ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка web-застосунку "Щоденник дієти"»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Максим ВЕРТІЙ
(підпис)

Виконав: здобувач вищої освіти групи ПД-43

Максим ВЕРТІЙ

Керівник: Світлана ШЕВЧЕНКО
к.п.н., доцент

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

_____ Вертію Максиму Олександровичу _____

1. Тема кваліфікаційної роботи: «Розробка web-застосунку "Щоденник дієти"» керівник кваліфікаційної роботи к.п.н., доцент Світлана ШЕВЧЕНКО, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості та програмні реалізації для контролю дієтичного харчування; технічна документація з описом Node.js та Express.

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

1. Провести аналіз сучасних вимог до веб-застосунків.
2. Проаналізувати існуючи щоденники дієти і визначити їхні переваги та недоліки.
3. Сформувати функціональні та нефункціональні вимоги до клієнт-серверного застосунку «Щоденник дієти».
4. Дослідити інструменти та технології для розробки клієнт-серверного застосунку «Щоденник дієти».

5. Розробити клієнт-серверний застосунок на основі обраних технологій і з урахуванням зазначених вимог для «Щоденника дієти».
6. Провести тестування клієнт-серверного застосунку «Щоденник дієти».

5. Перелік графічного матеріалу: *презентація*
   1. Аналіз аналогів.
   2. Вимоги до програмного забезпечення.
   3. Програмні засоби реалізації.
   4. Діаграма варіантів використання.
   5. Алгоритм роботи застосунку.
   6. Діаграма класів.
   7. Екранні форми.
   8. Апробація результатів дослідження

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02-16.03.2025	
2	Провести аналіз сучасних вимог до веб-застосунків.	17.03-23.03.2025	
3	Проаналізувати існуючи щоденники дієти і визначити їхні переваги та недоліки.	24.03-31.03.2025	
4	Сформувати функціональні та нефункціональні вимоги до клієнт-серверного застосунку «Щоденник дієти».	1.04-8.04.2025	
5	Дослідити інструменти та технології для розробки клієнт-серверного застосунку «Щоденник дієти».	9.04-15.04.2025	
6	Розробити web-застосунок на основі обраних технологій і з урахуванням зазначених вимог для «Щоденника дієти».	16.04-24.04.2025	
7	Провести тестування клієнт-серверного застосунку «Щоденник дієти».	24.04-28.04.2025	
8	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
9	Розробка демонстраційних матеріалів	12.05-18.05.2025	
10	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Максим ВЕРТІЙ
(підпис)

Керівник
кваліфікаційної роботи _____ Світлана ШЕВЧЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 53 стор., 2 табл., 35 рис., 20 джерел.

Мета роботи – допомога користувачам відстежувати своє харчування та прогрес у досягненні дієтичних цілей.

Об'єкт дослідження – процес контролю дієтичного харчування.

Предмет дослідження – web-застосунок «Щоденник дієти».

Короткий зміст роботи: Дане дослідження присвячене розробці web-застосунку щоденник дієти для запису та розрахунку калорій. У роботі здійснено аналіз існуючих аналогів додатків, таких як Tablycjakalorijnosti, MyFitnessPal, Lose It!, FatSecret. Було визначено їхні сильні та слабкі сторони, що дозволило сформулювати вимоги до створення нового додатка та його проєктування.

Розроблено алгоритм роботи застосунку та програмно реалізовані ключові функціональні можливості, зокрема: додавати продукти, рецепти і напої, розрахунок щоденної норми калорій, розрахунок індексу маси тіла, пошуку продукту для зареєстрованих користувачів та рецепту для незареєстрованих користувачів, використання AI помічника для особистих питань та відображення всіх продуктів у web-застосунку, а також продуктів для діабетиків. У роботі використано Node.js для back-end, технології HTML, CSS та JavaScript для front-end, MongoDB для бази даних.

Користуватися застосунком може будь-яка людина, котрій важливо відстежувати своє харчування та прогрес у досягненні дієтичних цілей. Здебільшого має попит для спортсменів, людей похилого віку та людям з особливою дієтою, де важливо рахувати спожиті калорії.

КЛЮЧОВІ СЛОВА: ЩОДЕННИК ДІЄТИ, WEB-ЗАСТОСУНОК, NODE.JS, MONGODB, JAVASCRIPT, КОРИСТУВАЧІ.

ЗМІСТ

ВСТУП

У наш час, багато людей дбають про своє здоров'я та фізичну форму, важливою частиною успішного схуднення та підтримки здорового способу життя є правильне харчування. Щоденник дієти стає ефективним інструментом для контролю калорій, які споживає людина протягом дня. Однак багато існуючих додатків для контролю дієти не зручні та застарі [1].

Тому розробка спеціалізованого веб-додатку для ведення щоденника дієти з інтеграцією AI помічника є надзвичайно актуальним завданням, яке дозволяє користувачам не лише зручно фіксувати свої калорії та аналізувати харчові звички, а й отримувати персоналізовані рекомендації для досягнення своїх цілей. Вбудований штучний інтелект є гарною додатковою функцією, яка доповнює основні можливості програми [2-5].

Завдяки AI помічнику, користувачі зможуть отримувати персоналізовані поради, зважати на потреби свого організму і коригувати раціон у реальному часі.

Враховуючи високий рівень конкуренції на ринку додатків для здоров'я, розробка функціонального та інтуїтивно зрозумілого інтерфейсу з можливістю використання AI помічника стане ключем до залучення користувачів. Такий додаток забезпечить користувачам зручне та ефективне планування харчування, допомагаючи досягати своїх цілей за допомогою інтелектуальних рекомендацій і зберігання даних для подальшого аналізу.

Мета роботи: допомога користувачам відстежувати своє харчування та прогрес у досягненні дієтичних цілей.

Об'єкт дослідження: процес контролю дієтичного харчування.

Предмет дослідження: web-застосунок «Щоденник дієти».

Для досягнення мети в роботі необхідно вирішити такі завдання:

1. Провести аналіз сучасних вимог до web-застосунків.
2. Проаналізувати існуючи щоденники дієти і визначити їхні переваги та недоліки.

3. Сформувати функціональні та нефункціональні вимоги до клієнт-серверного застосунку «Щоденник дієти».

4. Дослідити інструменти та технології для розробки клієнт-серверного застосунку «Щоденник дієти».

5. Розробити web-застосунок на основі обраних технологій і з урахуванням зазначених вимог для «Щоденника дієти».

6. Провести тестування клієнт-серверного застосунку «Щоденник дієти».

Методи дослідження. Для досягнення поставлених завдань у роботі використовуються наступні методи: системно-структурний аналіз, порівняльний метод, а також методи передачі, зберігання і обробки даних. Для розробки застосунку застосовані сучасні технології: серверна частина створена з використанням бібліотеки Express, для забезпечення безпеки паролів використано алгоритм Bcrypt, а взаємодія з базою даних забезпечена через Mongoose та MongoDB, для розробки інтерфейсу користувача HTML, CSS, JavaScript.

Наукова новизна полягає в створенні web-застосунку, який дозволить користувачам записувати дієтичне харчування, моніторити свої досягнення в поставлених цілях.

1 АНАЛІЗ WEB-ЗАСТОСУНКА «ЩОДЕННИК ДІЄТИ»

1.1 Аналіз та загальна характеристика веб-додатка для контролю дієтичного харчування.

Веб-додаток — це програмний продукт, який доступний через мережу Інтернет, він надає користувачам певні функції або послуги через веб-браузери. Замість того, щоб встановлювати програму на локальний комп'ютер, веб-застосунки працюють безпосередньо на сервері, де зберігаються всі необхідні дані та алгоритми. Користувачі можуть отримувати доступ до такого застосунку в будь-який час і з будь-якого пристрою, який має підключення до Інтернету.

Веб-застосунки можуть бути різних типів і спрямовані на виконання різних завдань, таких як управління бізнес-процесами, соціальні мережі, онлайн-магазини, ведення записів або облік, а також спеціалізовані додатки, такі як щоденники дієт, фітнес-трекери тощо. Вони забезпечують інтерактивну взаємодію з користувачем, дозволяючи здійснювати налаштування, вводити та обробляти дані, а також отримувати результати або рекомендації.

Web-додаток для контролю дієтичного харчування — це спеціалізований інструмент, що дозволяє користувачам автоматично фіксувати та аналізувати кількість калорій, які вони споживають протягом дня. Такий додаток може допомогти користувачам контролювати своє харчування, встановлювати цілі щодо калорійності раціону та отримувати персоналізовані рекомендації щодо здорового харчування. Web-додатки мають ряд переваг перед мобільними, оскільки дозволяють зберігати дані в хмарі та забезпечують доступ до них з будь-якого пристрою з підключенням до Інтернету.

Для розгортання web-застосунку на сервері необхідно налаштувати серверну частину, де розміщуватиметься сам додаток. Це включає в себе встановлення та налаштування серверної інфраструктури, бази даних для зберігання інформації та налаштування безпечного доступу до застосунку. Після розгортання web-

застосунку на сервері, він стає доступним для користувачів через веб-браузери, що забезпечує ефективне використання додатка без необхідності встановлення на локальні пристрої.

1.2 Цільова аудиторія

Цільова аудиторія web-застосунку «Щоденник дієти» включає кілька категорій користувачів, які мають цікавість до контролю за своїм харчуванням і здоров'ям. Однією з основних груп є люди, що прагнуть до здорового способу життя, які бажають підтримувати фізичну форму, стежити за своїм харчуванням та контролювати споживані калорії з метою досягнення оптимальної ваги та здоров’я. Для цієї категорії користувачів веб-застосунок є інструментом для більш детального аналізу спожитих продуктів та кількості калорій, що допомагає їм підтримувати здоровий баланс.

Інша категорія - це люди, які активно займаються спортом, фітнесом або працюють над збільшенням м'язової маси чи зниженням ваги. Веб-застосунок надає можливість таким користувачам зручно і точно підраховувати кількість спожитих та втрачених калорій під час тренувань, забезпечуючи ефективне підтримання балансу між раціоном і фізичною активністю.

Окрему групу складають особи, які мають проблеми з вагою та прагнуть зменшити або збільшити свою масу тіла через коригування харчування. Для таких користувачів додаток є інструментом для контролю за калорійністю їжі та створення персоналізованих планів харчування з урахуванням їхніх індивідуальних цілей.

Веб-застосунок також орієнтований на дієтологів та нутриціологів, які працюють із клієнтами, допомагаючи їм досягти здорового способу життя через правильне харчування. Застосунок може бути корисним інструментом для аналізу раціону клієнтів, надання рекомендацій та моніторингу їхнього прогресу.

А також, додаток може бути корисним для людей, які мають певні медичні обмеження, такі як діабет, проблеми з шлунково-кишковим трактом, серцево-

судинні захворювання, і потребують суворого контролю харчування. Веб-застосунок дозволяє таким користувачам стежити за споживаними калоріями, складом їжі та іншими параметрами, що сприяють стабільному стану здоров'я. А в додатку Slim-Body ще й можливо отримати продукти з найменшою кількістю вуглеводів для діабетиків натиснувши на кнопку в AI помічнику.

1.3 Дослідження існуючих аналогів

Для розробки web-застосунку «Щоденник дієти» (Slim-Body) було проаналізовано існуючі програмні рішення, які використовуються для:

- запису харчування;
- розрахунку денної норми калорій;
- відстеження зміни ваги;
- розрахунку індексу маси тіла.

Веб-застосунок повинен дозволяти користувачам вводити дані про своє харчування, зокрема про продукти, що були спожиті протягом дня. Користувачі можуть додавати їжу вручну або вибирати її зі списку запропонованих продуктів, що містять дані про калорії, білки, жири, вуглеводи та інші поживні елементи. Щоб забезпечити точність, застосунок також може включати функцію пошуку продуктів за їх назвою. Усі ці дані зберігаються у персоналізованому щоденнику, де можна легко переглядати список спожитих продуктів і аналізувати їхній внесок у загальний раціон.

На сьогоднішній день одними з найпопулярніших веб-сайтів для відстежування свого харчування є Таблиця Калорійності, MyFitnessPal, Lose It!, FatSecret. Отже розглянемо їх детальніше.

1.3.1 «Таблиця калорійності»

«Таблиця калорійності» [6], як один із аналогів web-застосунку «Щоденник дієти» (Slim-Body), надає користувачам можливість вести облік спожитих продуктів та контролювати калорії. Вона дозволяє записувати інформацію про

продукти, шукати їх за назвою, що дає змогу точно вказати калорійність кожного продукту.

Що стосується розрахунку денної норми калорій, «Таблиця калорійності» виконують цю функцію, враховуючи важливі фактори, такі як вага, зріст, вік, рівень фізичної активності та ціль, яку хочу підкорити, тобто це або підтримання ваги або схуднення або набір ваги.

Відстеження зміни ваги — ще один важливий аспект, який підтримується в «Таблиці калорійності». Користувачі можуть фіксувати свою цільову вагу і спостерігати за змінами основної ваги на графіках.

Розрахунок індексу маси тіла (ІМТ) є важливою функцією для цієї системи. У таблиці калорійності можна легко обчислити (ІМТ), ввівши лише зріст і вагу.

Найбільшою відмінністю між «Slim-Body» та «Таблицею калорійності» є використання штучного інтелекту в вашому продукті. «Slim-Body» має функціонал, який дозволяє використовувати ШІ для аналізу введених даних і надання індивідуальних рекомендацій, що допомагає користувачам коригувати свій раціон і фізичні навантаження в реальному часі. У той час як таблиця калорійності є лише інструментом для запису даних, без можливості надання персоналізованих порад на основі отриманих результатів. Приклад головної сторінки «Таблиці калорійності» наведено на рисунку 1.1.

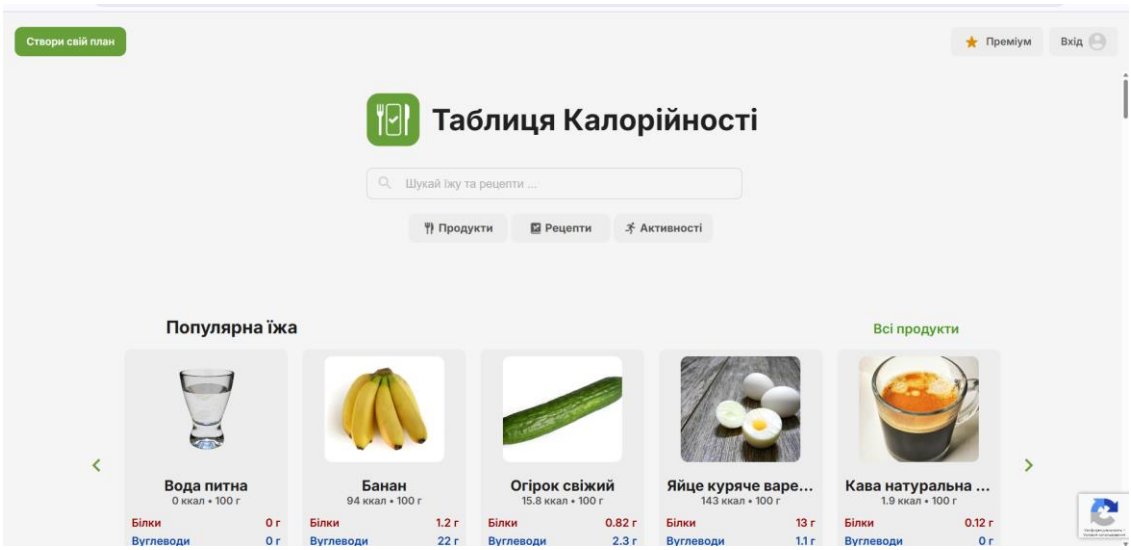


Рис. 1.1 Структура головної сторінки «Таблиці калорійності»

1.3.2 «MyFitnessPal»»

«MyFitnessPal» [7] – це ще один аналог web-застосунку «Щоденник дієти» (Slim-Body), який дозволяє користувачам вести облік спожитих продуктів та контролювати калорії. Застосунок надає можливість записувати інформацію про їжу, шукати продукти за їх назвою або сканувати штрих-коди для точного вказування калорійності кожного продукту. Це дозволяє користувачам легко відстежувати їх спожиті калорії та макроелементи протягом дня. Що стосується розрахунку денної норми калорій, «MyFitnessPal» також виконує цю функцію, враховуючи важливі фактори, такі як вага, зріст, вік, рівень фізичної активності та обрану мету — схуднення, підтримку ваги або набір маси. Це дозволяє точно визначити кількість калорій, яку потрібно споживати для досягнення бажаного результату.

Відстеження зміни ваги є ще одним важливим аспектом «MyFitnessPal». Користувачі можуть вносити дані про свою вагу і встановлювати цільову вагу, спостерігаючи за змінами за допомогою графіків і діаграм. Це дає можливість бачити прогрес і коригувати дієту або фізичну активність відповідно до досягнутих результатів. Розрахунок індексу маси тіла (ІМТ) також присутній у «MyFitnessPal». Користувачі можуть швидко розрахувати свій ІМТ, вводячи тільки вага і зріст. Це дозволяє визначити, чи є користувач у межах здорової ваги, і чи потрібно йому змінювати дієту або стиль життя для покращення загального стану здоров'я.

Однак, одним із суттєвих обмежень «MyFitnessPal» є відсутність штучного інтелекту (ШІ) та наявність веб-платформи. Приклад зареєстрованого користувача «MyFitnessPal» наведено на рисунку 1.2.

Це означає, що користувачі не отримують індивідуалізовані рекомендації, засновані на їхніх даних про харчування, фізичну активність чи прогрес у зниженні ваги. Всі розрахунки та рекомендації в «MyFitnessPal» базуються на загальних даних, без додаткової адаптації або коригування в реальному часі.

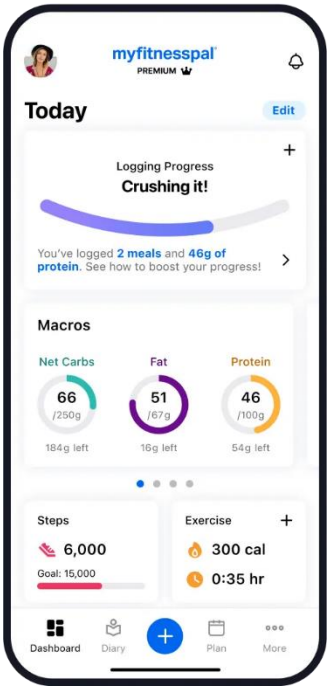


Рис. 1.2 Зареєстрований користувач «MyFitnessPal»

1.3.3 «Lose It!»

