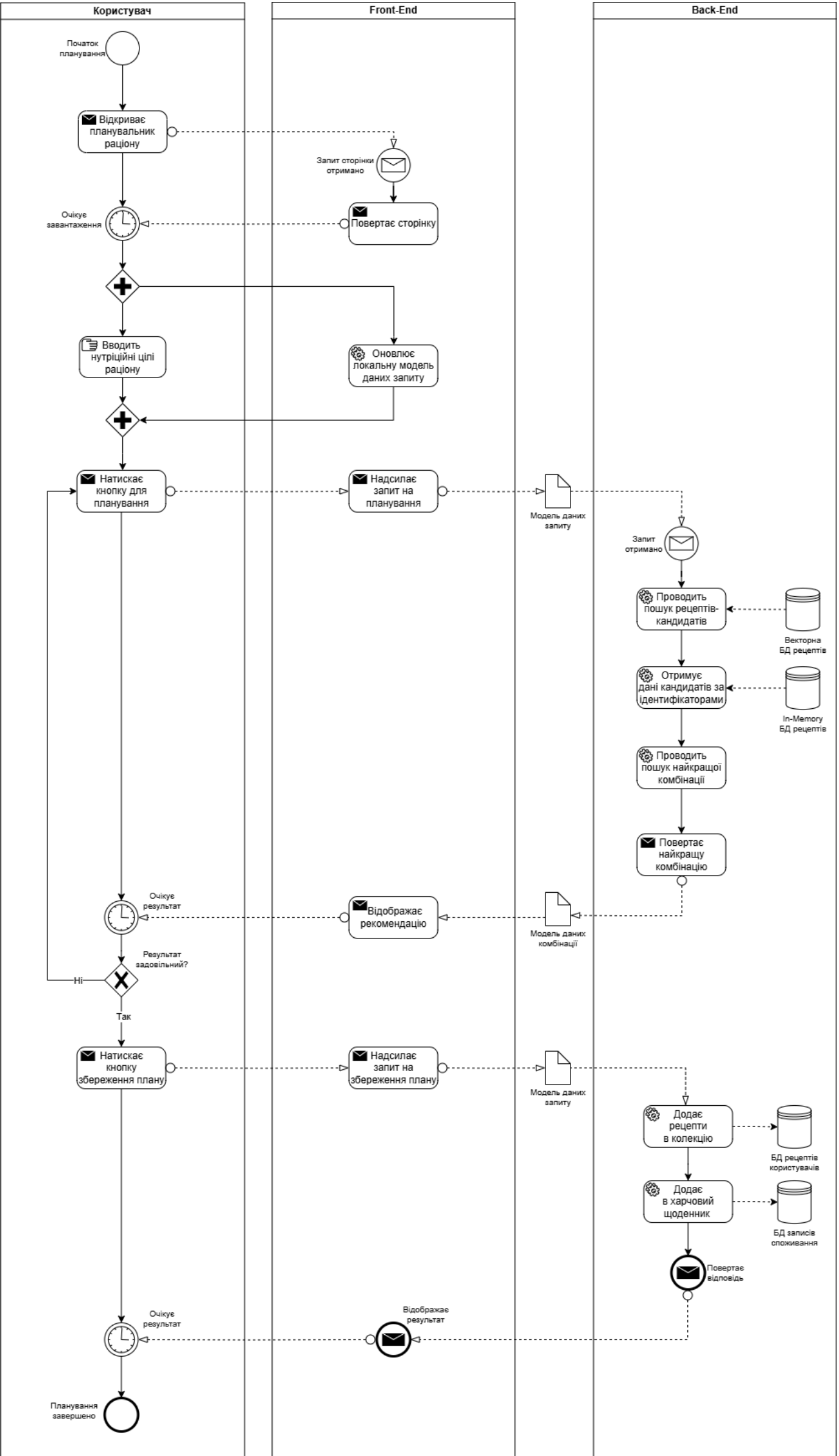
Нормоконтроль:

