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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка персоналізованого планеру калорій на основі меню кухонь світу з використанням штучного інтелекту»

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

_____ Сергій ШАПОВАЛ
(підпис)

Виконав: здобувач вищої освіти групи ПД-42

Сергій ШАПОВАЛ

Керівник: Андрій КОЛОДЮК
асистент кафедри ІПЗ

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

Шаповал Сергію Олександровичу

1. Тема кваліфікаційної роботи: «Розробка персоналізованого планеру калорій на основі меню кухонь світу з використанням штучного інтелекту»
керівник кваліфікаційної роботи асистент кафедри ІПЗ Андрій КОЛОДЮК,
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 про методи генерації рецептів за допомогою штучного інтелекту, принципи розрахунку харчової цінності продуктів (КБЖВ), документація API-сервісів для отримання харчових даних (Nutritionix) та генерації текстів (Gemini API), а також технічна документація щодо застосування технологій React, Express і Supabase у розробці вебзастосунків.

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

1. Огляд та аналіз існуючих технологій генерації текстового контенту за допомогою штучного інтелекту

2. Проєктування вебзастосунку для персонального планування харчування на основі цілей користувача та кухонь світу.
3. Програмна реалізація та опис функціонування застосунку для генерації AI-рецептів, розрахунку КБЖВ та збереження даних у Supabase.
4. Тестування функціональності застосунку та перевірка відповідності поставленим вимогам.

5. Перелік графічного матеріалу: *презентація*
1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Діаграма класів.
6. Діаграма діяльності.
7. Діаграма компонентів
8. Екранні форми.
9. Демонстрація роботи застосунку
10. 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.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-12.03.2025	
3	Огляд сучасних підходів до контролю харчування, обліку КБЖВ	13.03-18.03.2025	
4	Проектування веб-застосунку для планування харчування	19.03-23.03.2025	
5	Програмна реалізація веб-застосунку	24.03-24.04.2025	
6	Тестування застосунку	25.04-29.04.2025	
7	Оформлення роботи: вступ, висновки, реферат	30.04-11.05.2025	
8	Розробка демонстраційних матеріалів	12.05-18.05.2025	
9	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Сергій ШАПОВАЛ
(підпис)

Керівник
кваліфікаційної роботи _____ Андрій КОЛОДЮК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 50 стор., 1 табл., 20 рис., 17 джерел.

Мета роботи – спрощення процесу планування персоналізованого раціону харчування користувача.

Об'єкт дослідження – процес підбору продуктів харчування з урахуванням їх харчової цінності та кулінарних вподобань користувача.

Предмет дослідження – веб-застосунок для підбору та рекомендації страв різних кухонь світу з урахуванням харчової цінності продуктів і особистих переваг користувача за допомогою методів штучного інтелекту.

Короткий зміст роботи: В роботі проаналізовано сучасні підходи до персоналізованого планування харчування з використанням штучного інтелекту. Розглянуто інструменти для генерації текстового контенту (Gemini AI) та API-сервіси для отримання харчової цінності продуктів (Nutritionix). Розроблено архітектуру вебзастосунку та програмно реалізовано його ключові функції, зокрема: генерацію AI-рецептів на основі цілей користувача, розрахунок КБЖВ (калорій, білків, жирів, вуглеводів), збереження улюблених комбінацій продуктів, ведення вимірів тіла та перегляд згенерованих рецептів. Проведено модульне тестування функціональних блоків та перевірку взаємодії з базою даних Supabase. Інтерфейс реалізовано за допомогою React з використанням Tailwind CSS та бібліотеки shadcn/ui.

Результати роботи можуть бути застосовані для підтримки здорового способу життя, самостійного харчового планування та контролю за динамікою фізіологічних показників.