«Lose It!» - популярний веб-застосунок для контролю харчування та відстеження калорій. Він дозволяє користувачам легко записувати свої продукти харчування, шукати їх у базі даних . Застосунок надає інформацію про калорійність продуктів та їхні макроелементи, що дозволяє користувачам здійснювати облік споживаних калорій та оптимізувати своє харчування. Крім того, «Lose It!» дає можливість розрахувати індекс маси тіла (ІМТ) і денну норму калорій, що допомагає користувачам підтримувати здорову вагу та налаштовувати свій раціон відповідно до фізичних цілей.

Проте «Lose It!» має кілька суттєвих недоліків. Одним із них є відсутність відображення калорій безпосередньо на продуктах або стравах. Користувачі не можуть швидко оцінити, скільки калорій містить конкретний продукт або страва, оскільки інформація про калорії не завжди чітко виділена в описі продукту. Це може створювати незручності для користувачів, особливо, якщо вони хочуть швидко перевірити харчову цінність продуктів перед їх споживанням. Приклад відображення раціону на день «Lose It!» наведено на рисунку 1.3. Ще однією відсутньою функцією є штучний інтелект (ШІ). Відсутність ШІ в «Lose It! »

означає, що застосунок не може надавати персоналізовані рекомендації або аналізувати введені дані для оптимізації харчування [8].

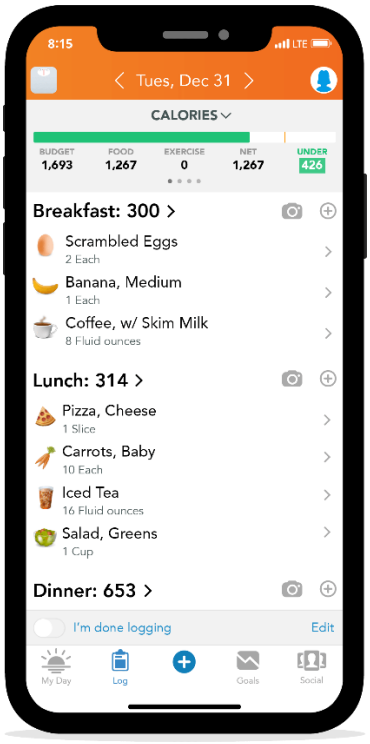


Рис. 1.3 Відображення раціону на день «Lose It!»

1.3.4 «FatSecret»

«FatSecret» [9] - інструмент для контролю харчування, який дозволяє користувачам вести облік спожитих продуктів, відстежувати калорії та інші поживні речовини. Завдяки широкій базі даних продуктів та можливості пошуку товарів, користувачі можуть швидко додавати продукти до свого щоденника харчування. Це дозволяє легко контролювати кількість калорій, білків, жирів і вуглеводів, що допомагає користувачам підтримувати здоровий раціон та досягати поставлених цілей.

Однією з важливих функцій цього застосунку є розрахунок денної норми калорій, який здійснюється на основі введених даних, таких як вага, зріст, вік, рівень фізичної активності та цілі користувача. За допомогою цього інструменту «FatSecret» дозволяє кожному користувачеві визначити, скільки калорій потрібно споживати для досягнення бажаного результату, будь то схуднення, підтримка ваги

або набір маси. Такий підхід дозволяє створити персоналізований план харчування, що відповідає індивідуальним потребам користувача.

Важливо додати, що є можливість відстеження зміни ваги. Користувачі можуть вводити свої показники ваги і стежити за їх динамікою, переглядаючи зміни через зручні графіки. Ця функція дозволяє відстежувати ефективність обраної дієти або програми фізичних навантажень і коригувати їх за потреби для досягнення бажаної ваги.

В «FatSecret» є розрахунок індексу маси тіла (ІМТ). Цей показник є важливим інструментом для оцінки здоров'я та визначення, чи знаходиться користувач у межах здорової ваги. Всі ці дані дозволяють отримати загальну картину фізичного стану і допомагають коригувати харчування та фізичну активність для досягнення найкращих результатів.

Попри свої переваги, «FatSecret» має й кілька суттєвих недоліків. Одним із них є відсутність можливості створення акаунту для збереження особистих даних і доступу до записів з будь-якого пристрою. У цьому застосунку користувачі можуть вести свій щоденник харчування, однак ці записи зберігаються тільки локально і не синхронізуються між пристроями без реєстрації. Це означає, що кожен раз після перевстановлення додатку або на іншому пристрої всі дані будуть втрачені, що обмежує зручність використання. Приклад відображення раціону на день «FatSecret» наведено на рисунку 1.4.

Крім того, відсутність штучного інтелекту (ШІ) у застосунку є ще одним важливим недоліком. Враховуючи, що інші рішення використовують ШІ для надання персоналізованих рекомендацій, «FatSecret» обмежений стандартними функціями, що не дозволяє йому адаптуватися під індивідуальні потреби користувача. У результаті застосунок не може здійснювати аналіз введених даних для оптимізації харчування і надавати користувачам індивідуальні поради щодо їхнього дієтичного плану.

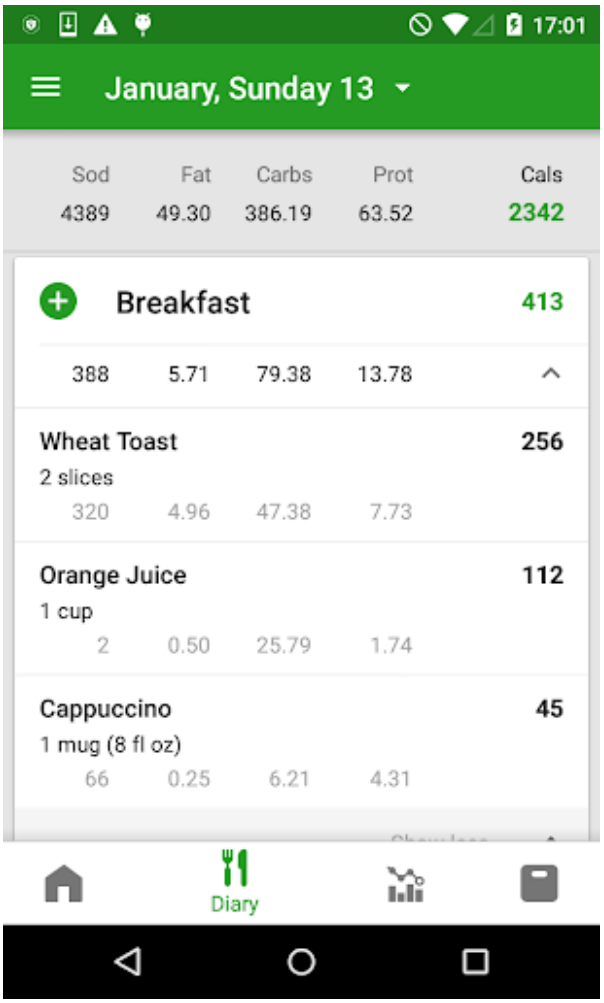


Рис. 1.4 Відображення раціону на день «FatSecret»

Зведені результати аналізу характеристик розглянутих додатків наведено у таблиці 1.1.

Таблиця 1.1

Зведені результати аналізу характеристик додатків для запису дієти.

Характеристика	Tablycjakalorijnosti	MyFitnessPal	Lose It!	FatSecret	Slim-Body
Наявність веб-платформи	+	-	+	+	+
Підрахунок калорій	+	+	+	+	+
Надання продуктів для діабетиків за допомогою ШІ.	-	-	-	+	+

Продовження таблиці 1.1

Зведені результати аналізу характеристик додатків для запису дієти.

Характеристика	Tablycjakalorijnosti	MyFitnessPal	Lose It!	FatSecret	Slim-Body
Список рецептів	+	+	+	+	+
Покажчики калорійності продуктів	+	+	-	+	+
Особистий кабінет	+	+	+	-	+
Можливість додавати продукти вручну	+	+	-	-	+

2 ЗАСОБИ РЕАЛІЗАЦІЇ

2.1 Середовище розробки

Інтегроване середовище розробки (IDE) - є програмним інструментом, що надає програмістам зручну платформу для написання, тестування та відлагодження програмного коду. IDE включає в себе текстовий редактор для написання коду, інструменти для компіляції та виконання програм, налагоджувач (debugger), а також інші засоби для зручного управління проєктом, таких як інтерфейси для роботи з базами даних або системами контролю версій.

Visual Studio Code (VS Code) — це популярне інтегроване середовище розробки, яке надає розробникам зручні інструменти для написання, налагодження та відлагодження коду. Це безкоштовний та відкритий редактор коду, розроблений компанією Microsoft, який підтримує безліч мов програмування, таких як JavaScript, Python, C++, HTML, CSS та багато інших. VS Code є одним з найпопулярніших IDE серед веб-розробників завдяки своїй легкості, швидкості та численним можливостям розширення.

Однією з ключових переваг Visual Studio Code є підтримка численних розширень, які дозволяють налаштувати середовище під конкретні потреби розробника. За допомогою розширень можна додавати підтримку нових мов програмування, налаштовувати форматування коду, інтегрувати з системами контролю версій, такими як Git, а також підключати різноманітні інструменти для автоматизації процесів, наприклад, для тестування чи побудови проєктів. Важливою особливістю є також вбудована підтримка Docker, що дозволяє розробникам працювати з контейнерами без необхідності використовувати окремі інструменти.

Інтерфейс Visual Studio Code дуже зручний і налаштовується під потреби користувача. Редактор має підсвічування синтаксису для багатьох мов, а також автозавершення коду, що значно прискорює процес розробки. Вбудований

термінал дозволяє запускати команди безпосередньо з IDE, що економить час і зменшує необхідність перемикатися між вікнами. Також є функція відлагодження коду, яка дозволяє швидко виявляти і виправляти помилки в процесі розробки.

Ще однією визначальною перевагою є підтримка багатьох платформ. Visual Studio Code доступний для Windows, macOS та Linux, що робить його універсальним вибором для розробників на різних операційних системах. Завдяки своїй легкості та модульності, VS Code дозволяє працювати з великими проєктами без проблем з продуктивністю.

Враховуючи ці можливості, Visual Studio Code є ідеальним середовищем для розробки веб-застосунків, включаючи додатки для запису та відстеження калорій, де зручність написання коду, швидкість його виконання та інтеграція з іншими інструментами мають важливе значення для ефективної роботи.

2.2 Мови програмування

Вибір мови програмування є ключовим етапом у розробці веб-застосунків, оскільки вона визначає, які інструменти та технології будуть використовуватися для створення функціональних можливостей додатку. Для розробки web-застосунку для запису та відстеження калорій використовуються різні мови програмування, кожна з яких виконує свою специфічну роль у серверній та клієнтській частинах проєкту.

2.2.1 JavaScript

JavaScript - це одна з найбільш поширених мов програмування для створення динамічних та інтерактивних веб-застосунків. JavaScript використовується на стороні клієнта для реалізації інтерфейсу користувача, забезпечення взаємодії з веб-сторінкою без необхідності перезавантажувати її. Ця мова дозволяє розробникам створювати інтерактивні елементи, обробляти введення користувачів, працювати з асинхронними запитами (через AJAX), а також взаємодіяти з іншими

веб-сервісами. Веб-застосунки для відстеження калорій, які мають інтуїтивно зрозумілий інтерфейс і взаємодіють з користувачем в реальному часі, неможливо реалізувати без використання JavaScript.

2.2.2 HTML та CSS

HTML (Hypertext Markup Language) є мовою розмітки, яка використовується для структурування та організації контенту веб-сторінок. Вона забезпечує основу для розташування текстових блоків, зображень, посилань і інших елементів, що складають зміст веб-ресурсу. У свою чергу, CSS (Cascading Style Sheets) виступає потужним інструментом для стилізації та оформлення цього контенту, дозволяючи надавати веб-сторінкам естетичний вигляд і забезпечувати привабливий візуальний досвід для користувача. Комбінація HTML та CSS створює ефективну платформу для побудови не лише структури веб-ресурсу, але і його дизайну, що є основою функціонального інтерфейсу будь-якого сучасного web-застосунку. Завдяки цій взаємодії можна гарантувати зручність та інтуїтивність користувацького досвіду, що є ключовим фактором у залученні та утриманні користувачів. Інтеграція цих технологій допомагає розробникам створювати гармонійний баланс між технічною функціональністю і привабливістю інтерфейсу, що суттєво підвищує якість взаємодії користувачів із додатком.

2.3 Веб-фреймворки

Веб-фреймворк [10] – це програмним фреймворком, спеціально розробленим для підтримки процесу створення та розробки веб-додатків. Він включає в себе широкий спектр застосувань, таких як веб-сервіси, веб-ресурси та веб-API [11]. Завдяки веб-фреймворкам розробникам надається систематизований підхід до побудови та розгортання веб-додатків в Інтернеті. Ці фреймворки зазвичай забезпечують набір інструментів та бібліотек, які спрощують рутинні завдання, такі як маршрутизація запитів, автентифікація користувачів або управління сеансами. Завдяки цьому, розробники можуть зосередити більше

уваги на унікальних аспектах своїх проєктів, підвищуючи ефективність процесу розробки та полегшуючи підтримку і масштабування кінцевих продуктів в мережі.

Node.js [12] - це серверне середовище виконання для JavaScript, яке дозволяє запускати JavaScript-код на сервері, що відкриває нові можливості для створення веб-застосунків. Node.js часто використовують як основу для побудови серверної частини веб-додатків, а разом з численними бібліотеками та інструментами, які надаються для цієї платформи, він фактично виконує функції фреймворка.

Node.js [13-14] активно використовується для розробки веб-застосунків у реальному часі, таких як чат-системи, платформи для відео-стримінгу, інструменти для обробки великих обсягів даних та інтерактивні додатки. Він також широко застосовується для створення RESTful API та серверних додатків, які обробляють запити від клієнтів і працюють з базами даних. Завдяки своїй продуктивності, масштабованості та можливості використовувати JavaScript на обох частинах додатка, Node.js є потужним інструментом для розробки сучасних веб-додатків.

2.4 База даних

База даних - це організований набір даних, що зберігаються і доступні в електронному вигляді. Вона дозволяє зберігати, управляти, запитувати та обробляти великі обсяги інформації, що використовуються додатками. Бази даних є важливою частиною будь-якої сучасної веб-розробки, оскільки вони забезпечують централізоване зберігання даних, що необхідні для функціонування веб-застосунків. Існує кілька типів баз даних, серед яких найбільш поширеними є реляційні та нереляційні бази даних:

1. Реляційні бази даних (SQL). Вони організовують дані в таблиці, що складаються з рядків і стовпців. Кожен рядок таблиці є окремим записом, а стовпці визначають типи даних, які зберігаються. Реляційні бази даних використовують мову запитів SQL (Structured Query Language) для роботи з даними. Прикладом реляційної бази є MySQL, PostgreSQL.

2. Нереляційні бази даних (NoSQL). Вони зберігають дані в більш гнучкій та масштабованій формі, не вимагаючи структури таблиць, як у реляційних базах. Ці бази даних використовуються для зберігання великих обсягів неструктурованих або слабо структурованих даних. До популярних нереляційних баз належать MongoDB, Cassandra, CouchDB.

У web-застосунку використовується нереляційна база даних Mongo. MongoDB - це одна з найбільш популярних нереляційних баз даних, що належить до категорії NoSQL. MongoDB використовує формат BSON (Binary JSON) для збереження даних, що дає можливість зберігати документи з гнучкою структурою, яка може містити вкладені об'єкти та масиви.

Це дозволяє значно спростити зберігання та обробку даних, які не мають чіткої таблиці, так як MongoDB підтримує зберігання об'єктів різної структури в одному колекції. MongoDB є чудовим вибором для веб-додатків, які потребують високої швидкості роботи з даними, легкості в масштабуванні та можливості працювати з великими обсягами даних.

Використання MongoDB в проєкті дозволяє нам зберігати інформацію про користувачів, їхні харчові записи та підрахунок калорій. Оскільки структура даних у додатку може змінюватися, MongoDB дає нам можливість гнучко адаптувати базу даних під різні типи даних і уникати складної міграції таблиць, як у реляційних базах [15].

2.5 Система контролю версій

Система контролю версій (Version Control System, VCS) — це механізм, що дозволяє відстежувати зміни у коді, зберігати різні версії проєкту та підтримувати ефективну співпрацю між розробниками. Завдяки таким системам розробники можуть вести історію модифікацій, повертатися до попередніх версій і узгоджено працювати в команді, синхронізуючи свої внески у проєкт. Однією з найвідоміших систем контролю версій є Git, а сервісом для хостингу Git-репозиторіїв, який здобув найбільшу популярність, є GitHub.

Git - це система управління версіями, яка дозволяє кожному програмісту мати особисту локальну копію репозиторію. Кожен розробник має доступ до повної історії змін і може працювати з проєктом незалежно від інших учасників, синхронізуючи свої зміни з центральним репозиторієм через команди push і pull. Git забезпечує зручне управління гілками, що дозволяє працювати над різними частинами проєкту паралельно, згодом зливаючи зміни в одну головну гілку.

GitHub - це онлайн-платформа для хостингу репозиторіїв Git, яка надає інтерфейс для зберігання та спільної роботи над проєктами. GitHub дозволяє зберігати код у вигляді репозиторіїв, надаючи можливість створювати окремі гілки для розробки нових функцій, а також робити pull-запити для об'єднання змін з головною гілкою проєкту. GitHub є одним з найбільш популярних сервісів серед розробників завдяки своїй простоті у використанні, потужним можливостям для співпраці та інтеграції з іншими інструментами.

3 ПРОЄКТУВАННЯ

3.1 Проєктування архітектури web-застосунка

Додаток розроблений за принципом клієнт-серверної архітектури, яка забезпечує чіткий розподіл обробки даних між серверною і клієнтською частинами. Такий підхід дозволяє раціонально розподіляти обчислювальні ресурси, забезпечуючи ефективність виконання задач. Завдяки розподілу ролей система стає більш гнучкою для масштабування, що сприяє підвищенню її загальної продуктивності та стабільності в роботі навіть при збільшенні обсягів даних чи навантаження. На рисунку 3.1 продемонстрована архітектура web-застосунку.

Клієнтська частина web-застосунку, яка розробляється за допомогою стандартних технологій HTML, CSS і JavaScript, відповідає за відображення інтерфейсу для користувача. Вона взаємодіє з серверною частиною, надсилаючи HTTP-запити для отримання або відправлення даних. Клієнт може, надіслати запит на створення продукта та для відображення продуктів, які є в наявності. Для цього використовуються технології JavaScript, які забезпечують динамічну взаємодію з користувачем, а також відправлення асинхронних запитів до серверу через REST API.