_____ Тетяна ШУЛЬКЕВИЧ

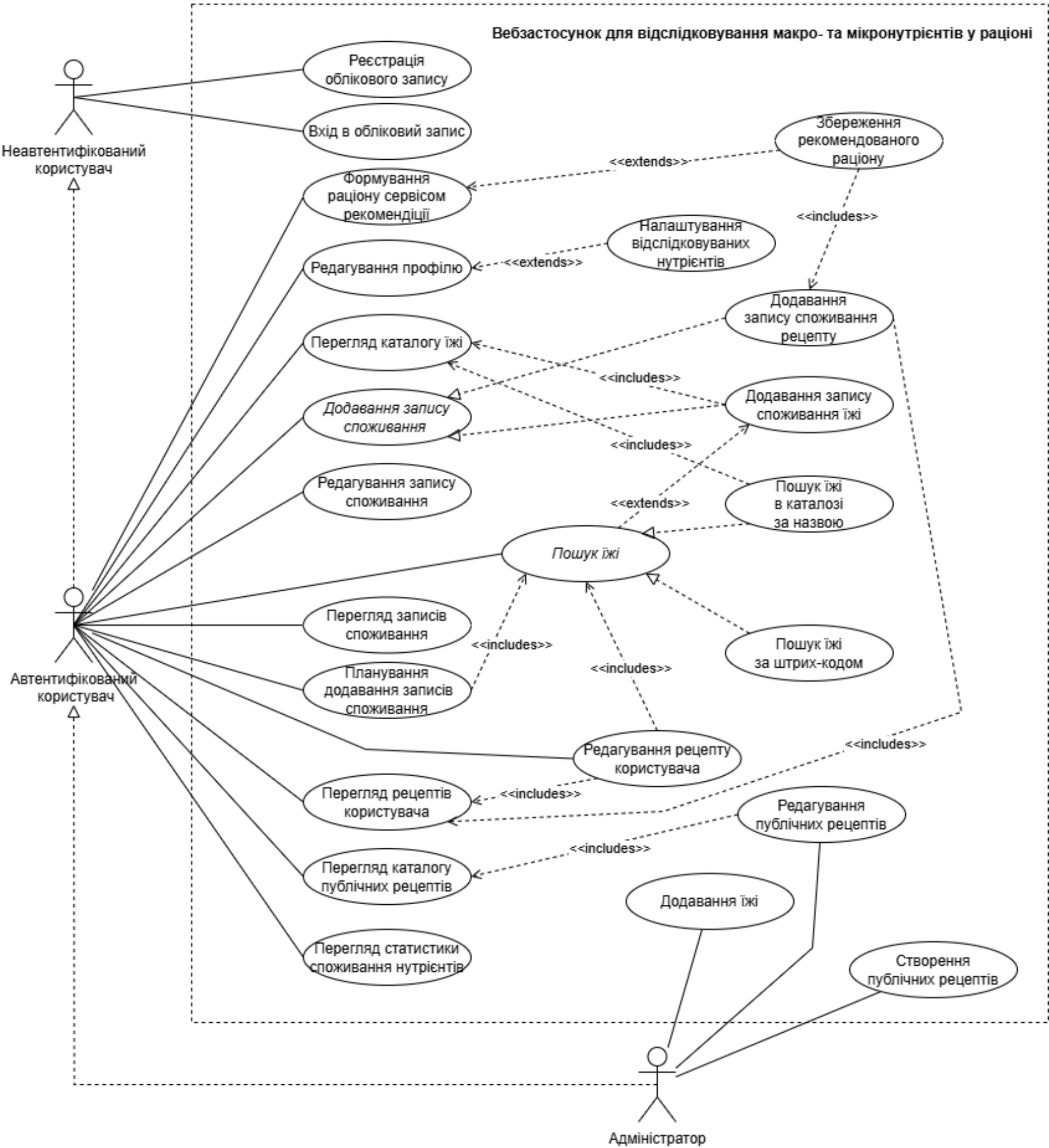
Виконавець:

_____ Артем ХІЛЬЧУК

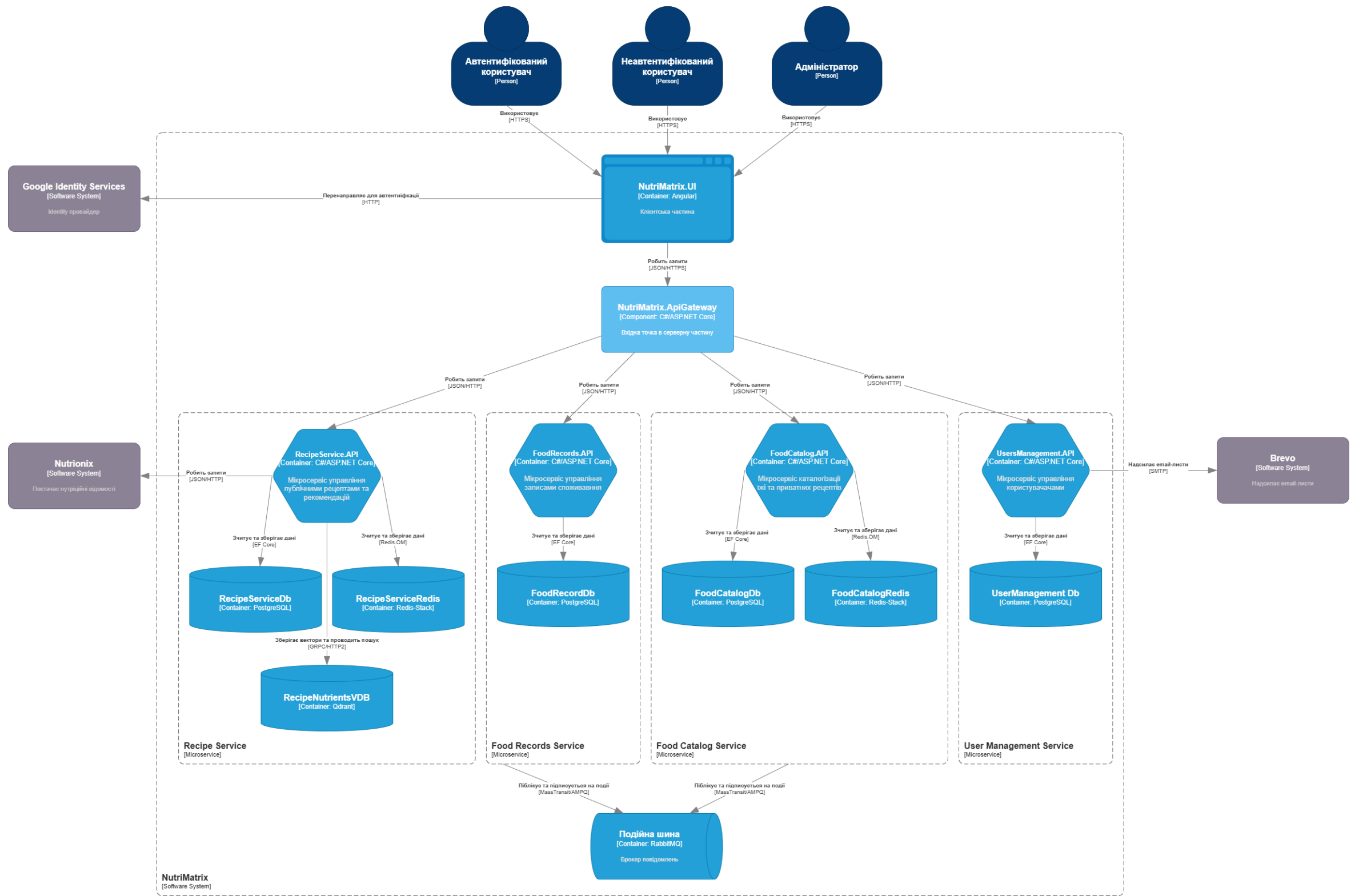
Київ – 2025



					КПІ.ІП-1527.044540.06.99.ССД						
					Схема структурна діяльності	Лит.			Маса	Масштаб	
Зм.	Арк.	№ докум.	Підп.	Дата							
Розробив		Хільчук А.В.									
Перевірів		Марченко О.І.									
Т. контр.						Аркуш 1			Аркушів 1		
					Вебзастосунок для відслідковування макро- та мікронутрієнтів у раціоні	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-15					
Н. контр.		Шулькевич Т.В.									
Затвердив		Жаріков Е.В.									



					КПІ.ІП-1527.044540.06.99.CCB							
					Схема структурна варіантів використання	Лит.		Маса		Масштаб		
Зм.	Арк.	№ докум.	Підп.	Дата								
Розробив		Хільчук А.В.										
Перевірів		Марченко О.І.										
Т. контр.						Аркуш 1		Аркушів 1				
Н. контр.		Шулькевич Т.В.			Вебзастосунок для відслідковування макро- та мікронутрієнтів у раціоні					КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-15		
Затвердив		Жаріков Е.В.										



					КПІ.ІП-1527.044540.06.99.CCM							
					Схема структурна компонентів програмного забезпечення	Літера			Маса		Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата								
Розробив		Хільчук А.В.										
Перевішив		Марченко О.І.										
Т. контр.						Аркуш 1			Аркушів 1			
					Вебзастосунок для відслідковування макро- та мікронутрієнтів у раціоні	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-15						
Н. контр.		Шулькевич Т.В.										
Затвердив		Жаріков Е.В.										