Серверна частина застосунку, що розробляється на платформі Node.js з використанням фреймворку Express, виконує роль обробки запитів, отриманих від клієнтської частини. Кожен запит, надісланий клієнтом, містить інформацію, яку сервер обробляє і взаємодіє з базою даних. Сервер може отримати запит на збереження даних про продукти, обробити його, зберегти інформацію в базі даних і надіслати відповідь клієнту.

Важливою частиною серверної частини є використання бібліотеки Nodemailer, що дозволяє відправляти електронні листи. Наприклад, це може бути підтвердження реєстрації користувача або нагадування про виконання дієти.

Взаємодія з базою даних здійснюється через MongoDB, яка є основною базою даних для зберігання інформації про користувачів та їхні харчові звички. MongoDB є нереляційною базою даних, це дає змогу зберігати дані у форматі BSON, що гарантує гнучкість при зберіганні, як структурованих, так і неструктурованих даних. Сервер використовує бібліотеку Mongoose, щоб взаємодіяти з MongoDB. Коли клієнт відправляє запит на сервер, сервер обробляє цей запит і взаємодіє з базою даних для збереження або отримання необхідної інформації, а потім повертає відповідь на клієнтську частину.

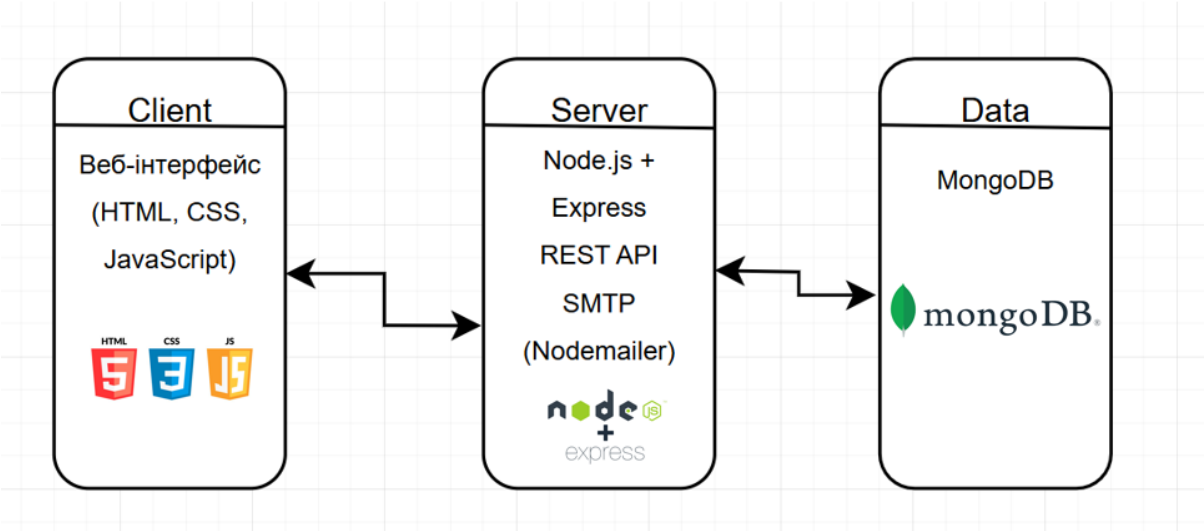


Рис. 3.1 Архітектура web-застосунку «Slim-Body»

3.2 Архітектура клієнтської частини

В проєкті є дві ролі користувачів: зареєстровані та незареєстровані. Кожна з цих ролей має свої можливості та доступ до різних функціональних частин web-застосунку.

Зареєстровані користувачі мають повний доступ до функціоналу особистого кабінету, який є основною сторінкою для управління особистими даними. Користувач може відстежувати свою вагу, вводити нові дані про фізичний стан і прогрес у дієті. Крім того, для зареєстрованих користувачів доступна можливість створювати продукти для подальшого використання в системі. Це дає змогу

користувачам додавати власні продукти до бази даних і підраховувати калорії на основі введених даних.

Іншою важливою функцією є можливість запису продуктів. Це дозволяє зберігати інформацію про спожиті продукти харчування та використовувати її для подальших обчислень, таких як автоматичний підрахунок калорій за день. Цей процес включає підрахунок загальної калорійності спожитої їжі за день, що допомагає користувачам стежити за своїм раціоном та коригувати його при необхідності.

Додатково, для зареєстрованих користувачів передбачена функція перегляду продуктів, додавати продукти в улюблене, переглядати вже створені продукти чи страви. Важливою функцією є AI-помічник, який відображає доступні продукти в застосунку, а також показує продукти в яких найменше вуглеводів, це дуже важливо для користувачів з хворобою діабет. При бажані користувач може спілкуватися з ШІ на будь-які теми та отримувати персоналізовані рекомендації щодо дієти і допомагаючи коригувати харчові звички. Також зареєстровані користувачі можуть переходити на всі сторінки, що й незареєстровані та користуватися всім функціоналом.

Для незареєстрованих користувачів web-застосунок має обмежений доступ до функціоналу, але все ж дозволяє використовувати базові можливості. Вони можуть переглядати популярні продукти, що надається для ознайомлення з найпопулярнішими продуктами в системі. Користувачі можуть переглядати популярні рецепти, що дозволяє їм ознайомитись з рецептами. Крім цього, незареєстровані користувачі можуть перейти на сторінку рецепта, щоб переглянути деталі певного рецепта, але вони не можуть взаємодіяти з ним, наприклад, зберігати чи додавати до свого списку вподобаних. Вони також можуть здійснювати пошук рецепту, що дозволяє їм знаходити рецепти за певними критеріями або інгредієнтами.

Іншою функцією для незареєстрованих користувачів є розрахунок добової калорійності, що дозволяє оцінити необхідну кількість калорій для підтримки здорової дієти без реєстрації. Однак для отримання більш персоналізованих

рекомендацій і доступу до збережених даних, таких як історія калорій чи ваги, користувачам пропонується реєстрація. Після реєстрації користувачі отримають доступ до розширених функцій.

Діаграма варіантів використання (Use Case Diagram) - це різновид діаграми в UML (Unified Modeling Language), що застосовується для візуалізації функціональності системи з позиції користувачів та їх взаємодії з системою. Вона описує варіанти використання і взаємодії між акторами та системою. Це допоможе зрозуміти, як користувачі будуть взаємодіяти з додатком, які функції доступні для різних типів користувачів і які сценарії вони можуть виконувати [16].

Діаграма компонентів (Component Diagram) — це діаграма в UML, яка використовується для моделювання фізичних компонентів програмного забезпечення та їх взаємодії. Вона зображує структуру системи з точки зору компонентів програмного забезпечення, що реалізують різні функціональні можливості, а також відносини між ними. Діаграма компонентів допомагає розробникам та архітекторам системи зрозуміти, як різні частини програмного забезпечення взаємодіють між собою [17].

Всю архітектуру клієнтської частини наглядно продемонстровано в діаграмах: «Варіантів використання» на рисунку 3.2 та діаграмі «Компонентів» рисунок 3.3.

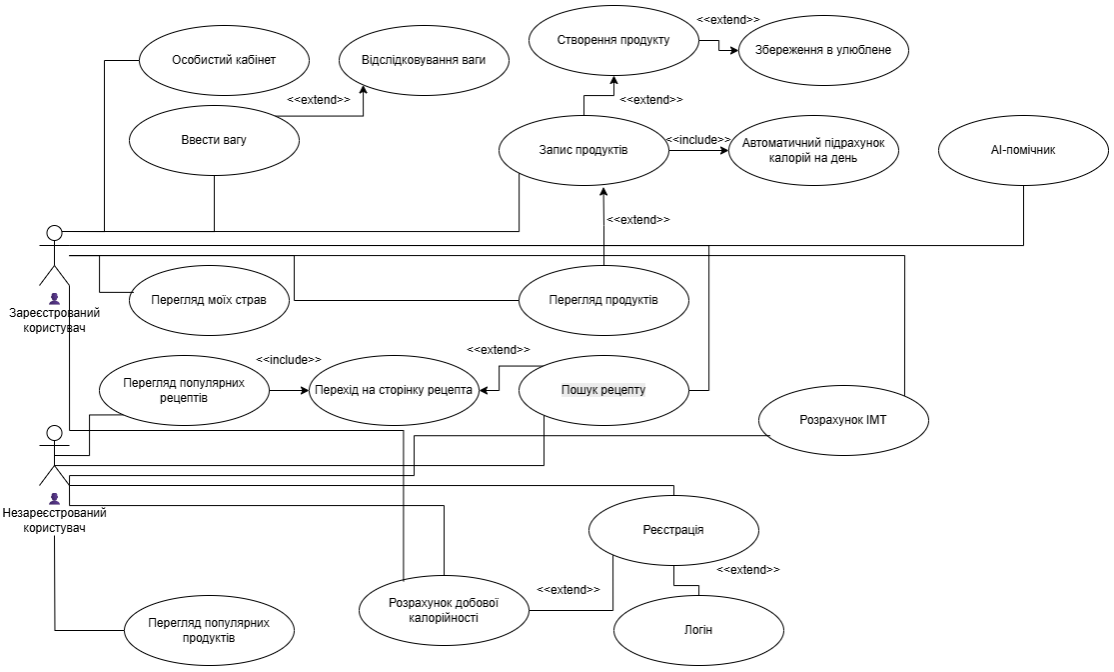


Рис. 3.2. Діаграма Варіантів використання

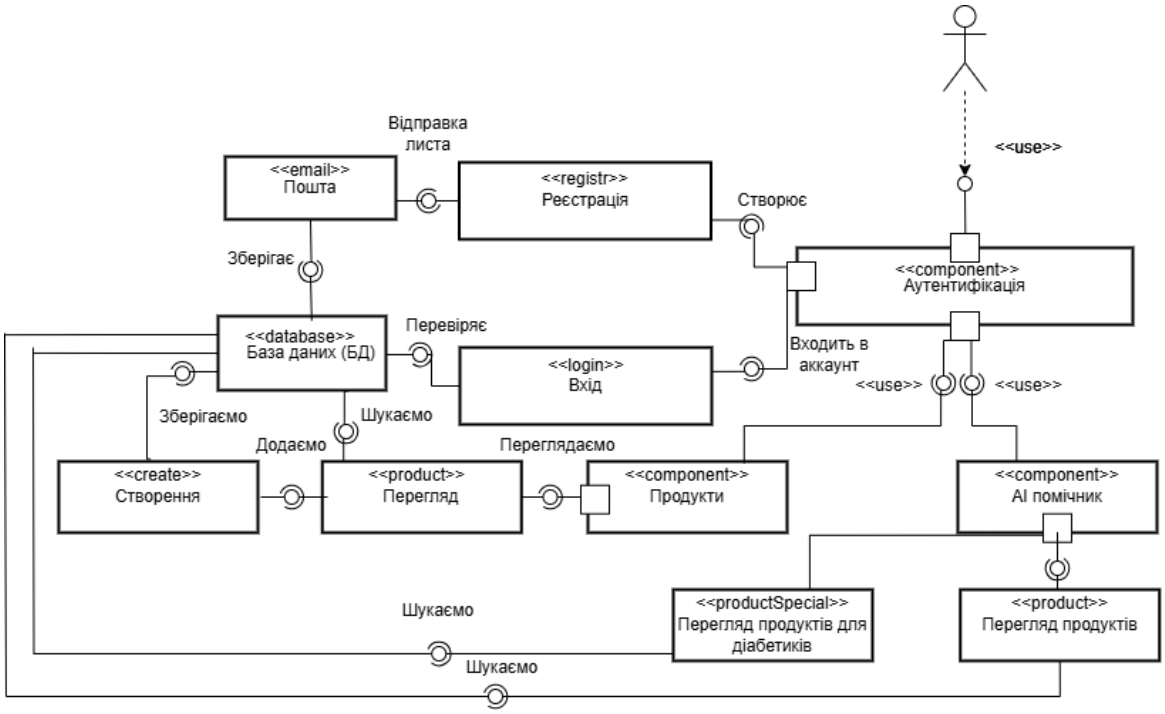


Рис. 3.3 Діаграма Компонентів

3.3 Архітектура серверної частини

Архітектура серверної частини web-застосунку визначає організацію та структуру компонентів, які виконують основні функції обробки запитів, зберігання та взаємодії з даними, а також забезпечують комунікацію між клієнтською частиною і базою даних. Важливою частиною серверної архітектури є її гнучкість, масштабованість та безпека, оскільки вона забезпечує основні сервіси, які підтримують функціонування всього додатку.

У даному випадку серверна частина реалізована на платформі Node.js з використанням фреймворка Express, що дозволяє ефективно організувати обробку запитів і маршрутизацію в веб-додатку. Node.js є асинхронним середовищем виконання для JavaScript, що дозволяє створювати високопродуктивні сервери, здатні обробляти велику кількість одночасних запитів.

Діаграма класів — це тип діаграми в UML, котра використовується для моделювання структури системи за допомогою класів та їх зв'язків. Діаграма класів описує класи об'єктів, їх атрибути, методи та відносини між ними. Вона дозволяє

чітко визначити, як складові системи взаємодіють між собою.

На рисунку 3.4 показана діаграма класів [18].

Клас ІМТ (Індекс маси тіла) описує обчислення індексу маси тіла користувача. Він має три атрибути: вага (тип даних - float), зріст (тип даних - float) та метод розрахунку ІМТ. Клас ІМТ має зв'язок "один до одного" з класом Користувач, оскільки кожен користувач має лише один ІМТ.

Клас Користувач є основним класом, який містить атрибути: id (ідентифікатор користувача), ім'я (тип даних - string), email (тип даних - string), пароль (тип даних - string) та дата реєстрації. Методи класу включають: реєстрація(), вхід(), переглянути Продукти(), переглянути Мої Страви(), ввести Вагу() та отримати Рекомендовану Калорійність(). Користувач має зв'язки "один до багатьох" з класами Продукт, Рецепт та Раціон. Це означає, що один користувач може взаємодіяти з кількома продуктами, рецептами та раціонами.

Клас Продукт описує окремий продукт з такими атрибутами, як назва (тип даних - string), калорії (тип даних - int), тип (тип даних - string) і популярність (тип даних - float). Методами класу є: створити(), зберегти В улюблене() та додати До Раціону(). Клас Продукт має зв'язок "багато до одного" з класом Раціон, оскільки один продукт може бути частиною кількох різних раціонів.

Клас Рецепт включає атрибути, що описують рецепт страви: назва (тип даних string), інгредієнти (тип даних - string), калорійність (тип даних - int) та популярність (тип даних - float). Методи класу включають перегляд рецептів та пошук рецептів. Рецепти пов'язані з користувачем, оскільки користувач може переглядати рецепти та знаходити відповідні страви для себе.

Клас Особистий Кабінет є частиною класу Користувач і надає доступ до особистої інформації користувача. Методи класу включають відслідковування ваги користувача та перегляд страв, що дозволяє користувачеві організовувати свій раціон та контролювати процес схуднення чи підтримання здоров'я.

Клас AI-помічник допомагає користувачеві з аналізом його раціону та надає рекомендації по продуктах. Методи цього класу включають аналіз Раціону() та рекомендації По Продуктах(), що дозволяє користувачеві отримувати

індивідуальні поради щодо харчування.

Клас Раціон описує раціон харчування користувача і має атрибути: дата (тип даних - date), список продуктів (масив об'єктів класу Продукт) та загальна калорійність (тип даних - int). Методи класу включають додати Продукт(), автоматичний Підрахунок Калорій(), що дозволяє додавати нові продукти до раціону та автоматично підраховувати кількість спожитих калорій.

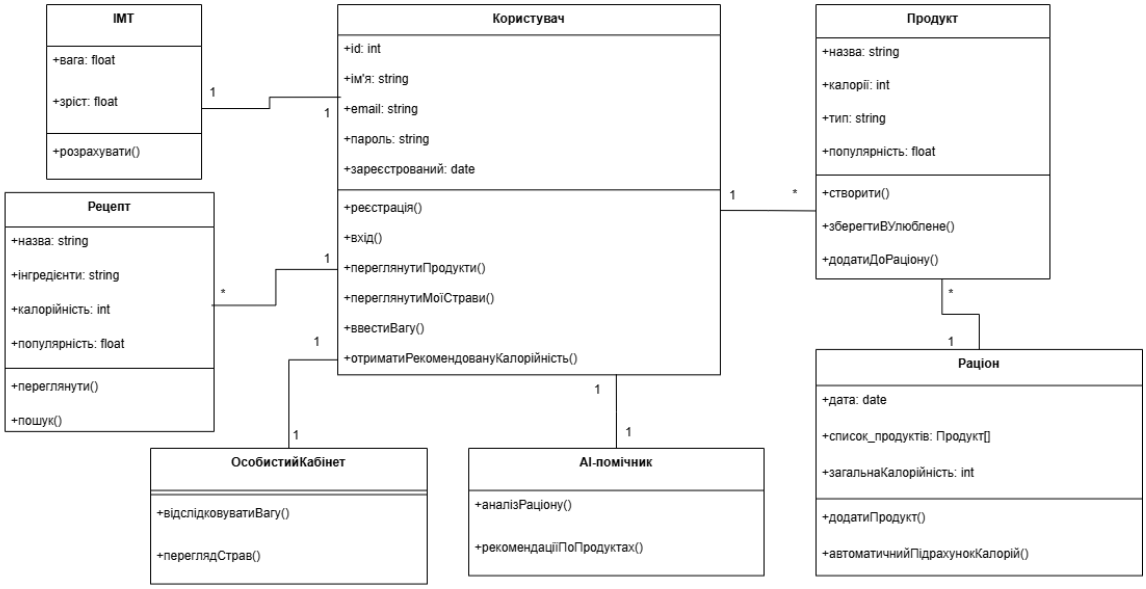


Рис. 3.4 Діаграма Класів

3.4 Архітектура бази даних

У web-застосунку використовується база даних MongoDB, яка є нереляційною.

Не реляційна база даних (NoSQL база даних) — це тип бази даних, який не використовує традиційну реляційну модель даних, що базується на таблицях з фіксованою структурою (як у SQL). Вона дозволяє зберігати дані в різних форматах: документи, графи, пари ключ-значення або стовпці, замість строго визначених рядків і стовпців.

ER-діаграма (Entity-Relationship) являє собою графічне зображення структури бази даних, яке показує взаємодії між різними сутностями (об'єктами). Вона є інструментом для моделювання даних та виявлення взаємозв'язків між об'єктами, що зберігаються в межах бази даних. ER-діаграми відіграють ключову роль у процесі проектування реляційних баз даних, сприяючи кращому розумінню організації даних та їх взаємозв'язків. На рисунку 3.5 продемонстровано ER діаграма проєкту [19].

ER діаграма описує структуру бази даних для системи, в котрій є кілька сутностей, таких як User, Registration, Food, і Login. Всі ці сутності взаємодіють одна з одною через різні зв'язки.

У таблиці User зберігаються дані про користувачів. Кожен користувач має унікальний ідентифікатор user_id, електронну пошту, хеш пароля та дату створення акаунта. Ця сутність має зв'язок з іншими сутностями, зокрема з таблицею Post і Registration. Кожен користувач може мати кілька постів, і кожен користувач також має один запис у таблиці реєстрацій.

Таблиця Registration зберігає інформацію про реєстрацію користувачів. Кожна реєстрація має унікальний ідентифікатор registration_id, зв'язок з користувачем через зовнішній ключ user_id, дату реєстрації та статус перевірки (is_verified).

Таблиця Food містить дані про їжу, яку додають користувачі. Кожен продукт має унікальний ідентифікатор food_id, ім'я продукту, кількість калорій на 100 грамів і дату створення. Ця таблиця також пов'язана з таблицею User, оскільки кожен продукт створюється певним користувачем.

Таблиця Login має ідентифікатор login_id, який є її первинним ключем (PK). У ній також є зовнішній ключ user_id, який посилається на таблицю User, щоб визначити, який саме користувач здійснив вхід. Зберігається дата і час входу користувача в систему через атрибут logined_at. Крім того, є поле is_verified, яке вказує, чи був вхід перевірений.

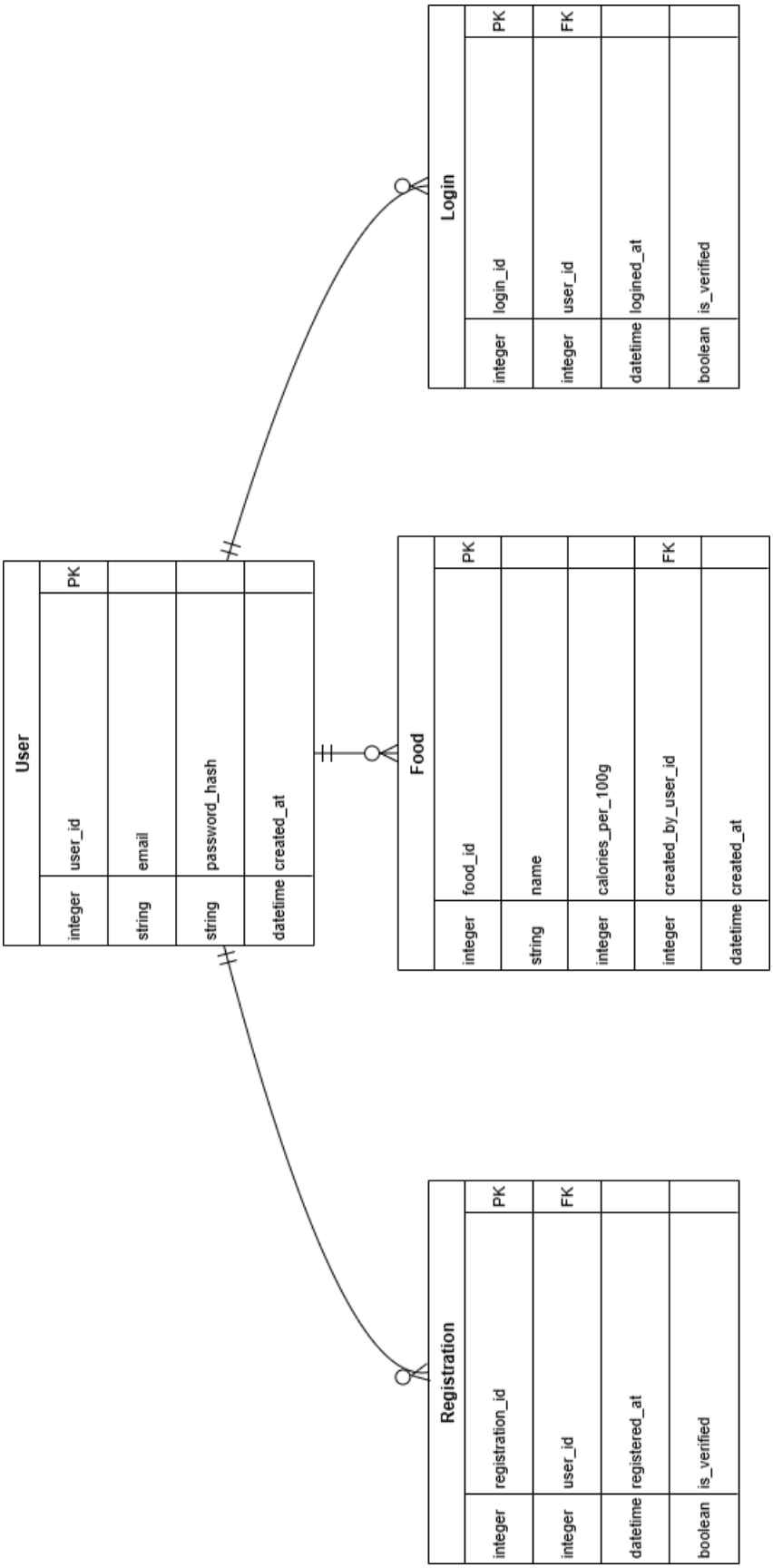


Рис. 3.5 ER діаграма

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.1 Підготовка середовища

Для розробки web-застосунку використовуються інструменти, які розглядали у попередніх розділах, а саме: Visual Studio, Git. Також для роботи з базами даних треба встановити MongoDB.

Після встановлення VS Code, потрібно додатково налаштувати кілька важливих пакетів. Для роботи з MongoDB необхідно встановити плагін MongoDB для Visual Studio Code, який дозволяє безпосередньо взаємодіяти з базою даних прямо з редактора. Також варто встановити плагін ESLint для перевірки якості коду, а для роботи з Node.js — плагін Node.js Extension Pack, який включає корисні інструменти для налагодження та відладки. Продемонстровано на рисунку 4.1.

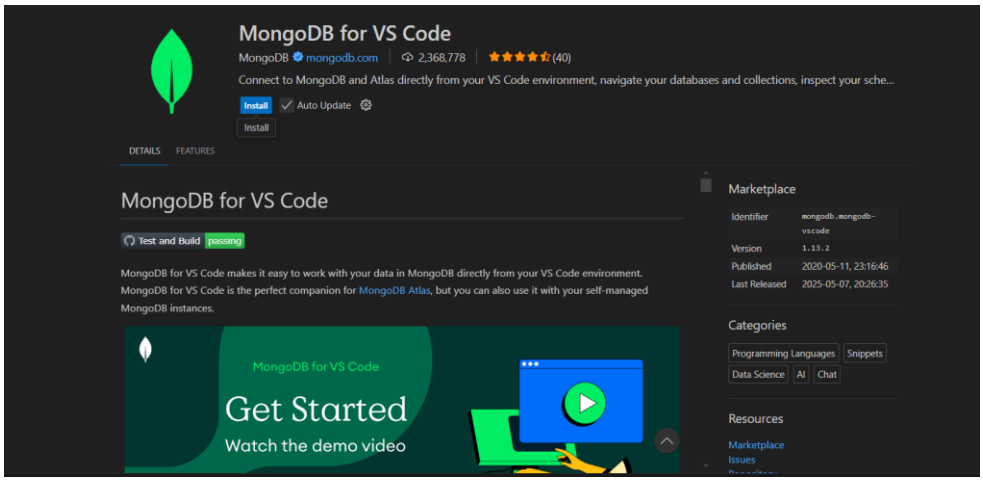


Рис. 4.1 Плагін MongoDB для Visual Studio Code

Для роботи з Git зручніше використовувати GitHub Desktop. Цей інструмент дозволяє безпосередньо керувати репозиторіями, зручно комітити зміни та переносити файли через графічний інтерфейс. За допомогою GitHub Desktop можна легко переміщати файли між папками в проекті (див. рисунок 4.2).

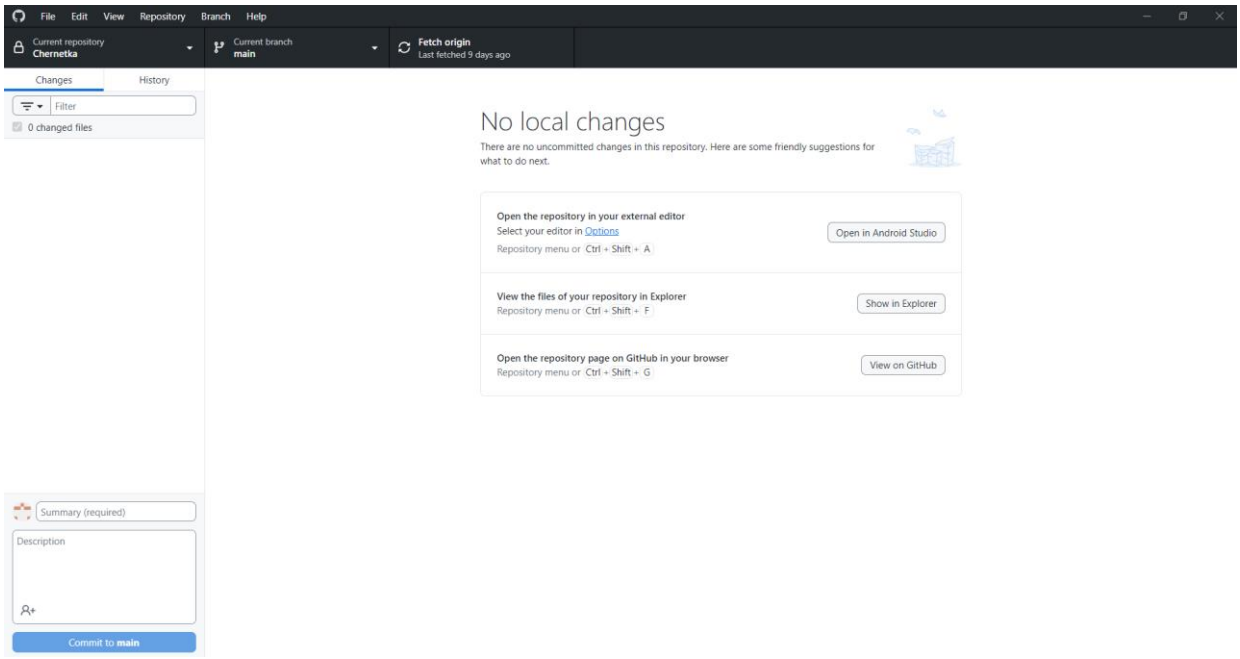


Рис. 4.2 GitHub Desktop

MongoDB, необхідно встановити останню стабільну версію, яка підтримує усі необхідні функції для вашого проекту. MongoDB треба мати версію 8.0.9 для забезпечення сумісності з більшістю сучасних фреймворків. MongoDB дозволяє зберігати дані в документному форматі BSON, що дуже зручно для зберігання структури даних у веб-застосунках (див. рисунок 4.3).

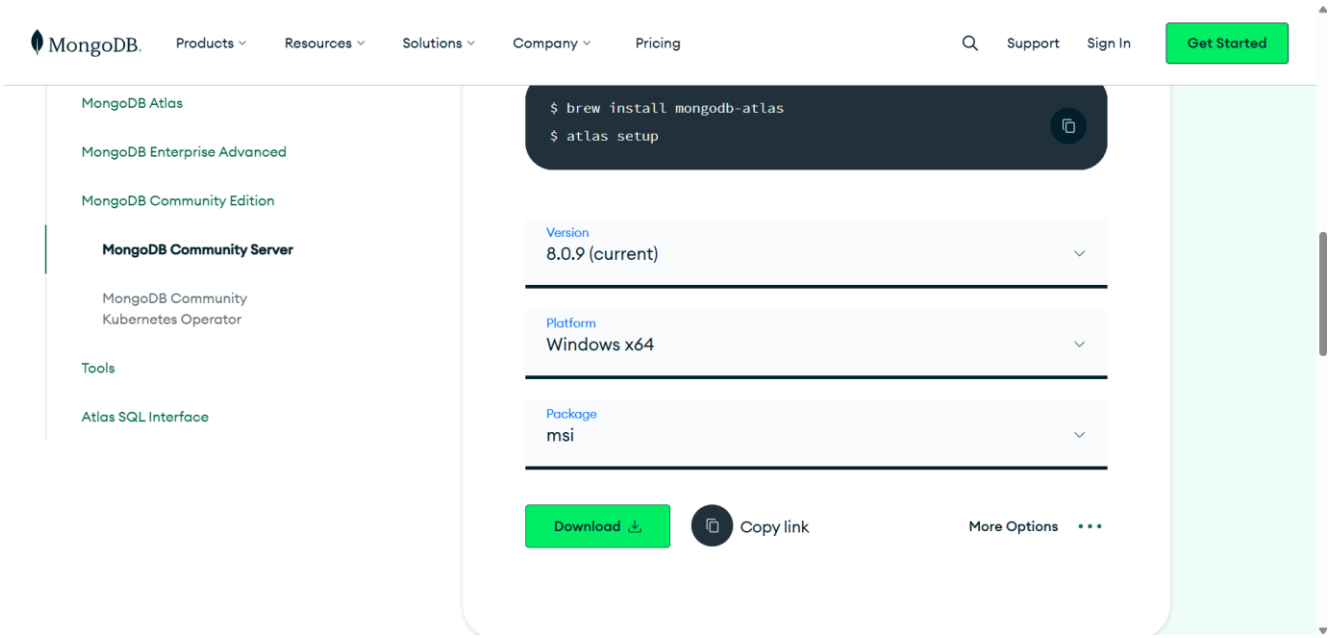


Рис. 4.3 GitHub Desktop

4.2 Реалізація клієнтської частини

В директорії проекту Slim-Bode створено папку фронтенд, яка містить 11 файлів сторінок html, 7 файлів стилів та 8 файлів скриптів, також шрифти та фотографії. На рисунку 4.4. показано папку фронтенда.

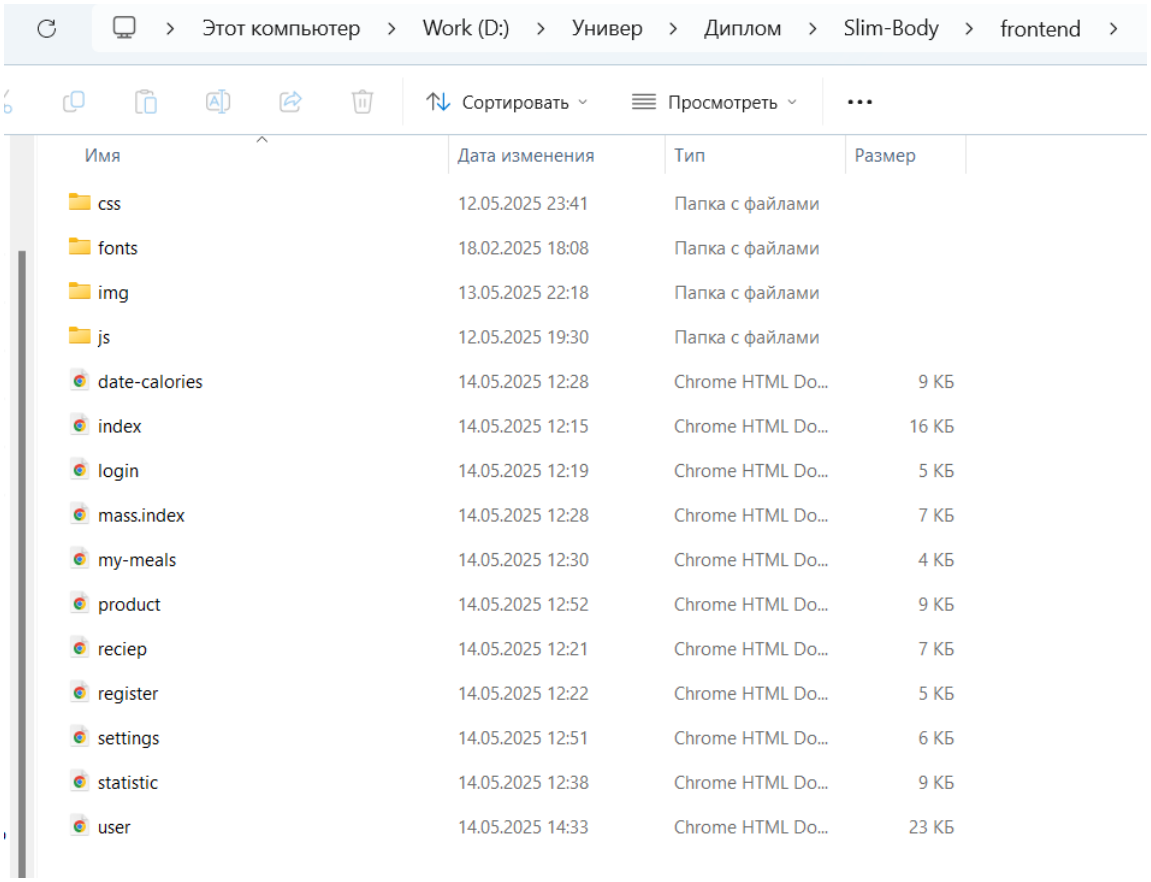


Рис. 4.4 Папка фронтенд

4.2.1 Index.html

Index.html – головна сторінка не тільки користувача, а й проєкта з цієї сторінки можна перейти на будь-яку, завдяки меню в шапці. На рисунку 4.5 показана головна сторінка.

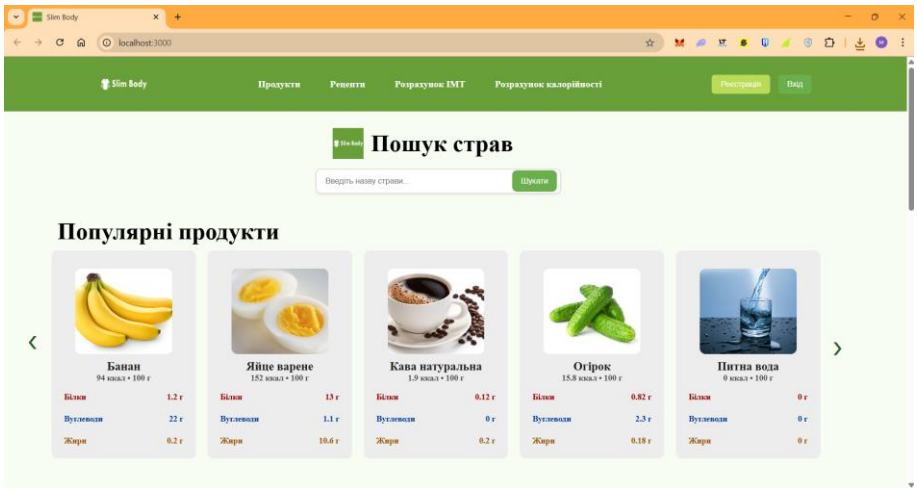


Рис. 4.5 Головна сторінка

Головна сторінка має пошук страв, де користувачі можуть знайти одразу страву, а не гортати карусель. На рисунку 4.6 продемонстрований пошук страви за назвою.

Рис. 4.6 Пошук страви на головній сторінці

Далі розташовано карусель карток популярні продукти на рисунку 4.7. В цій каруселі продукти змінюються динамічно завдяки файлам Json на сервері. Якщо спуститися трішки вниз можем побачити, ще одну карусель - це популярні рецепти, на рисунку 4.8, яка реалізовано також як динамічні данні Json але є відмінність, що при натисканні на рецепт відкривається окрема сторінка рецепта, що містить детальнішу інформацію про спосіб приготування та інгредієнти.

На рисунку 4.9 продемонстровано відкриття сторінок з рецептами. Це реалізовано завдяки використанню технології EJS для рендерингу шаблонів, де дані з JSON файлу динамічно підставляються в HTML-код сторінки. Кожен рецепт має свій унікальний ідентифікатор, і при натисканні на один з елементів каруселі,

відправляється запит на сервер, який за допомогою Node.js обробляє цей запит і надає відповідну інформацію про рецепт. В результаті користувач потрапляє на сторінку з детальною інформацією, де відображаються інгредієнти, спосіб приготування та інші деталі.

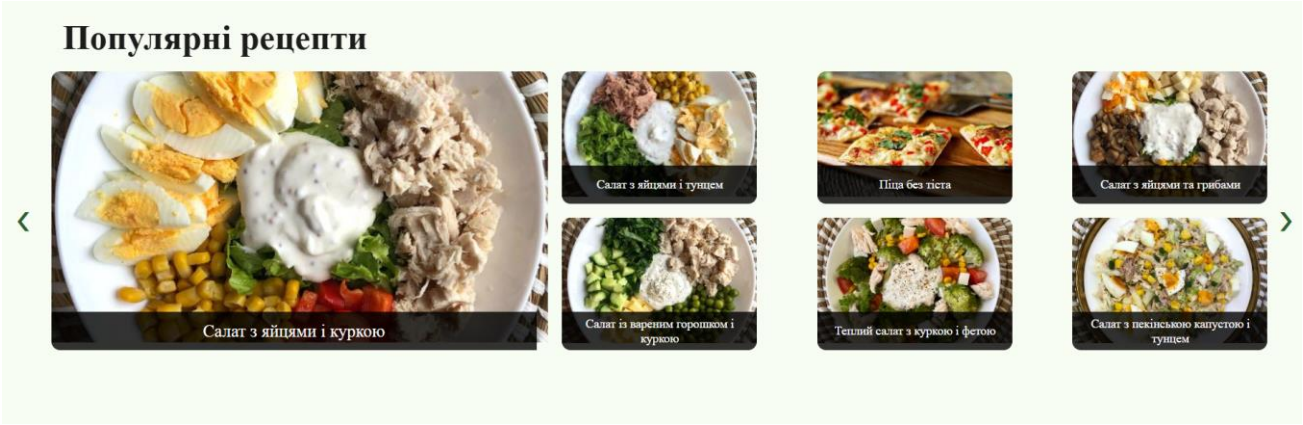


Рис. 4.7 Карусель популярні рецепти на головній сторінці

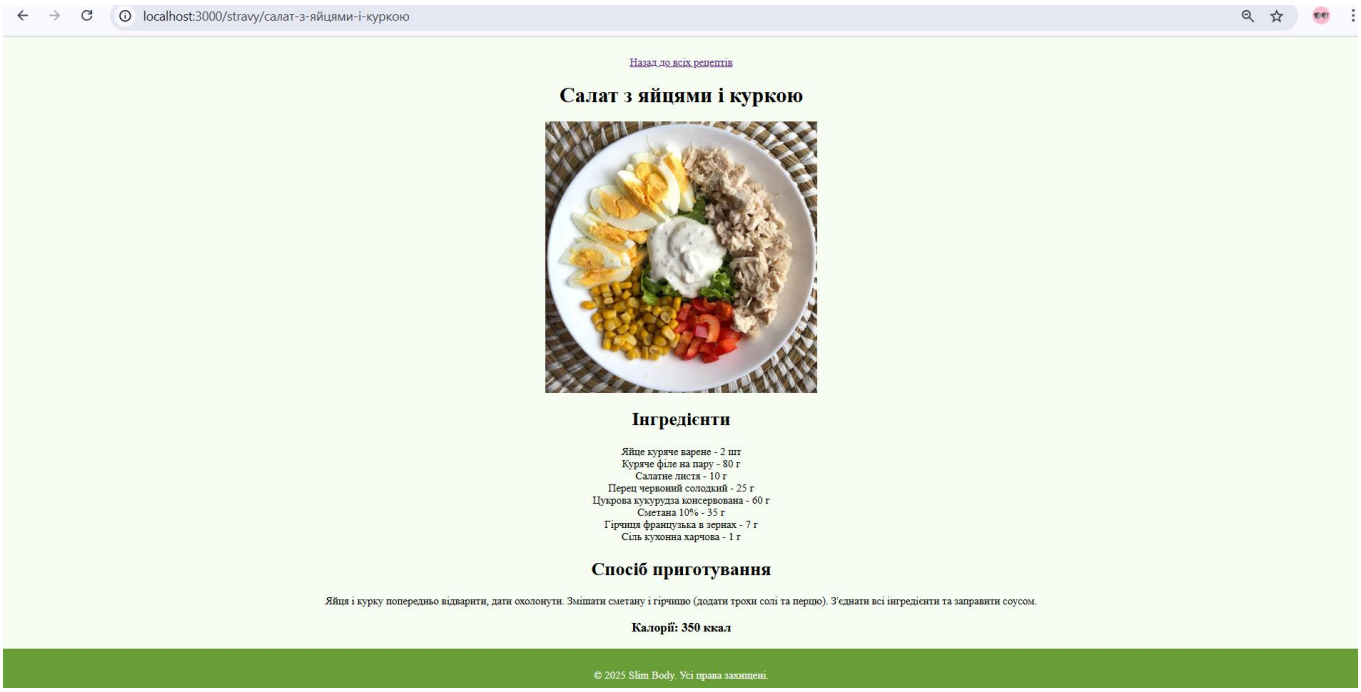


Рис. 4.8 Популярні рецепти

4.2.2 Сторінка реєстрації

Якщо користувачу захоче мати повний функціонал проєкту, то він переходить на сторінку реєстрації, де вводить коректну пошту, бо система має

перевірки на правильність, та пароль. На рисунку 4.9 продемонстровано сторінка реєстрації. Коли користувач успішно реєструється, сервер використовує nodemailer для відправки повідомлення на вказану електронну адресу користувача.

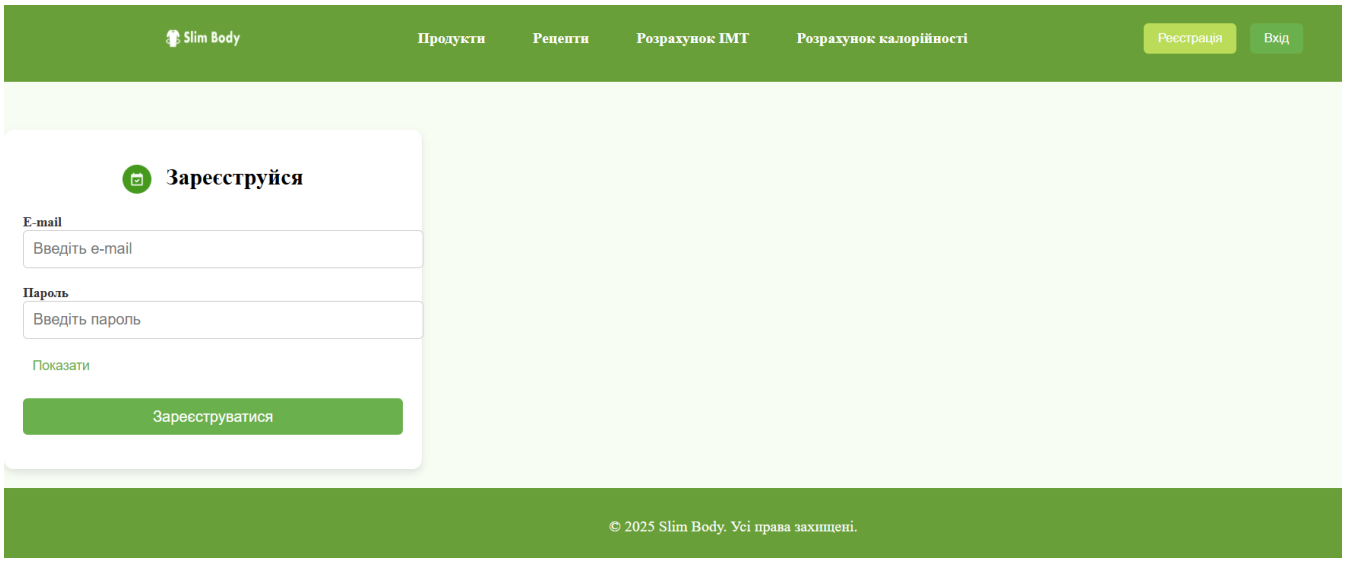


Рис. 4.9 Сторінка реєстрації

На рисунку 4.10 показано лист на пошті.

Рис. 4.10 Лист на пошті

Отже, як це працює: спочатку сервер імпортує модуль nodemailer. На рисунку 4.11 показаний модуль nodemailer.

```
const nodemailer = require("nodemailer");
```

Рис. 4.11 Модуль nodemailer

Далі створюємо транспортер за допомогою createTransport, як показано на рисунку 4.12.

```
const transporter = nodemailer.createTransport({
 service: "gmail",
 auth: {
   user: "maksimvertij416@gmail.com",
   pass: "lqne tjik hjfj hmkh",
 },
});
```

Рис. 4.12 Транспортер

Далі створюємо об'єкт mailOptions, який містить інформацію про те, який лист потрібно відправити користувачу. На рисунку 4.13 показано привітальне повідомлення по пошту. Також в майбутньому можна покращити повідомлення, зробить посилання через яке користувач підтверджує, що це саме він реєструвався, а не випадково хтось ввів його пошту.

```
const mailOptions = {
  from: "maksimvertij416@gmail.com",
  to: email,
  subject: "Slim-Body",
  text: "Вітаємо ви пройшли реєстрацію тепер уввідійть у свій профіль",
};
```

Рис. 4.13 Привітальне повідомлення

І використовуємо **sendMail** для відправки листа, як бачимо на рисунку 4.14.

```
await transporter.sendMail(mailOptions);
console.log('Лист успішно відправлено');
```

Рис. 4.14 Відправка листа

4.2.2 AI помічник

AI помічник знаходиться на сторінці зареєстрованого користувача в user.html. (див. рисунок 4.15).

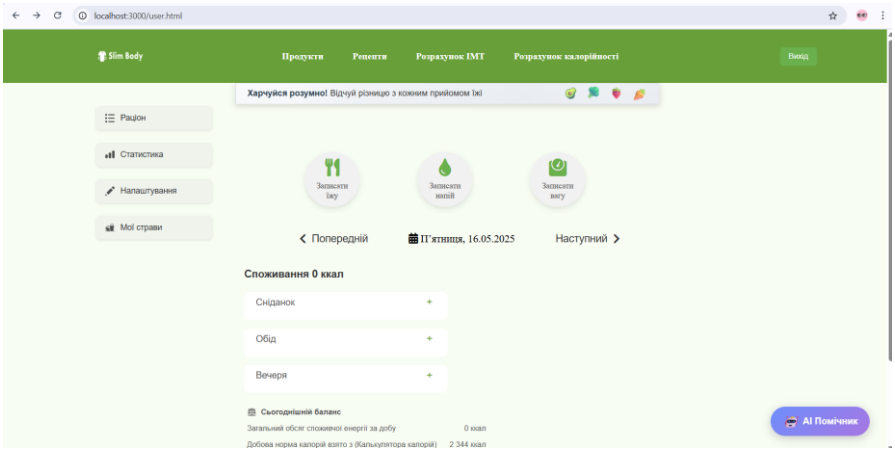


Рис. 4.15 AI помічник

При натисканні на кнопку AI помічника відкривається модальне вікно з чатом, де користувач може переписуватись, як з звичайний чатом Gemini так і отримати наявність всіх продуктів у застосунку та продуктів з найменшим показником вуглеводів для діабетиків (див. рисунок 4.16).

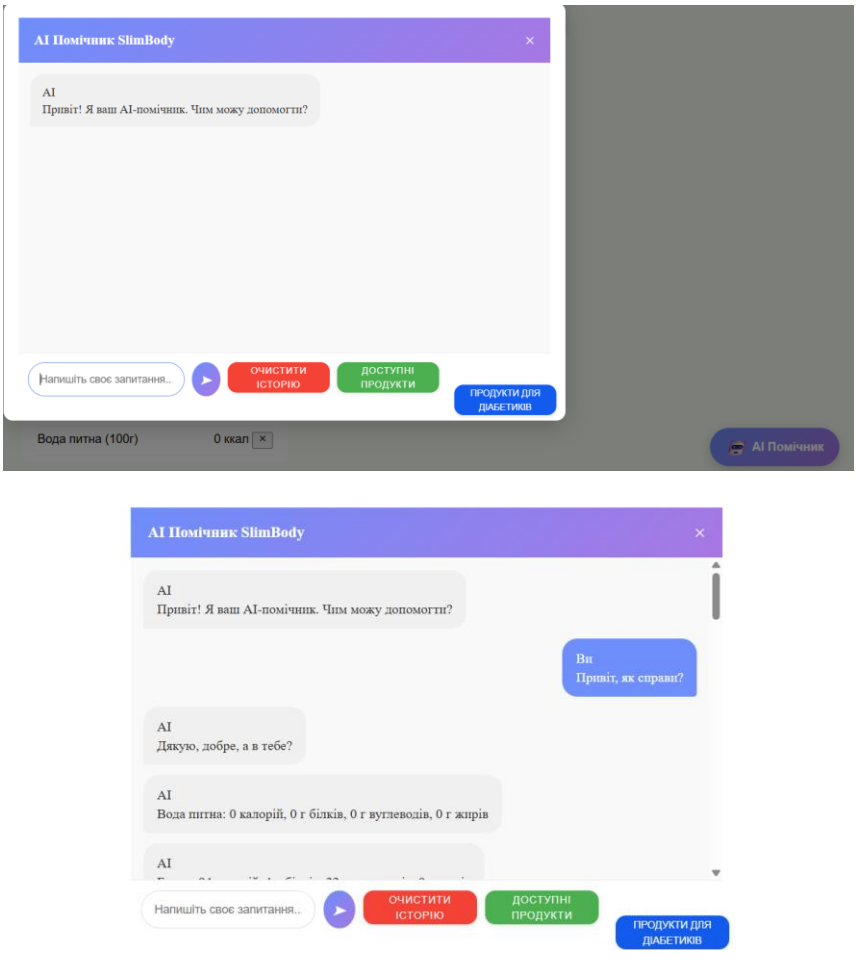


Рис. 4.16 Модальне вікно AI помічника

Ось як це працює:

- Клієнтська частина (JavaScript на фронтенді) — користувач вводить запит у чаті, і це повідомлення передається на сервер через API запит. Запит передається через POST запит до маршруту /api/gemini. На рисунку 4.17 відображена функція відправки запиту до API.

```
async function fetchAIResponse(message, signal) {
  console.log('Sending message:', message);

  const response = await fetch(config.apiUrl, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({ message: message, model: 'gemini-1.5-flash' }),
    signal,
  });
```

Рис. 4.17 AI помічник запит до API.

- Сервер отримує повідомлення через POST запит на маршруті /api/geminin та використовує метод GoogleGenerativeAI для генерації відповіді. На рисунку 4.18 показано частина кода з сервера.

```
const result = await model.generateContent(message);
const response = await result.response;
```

Рис. 4.18 AI помічник отримання запиту з API.

Якщо відповідь на запит успішно отримана, вона повертається назад клієнту в JSON форматі, як показано на рисунку 4.19.

```
if (response && response.text) {
  const text = await response.text();
  return res.json({
   user_message: message,
   bot_reply: text,
   status: 'success'
  });
} else {
  console.error('Не вдалося отримати текст від AI');
  return res.status(500).json({
   error: 'Не вдалося отримати відповідь від AI',
   status: 'no_response'
  });
```

Рис. 4.19 AI помічник перевірка відповіді

І тепер клієнтська частина отримує відповідь від AI і виводить її у чаті, як показано на рисунку 4.20.

```
addMessage('ai', aiResponse);
```

Рис. 4.20 AI помічник вивід відповіді.

4.3 Реалізація серверної частини

Для розробки сервера спочатку необхідно завантажити потрібні бібліотеки:

1. Express - це мінімалістичний веб-фреймворк для Node.js, який спрощує створення серверних додатків, обробку запитів і маршрутизацію. Для його активації потрібно ввести в консоль: npm install express.

2. CORS (Cross-Origin Resource Sharing) - це бібліотека для Node.js, яка дозволяє налаштувати, які домени можуть робити запити до вашого сервера. Для її активації потрібно ввести в консоль: npm install cors

3. Mongoose - це об'єктно-документний мапер для MongoDB, який дозволяє зручно працювати з базою даних MongoDB в Node.js. Для його активації потрібно ввести в консоль: npm install mongoose.

4. Path - це вбудований модуль у Node.js, який надає інструменти для роботи з файловими шляхами.

5. Nodemailer — це бібліотека для відправки електронних листів з Node.js. Команда для активації - npm install nodemailer.

6. Bcryptjs — це бібліотека для хешування паролів в Node.js. Для активації вводимо: npm install bcryptjs [20].

7. Google Generative AI — це бібліотека для інтеграції генеративних моделей штучного інтелекту від Google, таких як Gemini. Команда для цієї бібліотеки - npm install @google/generative-ai.

8. dotenv — це бібліотека для завантаження змінних середовища з файлу .env у вашому проєкті. В цій бібліотеці можуть зберігатися дані адміністраторів, такі як паролі або секретні ключі. Для активації потрібно написати в консоль: npm install dotenv.

Після підключення даних бібліотек на сервері, обов'язково треба налаштувати статичні файли, які не змінюються і одразу виводяться користувачеві при запуску сервера. На рисунку 4.21 продемонстровано обробка статичних даних.

```
app.use(express.static("frontend/css"));
app.use(express.static("frontend/js"));
app.use(express.static("frontend/img"));
```

Рис. 4.21 Обробка статичних даних на сервері

Далі потрібно поставити головний файлом index.html, щоб коли запускався сервер, користувачу відкривалось саме ця сторінка. На рисунку 4.22 показана основна сторінка на сервері.

```
app.get("/", (req, res) => {
    res.sendFile(path.join(__dirname, '..', 'frontend', 'index.html'));
});
```

Рис. 4.22 Головна сторінка на сервері

4.4 Реалізація бази даних

Підключення до MongoDB виконується за допомогою mongoose.connect(), що дозволяє підключити сервер до локальної бази даних MongoDB, як показано на рисунку 4.24.

```
mongoose.connect("mongodb://localhost:27017/slimbody")
  .then(() => console.log("MongoDB підключено"))
  .catch((err) => console.log("Помилка підключення до MongoDB:", err));
```

Рис. 4.24 Підключення до база даних

В базі даних створюється 2 моделі - це Food та User. На рисунку 4.25 та на рисунку 4.26 показані моделі Food та User. Модель Food відповідає за зберігання інформації про продукти, зокрема їх назву, калорійність, кількість білків, вуглеводів та жирів. Вона має кілька полів: назва продукту, калорії на 100 г, вміст білка, вуглеводів, жирів, а також поле isCustom, яке визначає, чи є цей продукт спеціально доданим користувачем.

Модель User використовується для зберігання даних про користувачів. Вона включає такі поля, як email та password. Ця модель дозволяє користувачам реєструватися та зберігати свої дані для подальшого використання в системі.

Обидві моделі взаємопов'язані через поле createdBy в моделі Food, яке містить посилання на User. Це дозволяє визначити, який користувач додав певний продукт. Модель User служить для керування обліковими записами користувачів, зокрема для реєстрації та аутентифікації. Таким чином, система організована таким чином, щоб кожен продукт міг бути пов'язаний із конкретним користувачем, що забезпечує персоналізацію даних.

```
const foodSchema = new mongoose.Schema({
  name: { type: String, required: true },
  calories: { type: Number, required: true },
  protein: { type: Number, default: 0 },
  carbs: { type: Number, default: 0 },
  fat: { type: Number, default: 0 },
  isCustom: { type: Boolean, default: false },
  createdBy: { type: mongoose.Schema.Types.ObjectId, ref: 'User' }
}, { timestamps: true });

const Food = mongoose.model("Food", foodSchema);
```

Рис. 4.25 Модель Food

```
const userSchema = new mongoose.Schema({
 email: { type: String, required: true, unique: true },
 password: { type: String, required: true },
});

const User = mongoose.model("User", userSchema);
```

Рис. 4.26 Модель User

4.5 Тестування застосунка

Тестування - це процес перевірки та оцінювання програмного забезпечення для виявлення помилок, дефектів або невідповідностей вимогам. Тестування є важливою частиною розробки програмного забезпечення, оскільки воно дозволяє гарантувати, що продукт працює правильно, ефективно і відповідає вимогам користувача.

Тест-кейс (test case) - це документ, який описує конкретні умови та кроки, необхідні для тестування певної функціональності або компонента програмного забезпечення. Тест-кейс є основним інструментом у процесі тестування, оскільки він допомагає систематизувати перевірку програмного продукту, забезпечити точність і повторюваність тестів.

Проведено було 6 тест-кейси, з яких 6 пройшли з очікуваним результатом, що демонструє успішність тестування і відповідність вимогам програмного забезпечення.

Всі тест-кейси були оформлені у вигляді таблиці 4.1.

Таблиця 4.1

Тест-кейси

Номер	**Назва**	**Кроки**	**Очікуваний результат**	**Реальний результат**
1	Перевірка відкриття застосунку	1. Відкрити застосунку	Застосунок відкривається	Застосунок відкривається
2	Перевірка головної сторінки пошук рецептів	1. Відкрити застосунок 2. Відкрити головну сторінку 3. Вводить назву рецепта 4. Натискає кнопку «Шукати»	Відображається потрібний рецепт	Програма шукає рецепт
3	Перевірка поля реєстрації, отримання повідомлення	1. Відкриття застосунку 2. Натиснути на кнопку «Реєстрація» 3. Заповнити форму пошта та пароль	Відображення спливаючого вікна після реєстрації	Відображення спливаючого вікна після реєстрації
4	Перевірка сторінки вхід, перехід до аккаунта	1. Відкриття застосунку 2. Натиснути на кнопку «Вхід» 3. Заповнити форму пошта та пароль	Перехід до аккаунту користувача	Користувач переходить у свій акаунт
5	Перевірка сторінки розрахунок калорійності	1. Відкриття застосунку 2. Натиснути на кнопку «Розрахунок калорійності» 3. Ввести свої дані такі як: стать, зріст, вага, вік, активність і ціль	Автоматично рахує рекомендовану добову норму на день	Автоматично підраховує рекомендовану добову норму на день
6	Перевірка роботи AI помічника в акаунті користувача	1. Відкриття застосунку 2. Вхід в акаунт 3. Відкриття модального вікна AI помічника	Відповідає в чаті штучний інтелект та показує продукти за кнопками «Доступні продукти» та «Продукти для діабетиків»	Штучний інтелект спілкується з користувачем та показує продукти за кнопками «Доступні продукти» та «Продукти для діабетиків»

ВИСНОВКИ

Під час кваліфікаційної роботи було виконано усі поставлені задачі:

1. Проведено аналіз сучасних вимог до веб-застосунків, а саме: розглянуто вимоги до продуктивності, масштабованості, а також принципи UI/UX.

2. Проаналізовано існуючи щоденники дієти і визначено їхні переваги, серед недоліків виділено відсутність веб-платформи та відсутність штучного інтелекту.

3. Сформовано функціональні та нефункціональні вимоги до клієнт-серверного застосунку «Щоденник дієти». Серед основних вимог: можливість додавати продукти, рецепти і напої; розрахунок щоденної норми калорій; розрахунок індексу маси тіла; можливість пошуку продукту для зареєстрованих користувачів та рецепту для незареєстрованих користувачів; використання AI помічника для особистих питань та відображення всіх продуктів у web-застосунку, а також продуктів для діабетиків.

4. Досліджено інструменти та технології для розробки клієнт-серверного застосунку «Щоденник дієти», на підставі цього було обрано: HTML, CSS, JS для фронтенду, Node.js для бекенду, MongoDB для зберігання даних.

5. Розроблено web-застосунок на основі обраних технологій і з урахуванням зазначених вимог для щоденника дієти.

6. Проведено функціональне та нефункціональне тестування системи, яке показало, що застосунок відповідає зазначеним вимогам.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Вертій М.О. Шевченко С.М. Розробка web-застосунку «Щоденник дієти» з використанням фреймворків NODE.JS ТА EXPRESS // Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24.04.2025, ДУІКТ, Київ, Україна, С. 38-40.

2. Вертій М.О. Забезпечення конфіденційності інформації у web-застосунку «Щоденник дієти» // Матеріали XII Всеукраїнської науково-практичної конференції молодих учених «Інформаційні технології – 2025», 15.05.25, КСУ імені Бориса Грінченка, Київ, Україна, С. 267-268.

ПЕРЕЛІК ПОСИЛАНЬ

1. Шмаюн С.В., Жураковський Б.Ю., Іваніченко Є.В. (2022). Система підбору харчування за показниками здоров'я. *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*, *4*(16), 63–75. URL: https://doi.org/10.28925/2663-4023.2022.16.6375.

2. Лещенко О.Б, Хлюпіна А.С., Богдан Д.О. (2018). Веб-додаток для ведення щоденника харчування та тренування: вимоги, розроблення і впровадження. *Радіоелектронні і комп'ютерні системи, 3(87),* 49-63. URL: https://doi.org/10.32620/reks.2018.3.06.

3. Lakmal Meegahapola, Salvador Ruiz-Correa, and Daniel Gatica-Perez. 2020. Protecting Mobile Food Diaries from Getting too Personal. In 19th International Conference on Mobile and Ubiquitous Multimedia (MUM 2020), November 22–25, 2020, Essen, Germany. ACM, New York, NY, USA, 11 pages. URL: https://doi.org/10.1145/3428361.3428468.

4. Kong NA, Moy FM, Ong SH, Tahir GA, Loo CK. MyDietCam: Development and usability study of a food recognition integrated dietary monitoring smartphone application. DIGITAL HEALTH. 2023;9. URL: https://doi.org/10.32620/reks.2018.3.06.

5. Schneider K and Herforth A. Software tools for practical application of human nutrient requirements in food-based social science research [version 1; peer review: 1 approved with reservations, 1 not approved]. Gates Open Res 2020, 4:179 URL: https://doi.org/10.12688/gatesopenres.13207.1.

6. Tablycjakalorijnosti. URL: https://www.tablycjakalorijnosti.com.ua/tablytsya-yizhyi (дата звернення 14.05.2025).

7. MyFitnessPal. URL: https://play.google.com/store/apps/details?id=com.myfitnesspal.android&hl=uk (дата звернення 14.05.2025).

8. Lose It! URL: https://www.loseit.com/ (дата звернення 14.05.2025).

9. FatSecret. URL: https://www.fatsecret.com/ (дата звернення 14.05.2025).

10. Денис Завацький. IT-компанія повного циклу розробки програмних продуктів WEZOM - Київ, Україна. Що таке Node.js: визначення, особливості та застосування | Wezom. 28.06.2023. URL: https://wezom.com.ua/ua/blog/vse-chto-nuzhno-znat-o-nodejs.

11. Node.js — Run JavaScript Everywhere. URL: https://nodejs.org/uk (дата звернення 12.05.2025).

12. Express - Node.js web application framework. URL: https://expressjs.com (дата звернення 12.05.2025).

13. Ковальчук, В. М. Програмування серверних додатків за допомогою Node.js: підручник. — Львів: Видавництво Львівської політехніки, 2018. — 290 с.

14. Сидоренко, О. І. Розробка веб-додатків на Node.js: сучасні підходи та інструменти. — Київ: Либідь, 2017. — 250 с.

15. MongoDB. MongoDB: The World’s Leading Modern Database. URL: https://www.mongodb.com/ (дата звернення 10.05.2025).

16. Юлія Каграманова. Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти. 27 жовтня 2022, 12:00. URL: https://dou.ua/forums/topic/40575/.

17. Джейд Моралес. MindOnMap | Free Mind Mapping Tool to Draw Ideas Easily Online. What is A UML Component Diagram: Symbols and Tutorial. 10.03.2023. URL: https://www.mindonmap.com/uk/blog/uml-component-diagram/.

18. FoxmindEd. UML діаграми, їхні основні типи та процес розроблення. URL: https://foxminded.ua/uml-diagramy/.

19. Фіона Браун. Guru99. Модель діаграми зв’язків сутностей (ER) із прикладом СУБД. URL: https://www.guru99.com/uk/er-diagram-tutorial-dbms.html.

20. Бондаренко О., Ушкаленко І. (2017). Безпека web-додатків: актуальні проблеми та їх аналіз. *Формування ринкової економіки в Україні*. Вип. 38., 28-36.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка web-застосунку «Щоденник дієти»

Виконав студент 4 курсу
групи ПД-43
Вертій Максим Олександрович
Керівник роботи
к.п.н., доц., доцент кафедри ІПЗ Шевченко Світлана Миколаївна

Київ – 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – допомога користувачам відстежувати своє харчування та прогрес у досягненні дієтичних цілей.
- **Об'єкт дослідження** – процес контролю дієтичного харчування.
- **Предмет дослідження** – розробка web-застосунка «Щоденник дієти».

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Провести аналіз сучасних вимог до web-застосунків.
2. Проаналізувати існуючи щоденники дієти і визначити їхні переваги та недоліки.
3. Сформувати функціональні та нефункціональні вимоги до клієнт-серверного застосунку «Щоденник дієти».
4. Дослідити інструменти та технології для розробки клієнт-серверного застосунку «Щоденник дієти».
5. Розробрити клієнт-серверний застосунок на основі обраних технологій і з урахуванням зазначених вимог для «Щоденника дієти».
6. Провести тестування клієнт-серверного застосунку «Щоденник дієти».

3

АНАЛІЗ АНАЛОГІВ

Характеристика	Tablycjakal orijnosti	MyFitnessPal	Lose It!	FatSecret	Slim-Body
Наявність веб-платформи	+	-	+	+	+
Підрахунок калорій	+	+	+	+	+
Надання продуктів для діабетиків за допомогою ШІ.	-	-	-	+	+
Список рецептів	+	+	+	+	+
Покажчики калорійності продуктів	+	+	-	+	+
Особистий кабінет	+	+	+	-	+
Можливість додавати продукти вручну	+	+	-	-	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

1. Реєстрація та авторизація, розлогування.
2. Можливість користувачу додавати продукти, рецепти і напої.
3. Переглядати рецепти незареєстрованим користувачам.
4. Отримати підтвердження реєстрації через gmail.
5. Розрахунок щоденної норми калорій.
6. Розрахунок індекс маси тіла.
7. Можливість пошуку продукту для зареєстрованих користувачів та рецепту для не зареєстрованих користувачів.
8. Можливість використання AI помічника для особистих питань та відображення всіх продуктів у web-застосунку, а також продуктів для діабетиків.

Нефункціональні вимоги:

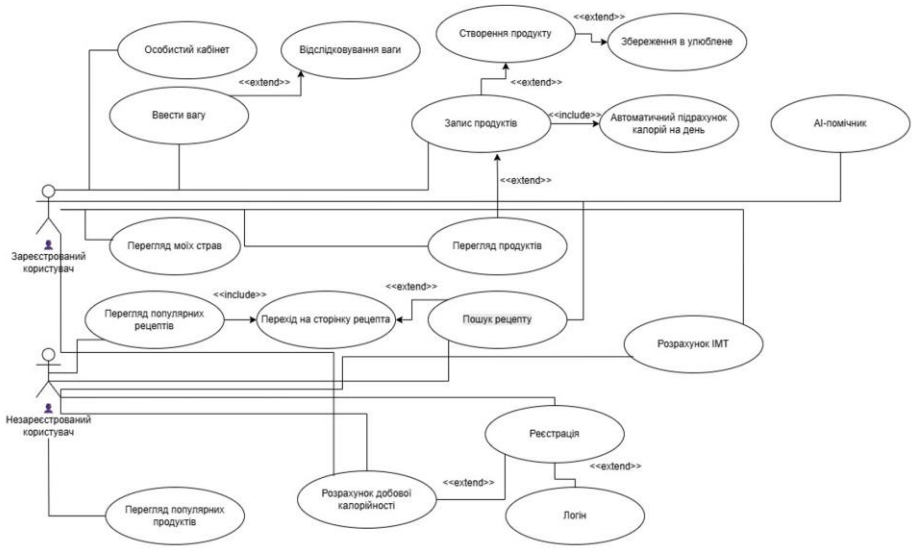
1. Додаток може працювати в різних браузерах: Chrome, Firefox, Opera, Internet Explorer
2. Здійснюється хешування паролів для безпеки користувачів (за допомогою bcrypt).
3. Зручний інтерфейс AI помічника та розрахунок калорій.
4. Завантаження сторінок на сайті до 2 секунд.

5

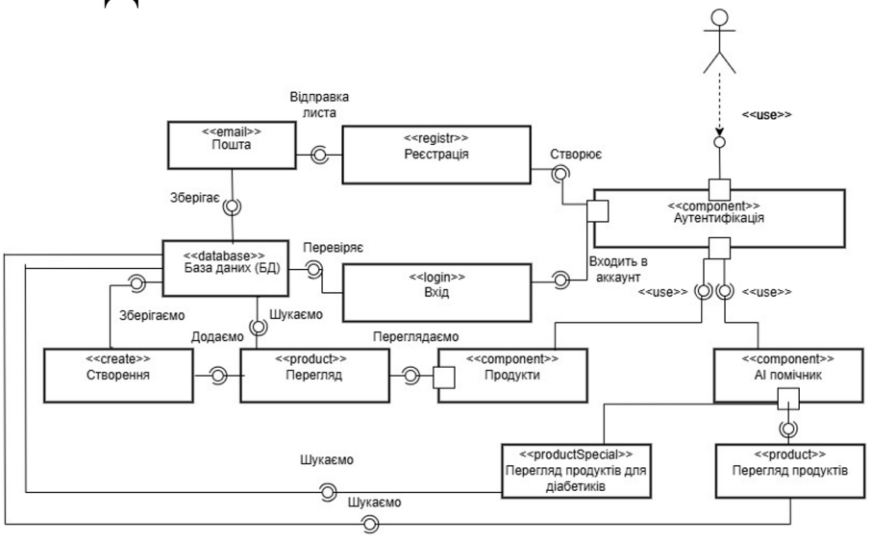
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

6

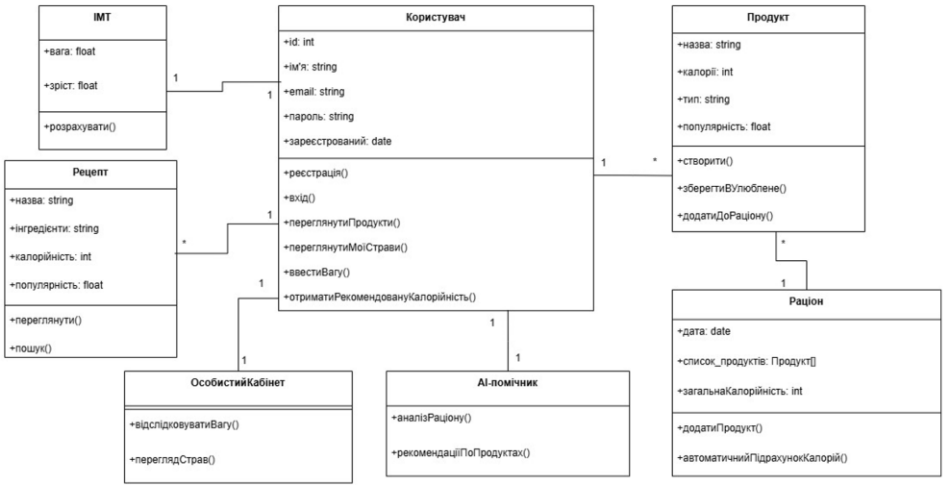
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



ДІАГРАМА КОМПОНЕНТІВ

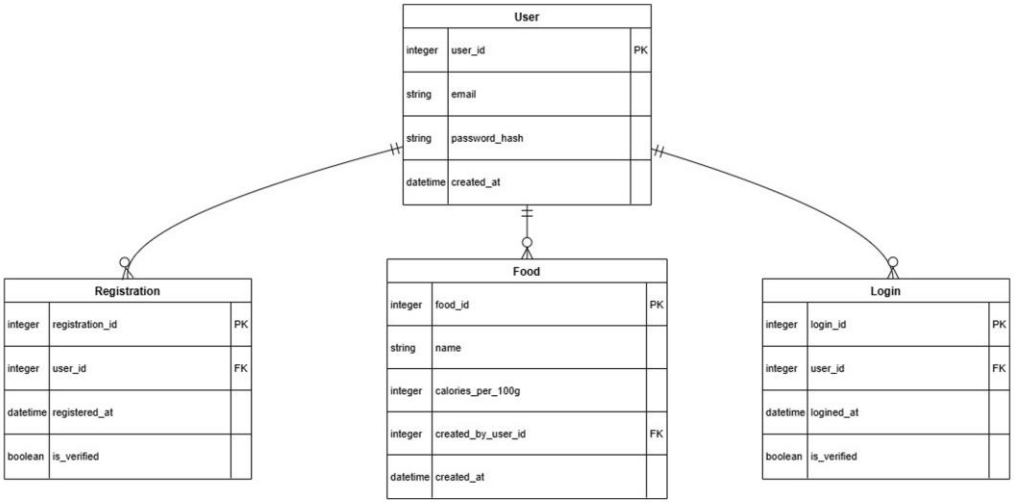


ДІАГРАМА КЛАСІВ



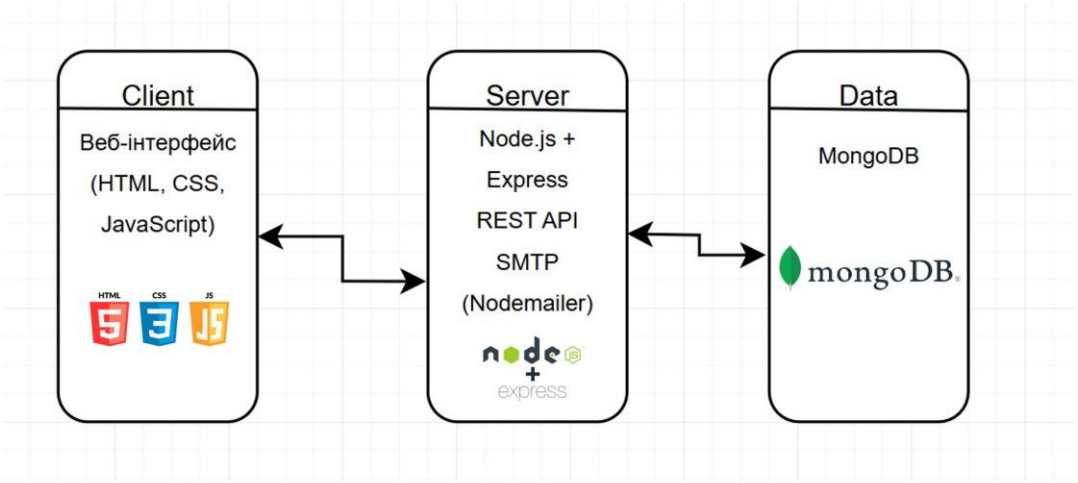
9

ER ДІАГРАМА



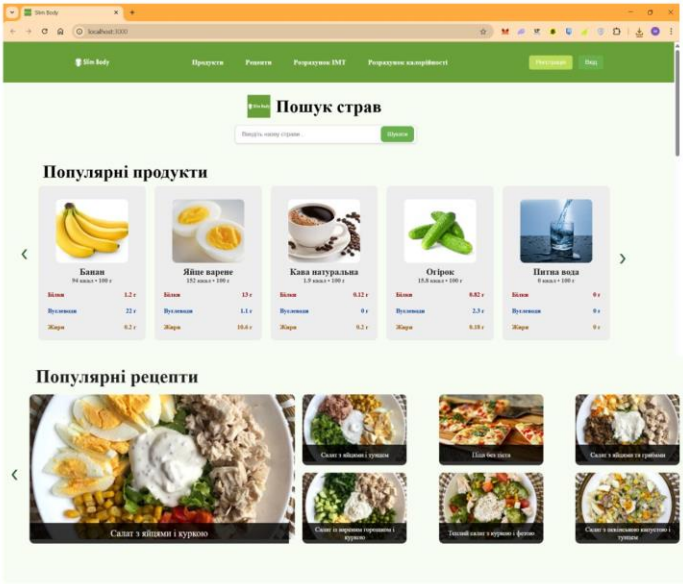
10

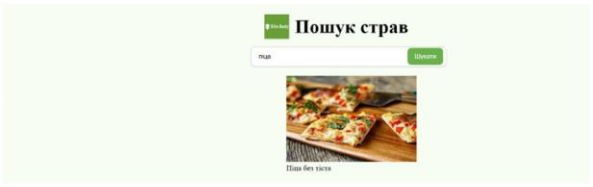
АРХІТЕКТУРА WEB-ЗАСТОСУНКУ



11

ЕКРАННІ ФОРМИ

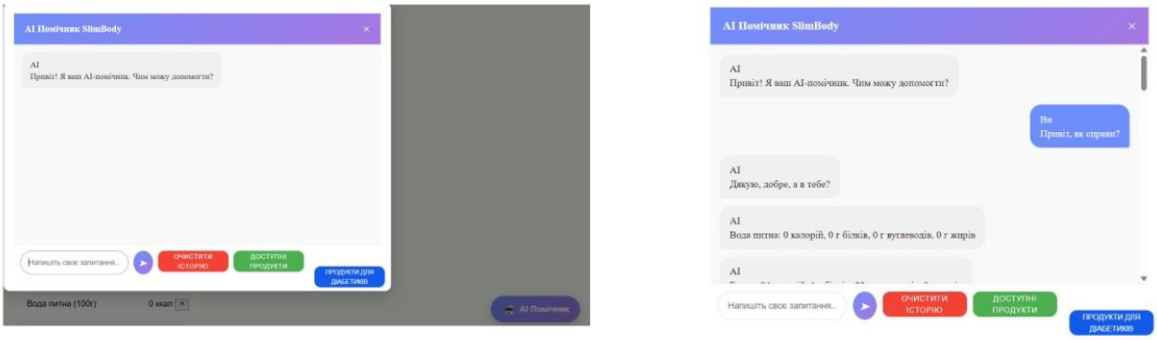




Головна сторінка

12

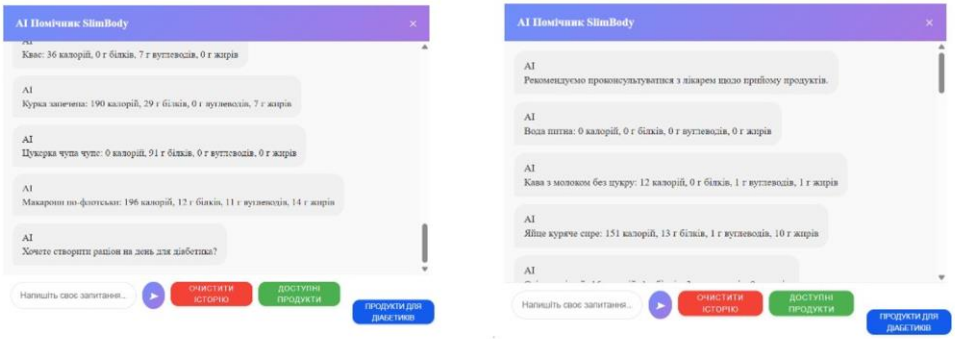
ЕКРАННІ ФОРМИ



AI помічник

18

ЕКРАННІ ФОРМИ



AI помічник

19

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Вертій М.О. Шевченко С.М. Розробка web-застосунку «Щоденник дієти» з використанням фреймворків NODE.JS ТА EXPRESS // Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24.04.25, ДУІКТ, Київ, Україна, с. 37.
2. Вертій М.О. Забезпечення конфіденційності інформації у web-застосунку «Щоденник дієти» // Матеріали XII Всеукраїнської науково-практичної конференції молодих учених «Інформаційні технології – 2025», 15.05.25, КСУ імені Бориса Грінченка, Київ, Україна, с. 267.

ВИСНОВКИ

1. Проведено аналіз сучасних вимог до веб-застосунків, а саме: розглянуто вимоги до продуктивності, масштабованості, а також принципи UI/UX.
2. Проаналізовано існуючи щоденники дієти і визначено їхні переваги, серед недоліків виділено відсутність веб-платформи та відсутність штучного інтелекту.
3. Сформовано функціональні та нефункціональні вимоги до клієнт-серверного застосунку «Щоденник дієти». Серед основних вимог: можливість додавати продукти, рецепти і напої; розрахунок щоденної норми калорій; розрахунок індексу маси тіла; можливість пошуку продукту для зареєстрованих користувачів та рецепту для незареєстрованих користувачів; використання AI помічника для особистих питань та відображення всіх продуктів у web-застосунку, а також продуктів для діабетиків.
4. Досліджено інструменти та технології для розробки клієнт-серверного застосунку «Щоденник дієти», на підставі цього було обрано: HTML, CSS, JS для фронтенду, Node.js для бекенду, MongoDB для зберігання даних.
5. Розроблено клієнт-серверний застосунок на основі обраних технологій і з урахуванням зазначених вимог для щоденника дієти.
6. Проведено функціональне та нефункціональне тестування системи, яке показало, що застосунок відповідає зазначеним вимогам.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Вихідний код файлу index.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Slim Body</title>
   <link rel="stylesheet" href="css/style.css">
   <link rel="icon" type="image/png" href="img/logo2.png">
   <link href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css" rel="stylesheet">
</head>
<body>
   <header>
      <div class="container">
         <div class="logo-container">
            <img src="img/logo2.png" alt="Slim Body Logo"
class="logo-img">
            <!-- <h1 class="name">Slim Body</h1> -->
         </div>
         <nav>
            <ul>
               <li><a href="login.html">Продукти</a></li>
               <li><a href="index.html" id="recipes-
link">Рецепти</a></li>
               <li><a href="mass.index.html">Розрахунок
ІМТ</a></li>
               <li><a href="date-calories.html">Розрахунок
калорійності</a></li>

 <div class="centered-image-container" id="centered-image-
container">
   <a id="centered-image-link" href="#">
      <img src="" alt="" id="centered-image" />
   </a>
</div>
<div class="search-results-container" id="search-
results"></div>
 <div class="carousel-title">Популярні продукти</div>
 <button class="scroll-btn-prev">❮</button>
 <div class="products-container">
   <div class="fs-item">
      <a class="fs-link" href="/stravy/banan">
         <img class="best-fs-image" src="img/banana.jpg"
alt="Банан">
         <div class="best-fs-title">Банан</div>
         <div class="best-fs-kcal-info">94 ккал • 100 г</div>
         <div class="best-fs-protein"
<button class="scroll-btn-next">❯</button>
  <div class="best-recipes-container">
   <div class="best-recipes-heading layout-align-center-center
layout-row" layout="row" layout-align="center center">
      <div class="carousel-title-2">Популярні рецепти</div>
      <button class="scroll-btn-prev-2">❮</button>

 <div flex="auto" style="max-width: 367px" class="flex-
auto"></div></div>
<div class="no-select">
   <div layout="row" layout-align="center center"
style="margin-top: 16px;" class="layout-align-center-center
layout-row">
      <div class="best-recipe">
            </ul>
         </nav>
         <div class="auth-buttons">
            <button class="register"> <a
href="register.html">Реєстрація</a></button>
            <button class="login"><a
href="login.html">Вхід</a></button>
         </div>
      </div>
   </header>
<div class="search">
  <div class="search-container">
   <img src="img/logo2.png" alt="Slim Body Logo"
class="search-logo">
   <div class="search-content">
    <div class="search-eat-container">
     <div class="search-eat">Пошук страв</div>
    </div>
    <div class="search-box-container">
     <div class="search-box">
      <input type="text" class="search-input"
placeholder="Введіть назву страви..." id="search-input">
       <button class="search-button" id="search-
button">Шукати</button>
      </div>
    </div>
   </div>
  </div>
</div>

         <div class="best-recipe-cards" id="best-recipe-cards"
ng-if="$mdMedia('md')">
            <!-- Пример для одного рецепта -->
            <div class="best-recipe-card first-card">
            <a href="/stravy/salat-z-yaytsyami-i-
kurkoyu">Салат з яйцями і куркою</a>
               <img class="best-recipe-image first-card-image"
                  onerror="this.onerror=null;
this.src='/resources/image/landingpage/graphics.webp';"
                  alt="Салат з яйцями і куркою"
                  title="Салат з яйцями і куркою"
                  src="img/recipe-1.jpg" width="163"
height="144">
               <div class="best-recipe-title2">Салат з яйцями
і куркою</div>
            </a>
            </div>
            <div class="best-recipe-card">
               <a href="/stravy/salat-z-yaytsyami-i-tuntsem">
                  <img class="best-recipe-image"
                     onerror="this.onerror=null;
this.src='/resources/image/landingpage/graphics.webp';"
                     alt="Салат з яйцями і тунцем"
                     title="Салат з яйцями і тунцем"
                     src="img/recipe-2.jpg">
                  <div class="best-recipe-title">Салат з яйцями і
тунцем</div>
               </a>
            </div>
            <div class="best-recipe-card">
               <a href="/stravy/pitsa-bez-tista">
                  <img class="best-recipe-image"
```

```
                    onerror="this.onerror=null;
this.src='/resources/image/landingpage/graphics.webp';"
                    alt="Піца без тіста"
                    title="Піца без тіста"
                    src="img/recipe-3.jpg" width="163"
height="144">
                <div class="best-recipe-title">Піца без
тіста</div>
              </a>
            </div>
            <div class="best-recipe-card">
              <a href="/stravy/salat-z-yaytsyami-ta-gribami">
                <img class="best-recipe-image"
                    onerror="this.onerror=null;
this.src='/resources/image/landingpage/graphics.webp';"
                    alt="Салат з яйцями та грибами"
                    title="Салат з яйцями та грибами"
                    src="img/recipe-4.jpg">
                <div class="best-recipe-title">Салат з яйцями
та грибами</div>
              </a>
            </div>
            <div class="best-recipe-card">
              <a href="/stravy/salat-iz-varenim-goroskom-i-
kurkoyu">
                <img class="best-recipe-image"
                    onerror="this.onerror=null;
this.src='/resources/image/landingpage/graphics.webp';"
                    alt="Салат із вареним горошком і куркою"
                    title="Салат із вареним горошком і
куркою"
                    src="img/recipe-5.jpg">
                <div class="best-recipe-title">Салат із вареним
горошком і куркою</div>
              </a>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>

    <div class="intro-image">
      <img src="img\about-me.png" alt="Healthy food and
fitness" />
    </div>
  </section>    </div>
</div>
  <footer>
    <p>© 2025 Slim Body. Усі права захищені.</p>
  </footer>
<script>
  document.getElementById('recipes-
link').addEventListener('click', function (e) {
    e.preventDefault();  // Забираємо стандартну поведінку
(перехід по лінку)
    const recipesSection = document.querySelector('.best-
recipes-container');  // Знаходимо карусель рецептів
    if (recipesSection) {
      recipesSection.scrollIntoView({ behavior: 'smooth'
});  // Прокручуємо до каруселі
    }
  });</script>
  <script src="js/script.js"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.1.2/sock
et.io.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</body>
</html>
```

Вихідний код файлу user.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>User</title>
  <link rel="stylesheet" href="css/style.css">
  <link rel="stylesheet" href="css/aiteh.css">
  <link rel="icon" type="image/png" href="img/logo2.png">
  <link href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css" rel="stylesheet">
  <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght
@400;600&display=swap" rel="stylesheet">
</head>
<body>
  <header>
    <div class="container">
      <div class="logo-container">
        <img src="img/logo2.png" alt="Slim Body Logo"
class="logo-img">
      </div>
      <nav>
        <ul>
          <li><a href="product.html">Продукти</a></li>
          <li><a href="index.html">Рецепти</a></li>
          <li><a href="mass.index.html">Розрахунок
ІМТ</a></li>
          <li><a href="date-calories.html">Розрахунок
калорійності</a></li>
        </ul>
      </nav>
      <div class="auth-buttons">
        <button class="logins"><a
href="login.html">Вихід</a></button>
      </div>
    </div>
  </header>
  <div class="nutrition-banner">
    <div class="banner-content">
      <div class="banner-text">
        <span class="highlight">Харчуйся розумно!</span>
Відчуй різницю з кожним прийомом їжі
      <div class="meal-section" data-meal="breakfast"
style="margin-bottom: 15px;">
        <div style="display: flex; justify-content: space-
between; align-items: center;
            background-color: white; border-radius: 8px;
padding: 8px 20px;
            color: #515151; height: 36px; width: 100%;
            box-sizing: border-box;">
          <span>Сніданок</span>
          <button class="add-meal-btn" data-meal="breakfast"
 <div class="meal-section" data-meal="lunch" style="margin-
bottom: 15px;">
  <div style="display: flex; justify-content: space-between;
align-items: center;
      background-color: white; border-radius: 8px; padding:
8px 20px;
      color: #515151; height: 36px; width: 100%;
      box-sizing: border-box;">
    <span>Обід</span>
    <button class="add-meal-btn" data-meal="lunch"
      style="background: none; border: none; color:
#689f38;
```

```
            font-size: 18px; cursor: pointer;" id="open-food-
btn">+</button>
    </div>
    <div class="lunch-items" style="background: white; border-
radius: 0 0 8px 8px; margin-top: -8px; padding: 10px 20px;">
     </div>
</div>
 <div class="meal-section" data-meal="dinner">
    <div style="display: flex; justify-content: space-between;
align-items: center;
            background-color: white; border-radius: 8px; padding:
8px 20px;
            color: #515151; height: 36px; width: 100%;
            box-sizing: border-box;">
       <span>Вечеря</span>
       <button class="add-meal-btn" data-meal="dinner"
              style="background: none; border: none; color:
#689f38;
            font-size: 18px; cursor: pointer;"id="open-food-
btn">+</button>
    </div>
    <div class="dinner-items" style="background: white; border-
radius: 0 0 8px 8px; margin-top: -8px; padding: 10px 20px;">
     </div>
</div>
</div>
</div>
 <div class="balance" style="width: 451px; background-color:
#f8fdf4; border-radius: 4px; padding: 8px 20px; box-sizing:
border-box; margin: 0 auto; font-family: Arial, sans-serif;">
    <div style="display: flex; align-items: center; height: 20px;
margin-bottom: 8px;">
      <img src="img/user-2.png" width="16" height="16"
style="margin-right: 8px;">
      <span style="font-weight: bold; font-size: 13px; color:
#515151;">Сьогоднішній баланс</span>
    </div>
    <div style="display: flex; flex-direction: column; gap:
8px;">
      <div style="display: flex; justify-content: space-between;
height: 20px; align-items: center;">
        <span style="font-size: 13px; color:
#515151;">Загальний обсяг споживчої енергії за
добу</span>
        <span id="daily-total-calories" style="font-size: 13px;
font-weight: 500; color: #515151;">0 ккал</span>
      </div>
      <div style="display: flex; justify-content: space-between;
height: 20px; align-items: center;">
        <span style="font-size: 13px; color: #515151;">Добова
норма калорій взято з (Калькулятора калорій)</span>
        <span id="caloriesDailyNorm" style="font-size: 13px;
font-weight: 500; color: #515151;">0 ккал</span>
      </div>
      <div style="border-top: 1px solid #eee; margin: 8px
0;"></div>
  <div style="display: flex; justify-content: space-between;
height: 20px; align-items: center;">
  <span style="font-size: 13px; color: #515151;">Сьогодні
уже спожито</span>
  <span id="consumedRow" style="font-size: 13px; font-
weight: 500; color: #515151;">
    <span id="todayConsumed">0</span> / <span
id="dailyNorm">0</span> ккал (<span
id="consumedPercent">0</span>)
  </span>
</div>
<div style="display: flex; justify-content: space-between;
height: 20px; align-items: center;">
  <span style="font-size: 13px; color: #515151;">Залишилося
спожити</span>
  <span id="remainingRow" style="font-size: 13px; font-
weight: 500; color: #515151;">0 / 0 ккал (100%)</span>
</div>
   </div>
</div>
 <div id="modalOverlay" style="display: none;">
   <div id="productModal" class="modal-content">
     <div class="product-header">
       <h3 id="modalProductName">Назва продукту</h3>
     </div>
     <div class="product-details">
       <div class="detail-row">
         <span>Кількість</span>
         <div class="quantity-input">
           <input type="number" min="1" value="1"
class="quantity" id="productQuantity">
           <span class="grams" id="productWeight">Г</span>
         </div>
       </div>
       <div class="detail-row">
         <span>Період харчування</span>
         <select class="meal-period" id="mealPeriod">
           <option value="breakfast">Сніданок</option>
           <option value="lunch">Обід</option>
           <option value="dinner">Вечеря</option>
         </select>
       </div>
       <div class="detail-row">
         <span>Дата</span>
         <input type="date" class="food-date" id="foodDate"
value="">
       </div>
     </div>

     <div class="modal-buttons">
       <button class="save-btn"
id="saveFoodBtn">Записати</button>
       <button class="cancel-btn"
id="closeModalBtn">Закрити</button>
     </div>
   </div>
  </div>
  <div class="modal-overlay" id="modalOverlay"
style="display: none;">
    <div class="add-product-modal" id="productModal">
     </div>
  </div>
<button id="open-ai-button" class="ai-button">
  <i class="ai-icon"></i> AI Помічник
</button>
 <div id="ai-modal" class="modal">
  <div class="modal-content">
      <div class="weight-dialog__body">
          <div class="weight-input-group">
              <label class="weight-input-label">Вага (кг)</label>
              <input type="number" class="weight-input-field"
id="weightValue" placeholder="Введіть вагу">
          </div>
          <div class="weight-input-group">
              <label class="weight-input-label">Дата</label>
              <input type="date" class="weight-input-field"
id="weightDate">
          </div>
      </div>
      <div class="weight-dialog__footer">
          <button class="weight-dialog__btn weight-dialog__btn-
-secondary" id="closeDialogBtn">Закрити</button>
```

```
        <button class="weight-dialog__btn weight-dialog__btn-
-primary" id="saveWeightBtn">Записати</button>
      </div>
   </div>
</div>
   <footer>
      <p>© 2025 Slim Body. Усі права захищені.</p>
   </footer>
<script src="js/calendar.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.1.2/sock
et.io.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</body>
</html>
```

Вихідний код файлу server.js

```
const express = require("express");
const cors = require("cors");
const mongoose = require("mongoose");
const path = require("path");
const nodemailer = require("nodemailer");
const bodyParser = require("body-parser");
const bcrypt = require('bcryptjs');
const { GoogleGenerativeAI } = require("@google/generative-
ai");
require('dotenv').config();
const app = express();
const PORT = 3000;
const genAI = new
GoogleGenerativeAI(process.env.API_KEY);
app.use(cors());
app.use(express.json());
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));
const calendarData = require("../backend/data-calendar.json");
app.get("/api/calendar", (req, res) => {
  res.json(calendarData);
});
app.get("/api/foods", async (req, res) => {
  try {
    const products = await Food.find();  // Отримуємо всі
продукти з бази
    res.json(products);  // Відправляємо їх в JSON форматі
  } catch (err) {
    res.status(500).json({ error: "Не вдалося завантажити
продукти" });
  }
});
app.post('/api/gemini', async (req, res) => {
  const { message } = req.body;
  if (!message) {
    return res.status(400).json({
      error: 'Порожнє повідомлення',
      status: 'bad_request'
    });
  }
  try {
    const model = genAI.getGenerativeModel({ model: "gemini-
1.5-flash" });
    const result = await model.generateContent(message);
    const response = await result.response;
    if (response && response.text) {
      const text = await response.text();
       return res.json({
        user_message: message,
        bot_reply: text,
        status: 'success'
      });
    } else {
      console.error('Не вдалося отримати текст від AI');
      return res.status(500).json({
        error: 'Не вдалося отримати відповідь від AI',
        status: 'no_response'
      });
    }
  } catch (error) {
    console.error('Помилка при запиті до API:', error.message);
    return res.status(500).json({
      error: 'Внутрішня помилка сервера',
      status: 'server_error',
      details: error.message
    });
  }
});
app.get("/stravy/:recipeName", (req, res) => {
    const recipeName = req.params.recipeName;
    const recipe = recipes.find(r =>
r.title.toLowerCase().replace(/\s+/g, '-') === recipeName);
    if (recipe) {
       res.render("recipe", { recipe }); // Отображаем рецепт в
шаблоне
    } else {
       res.status(404).send("Рецепт не знайдений.");
    }
});
const foodSchema = new mongoose.Schema({
  name: { type: String, required: true },
  calories: { type: Number, required: true },
  protein: { type: Number, default: 0 },
  carbs: { type: Number, default: 0 },
  fat: { type: Number, default: 0 },
  isCustom: { type: Boolean, default: false },
  createdBy: { type: mongoose.Schema.Types.ObjectId, ref:
'User' }
}, { timestamps: true });
const Food = mongoose.model("Food", foodSchema);
 const defaultFoods = [
  { name: "Вода питна", calories: 0 },
  { name: "Банан", calories: 94 },
  { name: "Кава з молоком без цукру", calories: 12 },
  { name: "Яйце куряче сире", calories: 151 },
  { name: "Огірок свіжий", calories: 16 },
  { name: "Грецький білий йогурт 0% жиру Milko", calories:
58 },
  { name: "Картопля варена", calories: 85 },
  { name: "Яблуко червоне", calories: 73 },
  { name: "Кава еспресо", calories: 4 },
  { name: "Молоко 1.5% жирності", calories: 47 }
];
 async function initializeFoods() {
  try {
    const count = await Food.countDocuments();
    if (count === 0) {
      const defaultFoodsWithUser = defaultFoods.map(food =>
({
        ...food,
        createdBy: "someUserId"
      }));
      await Food.insertMany(defaultFoodsWithUser);
      console.log("✅ Стандартні продукти додано до бази");
    }
  } catch (err) {
    console.error("❌ Помилка при ініціалізації продуктів:",
err);
  }
```

```
}
app.use(express.static("frontend/css"));
app.use(express.static("frontend/js"));
app.use(express.static("frontend/img"));
app.use(express.static(path.join(__dirname, '..', 'frontend')));
 app.get("/", (req, res) => {
    res.sendFile(path.join(__dirname, '..', 'frontend',
'index.html'));
});
 mongoose.connect("mongodb://localhost:27017/slimbody")
  .then(() => console.log("MongoDB підключено"))
  .catch((err) => console.log("Помилка підключення до
MongoDB:", err));
 const userSchema = new mongoose.Schema({
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
});
const User = mongoose.model("User", userSchema);
app.post("/api/register", async (req, res) => {
  const { email, password } = req.body;
  try {
    const existingUser = await User.findOne({ email: email });
    if (existingUser) {
      return res.status(400).json({ success: false, message:
"Користувач з таким email вже існує." });
    }
     const newUser = new User({ email, password });
    await newUser.save();
     const transporter = nodemailer.createTransport({
      service: "gmail",
      auth: {
        user: "maksimvertij416@gmail.com",
        pass: "lqne tjik hjfj hmkh",
      },
    });
    const mailOptions = {
      from: "maksimvertij416@gmail.com",
      to: email,
      subject: "Slim-Body",
      text: "Вітаємо ви пройшли реєстрацію тепер уввідійть у
свій профіль",
    };
     await transporter.sendMail(mailOptions);
    console.log('Лист успішно відправлено');
    return res.json({ success: true, message: "Реєстрація
успішна! Перевірте свій email." });
  } catch (err) {
    console.error('Помилка при реєстрації:', err);
    return res.status(500).json({ success: false, message:
"Виникла помилка при реєстрації." });
  }
});
 app.post("/api/login", async (req, res) => {
  const { email, password } = req.body;
  try {
     const existingUser = await User.findOne({ email: email });
    if (!existingUser) {
      return res.status(400).json({ success: false, message:
"Користувач не знайдений." });
    }
     if (existingUser.password !== password) {
      return res.status(400).json({ success: false, message:
"Невірний пароль." });
    }

     return res.json({ success: true, message: "Вхід успішний!"
});
  } catch (err) {
    console.error('Помилка при вході:', err);
    return res.status(500).json({ success: false, message:
"Сталася помилка при вході." });
  }
});
app.post('/update-profile', async (req, res) => {
  const { email, password } = req.body;
  try {
     const updatedUser = await User.findOneAndUpdate(
      { email: email },
      { $set: { email: email, password: password } },
      { new: true }
    );
    res.json({ message: 'Профіль оновлено', user: updatedUser
});
  } catch (error) {
    res.status(500).json({ message: 'Помилка при оновленні
профілю', error: error });
  }
});
 app.post('/api/foods', async (req, res) => {
  try {
     const newFood = new Food({
        name: req.body.name,
        calories: req.body.calories,
        mealType: req.body.mealType,
        date: req.body.date
      });
     const savedFood = await newFood.save();
     res.status(201).json(savedFood);
  } catch (error) {
     res.status(400).json({ error: error.message });
  }
});
const recipes = [
    {
       id: 1,
       title: "Салат з яйцями і куркою",
       img: "/img/recipe-1.jpg",
       ingredients: [
          "Яйце куряче варене - 2 шт",
          "Куряче філе на пару - 80 г",
          "Салатне листя - 10 г",
          "Перец червоний солодкий - 25 г",
          "Цукрова кукурудза консервована - 60 г",
          "Сметана 10% - 35 г",
          "Гірчиця французька в зернах - 7 г",
          "Сіль кухонна харчова - 1 г"
       ],
       instructions: "Яйця і курку попередньо відварити, дати
охолонути. Змішати сметану і гірчицю (додати трохи солі
та перцю). З'єднати всі інгредієнти та заправити соусом.",
       calories: 350
    },
    {
       id: 2,
       title: "Салат з яйцями і тунцем",
       img: "/img/recipe-2.jpg",
       ingredients: [
          "Яйця - 3 шт",
          "Тунець консервований - 150 г",
          "Салатне листя - 20 г",
          "Огірок свіжий - 30 г",
          "Томат червоний - 20 г",
          "Оливкова олія - 10 г",
          "Сіль та перець - за смаком"
       ],
       instructions: "Зварити яйця та нарізати їх кубиками.
З'єднати всі інгредієнти в мисці, заправити олією та
спеціями.",
```

```
    calories: 400
  },
{
    id: 3,
    title: "Піца без тіста",
    img: "/img/recipe-3.jpg",
    ingredients: [
      "Моцарела - 100 г",
      "Томатний соус - 2 ст. ложки",
      "Помідори - 2 шт",
      "Оливки чорні - 5 шт",
      "Базилік - кілька листочків",
      "Часник - 1 зубчик"
    ],
    instructions: "Розігріти духовку до 180°C. Викласти моцарелу, нарізані помідори та оливки на пекарський папір, запікати 15-20 хвилин. Після запікання додати базилік та часник.",
    calories: 250
  },
  {
    id: 4,
    title: "Салат з яйцями та грибами",
    img: "/img/recipe-4.jpg",
    ingredients: [
      "Яйця варені - 3 шт",
      "Гриби свіжі - 100 г",
      "Салатне листя - 20 г",
      "Цибуля ріпчаста - 30 г",
      "Майонез - 20 г",
      "Сіль, перець - за смаком"
    ],
    instructions: "Нарізати гриби і цибулю, підсмажити їх на олії. Відварити яйця, нарізати кубиками. З'єднати всі інгредієнти, заправити майонезом та спеціями.",
    calories: 450
  },
  {
    id: 5,
    title: "Салат із вареним горошком і куркою",
    img: "/img/recipe-5.jpg",
    ingredients: [
      "Куряче філе - 150 г",
      "Горох варений - 100 г",
      "Морква - 1 шт",
      "Огірок свіжий - 50 г",
      "Сметана 10% - 30 г",
      "Сіль, перець - за смаком"
    ],
    instructions: "Відварити куряче філе, нарізати його. Моркву натерти на тертці. З'єднати всі інгредієнти, додати сметану та спеції.",
    calories: 380
  },
    {
    id: 6,
    title: "Теплий салат з куркою і фетою",
    img: "/img/recipe-6.jpg",
    ingredients: [
      "Куряче філе - 150 г",
      "Фета - 50 г",
      "Огірок свіжий - 30 г",
      "Помідори червоні - 40 г",
      "Листя салату - 20 г",
      "Оливкова олія - 15 г",
      "Сіль, перець - за смаком"
    ],
    instructions: "Відварити куряче філе, нарізати його на смужки. Овочі нарізати та змішати з листям салату. Додати фету і заправити оливковою олією та спеціями.",
    calories: 320
  },
  {
    id: 7,
    title: "Салат з пекінською капустою і тунцем",
    img: "/img/recipe-7.jpg",
    ingredients: [
      "Пекінська капуста - 150 г",
      "Тунець консервований - 100 г",
      "Огірок свіжий - 50 г",
      "Морква - 1 шт",
      "Оливкова олія - 15 г",
      "Лимонний сік - 5 г",
      "Сіль, перець - за смаком"
    ],
  {
    id: 10,
    title: "Млинці вівсяні з яблуками та корицею",
    img: "/img/recipe-10.jpg",
    ingredients: [
      "Вівсянка - 100 г",
      "Яйце - 1 шт",
      "Молоко - 200 мл",
      "Яблуко - 1 шт",
      "Кориця - 1 ч. л.",
      "Мед - 1 ст. л.",
      "Сіль - щіпка"
    ],
    instructions: "Перемолоти вівсянку до порошкоподібної консистенції. Змішати з яйцем, молоком і медом. Нарізати яблуко та додати в тісто. Смажити млинці на сковороді з обох боків до золотистої скоринки. Посипати корицею перед подачею.",
    calories: 320
  },
  {
    id: 11,
    title: "Оладки яблучно-вівсяні",
    img: "/img/recipe-11.jpg",
    ingredients: [
      "Вівсянка - 100 г",
      "Яблуко - 1 шт",
      "Яйце - 1 шт",
      "Молоко - 100 мл",
      "Борошно - 50 г",
      "Розпушувач - 1 ч. л.",
      "Мед - 1 ст. л.",
      "Сіль - щіпка"
    ],
    instructions: "Перемолоти вівсянку до борошна. Змішати з яйцем, молоком і медом. Додати терте яблуко та борошно з розпушувачем. Смажити оладки на розігрітій сковороді з обох боків до золотистої скоринки.",
    calories: 280
  },
{
    id: 12,
    title: "Капустяний млинець",
    img: "/img/recipe-12.jpg",
    ingredients: [
      "Капуста - 200 г",
      "Яйце - 1 шт",
      "Борошно - 100 г",
      "Молоко - 150 мл",
      "Цибуля ріпчаста - 50 г",
      "Сіль, перець - за смаком",
```

```
        "Олія для смаження - 20 г"
      ], instructions: "Нарізати капусту та цибулю, обсмажити до м'якості. Змішати з борошном, молоком і яйцем. Смажити млинці на сковороді з обох боків до золотистої скоринки.",
      calories: 220
    }, {
      id: 13,
      title: "Салат з куркою та фетою",
      img: "/img/recipe-13.jpg",
      ingredients: [
        "Куряче філе - 150 г",
        "Фета - 50 г",
        "Листя салату - 30 г",
        "Огірок свіжий - 50 г",
        "Оливкова олія - 20 г",
        "Лимонний сік - 5 г",
        "Сіль, перець - за смаком"
      ],
      instructions: "Відварити куряче філе, нарізати. Нарізати огірок та фету. Змішати всі інгредієнти, заправити олією та лимонним соком.",
      calories: 350
    },{id: 14,
      title: "Стручкова квасоля з грибами та вершковим соусом",
      img: "/img/recipe-14.jpg",
      ingredients: [
        "Стручкова квасоля - 200 г",
        "Гриби свіжі - 100 г",
        "Вершки 10% - 100 мл",
        "Часник - 1 зубчик",
        "Оливкова олія - 15 г",
        "Сіль, перець - за смаком",
        "Лимонний сік - 5 г"
      ],
      instructions: "Почистити і нарізати гриби. Обсмажити їх на оливковій олії з часником. Додати стручкову квасолю, смажити ще 5 хвилин. Додати вершки, сіль, перець і тушкувати до м'якості. Перед подачею полити лимонним соком.",
      calories: 300
    }
  ];
app.get("/api/products", (req, res) => {
    try {
        res.json(products);
    } catch (error) {
        console.error("Ошибка при загрузке продуктов из JSON:", error);
        res.status(500).json({ error: "Не удалось загрузить продукты" });
    }
});
app.get("/api/recipes", (req, res) => {
  res.json(recipes);
});
 app.get("/api/foods", async (req, res) => {
  try {
    const products = await Food.find();
    res.json(products);
  } catch (err) {
    console.error("Помилка при отриманні продуктів:", err);
    res.status(500).json({ error: "Не вдалося завантажити продукти" });
  }
});
 app.listen(PORT, async () => {
  console.log(`\n🔷 Запуск сервера...`);
  try {
     await initializeFoods();
    console.log(`✅ Сервер працює на http://localhost:${PORT}`);
    console.log(`🔷 Стандартні продукти перевірено та готові до використання`);
  } catch (err) {
    console.error("❌ Помилка при запуску сервера:", err);
    process.exit(1);
  }
})
```