КЛЮЧОВІ СЛОВА: PLANNER-CALORIE, AI-РЕЦЕПТИ, REACT, SUPABASE, КБЖВ, ХАРЧОВЕ ПЛАНУВАННЯ, GEMINI AI, NUTRITIONIX, JWT

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI – Artificial Intelligence

API – Application Programming Interface

JWT – JSON Web Token

CRUD – Create, Read, Update, Delete

SPA – Single Page Application

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

JS – JavaScript

TS – TypeScript

React – JavaScript-бібліотека для створення інтерфейсів користувача

Express – Фреймворк для створення серверної частини на Node.js

Supabase – Хмарна платформа для зберігання даних і авторизації (на основі PostgreSQL)

BaaS – Backend-as-a-Service

КБЖВ – Калорії, Білки, Жири, Вуглеводи

JSON – JavaScript Object Notation

REST – Representational State Transfer

UI – User Interface

UX – User Experience

СУБД – Система управління базами даних

ВСТУП

Актуальність: Збереження фізичного здоров’я, правильне харчування та усвідомлений підхід до складання раціону набувають дедалі більшого значення в умовах сучасного способу життя. Завдяки розвитку цифрових технологій користувачі отримали можливість контролювати свої харчові звички за допомогою мобільних і вебзастосунків. Такі сервіси дозволяють самостійно планувати харчування, відстежувати споживання калорій, білків, жирів, вуглеводів (КБЖВ), а також формувати рецепти відповідно до індивідуальних цілей і вподобань. Особливо актуальним є використання штучного інтелекту для автоматичної генерації страв із врахуванням дієтичних обмежень або вподобань користувача.

Вебплатформа виступає зручним середовищем, яке об’єднує всі необхідні інструменти: облік продуктів, розрахунок КБЖВ, планування прийомів їжі, перегляд аналітики змін, ведення персональних вимірів (вага, зріст), а також збереження улюблених рецептів. Завдяки інтеграції з зовнішніми API (Gemini, Nutritionix), користувач отримує якісний функціонал в реальному часі без потреби в ручному введенні складної інформації.

Об’єкт дослідження є процес підбору продуктів харчування з урахуванням їх харчової цінності та кулінарних вподобань користувача.

Предмет дослідження є веб-застосунок для підбору та рекомендації страв різних кухонь світу з урахуванням харчової цінності продуктів і особистих переваг користувача за допомогою методів штучного інтелекту.

Метою роботи є спрощення процесу планування персоналізованого раціону харчування користувача.

Методи дослідження: На першому етапі було проаналізовано сучасні рішення у сфері харчових планерів та трекерів з метою визначення основних функціональних і нефункціональних вимог. Далі було обрано технологічний стек: React — для реалізації клієнтської частини, Express.js — для побудови серверної логіки, Supabase — як хмарну платформу для зберігання даних та автентифікації,

Gemini AI — для генерації AI-рецептів, Nutritionix — для розрахунку харчової цінності продуктів, а також Tailwind CSS і shadcn/ui — для побудови зручного й адаптивного інтерфейсу. Дослідження реалізації застосунку відбувалося на практиці під час розробки MVP (мінімального життєздатного продукту).

Наукова новизна: Унікальність розробленого застосунку полягає в тому, що він об’єднує кілька важливих функцій: автоматичну генерацію рецептів за допомогою штучного інтелекту, розрахунок КБЖВ для кожної страви, ведення особистих фізіологічних показників користувача (ваги, зросту), збереження улюблених комбінацій продуктів та аналіз змін у харчуванні. Застосунок забезпечує персоналізований підхід до харчування.

Практична значущість: Розроблений вебзастосунок може бути використаний широким колом користувачів для зручного й ефективного складання раціону, контролю харчових звичок та покращення фізичного самопочуття. Платформа є кросплатформною і доступна з більшості сучасних пристроїв через браузер

1 АНАЛІЗ ЗАСТОСУНКУ ДЛЯ ПЛАНУВАННЯ ХАРЧУВАННЯ

1.1 Веб-застосунки для планування харчування

Вебзастосунок для планування харчування — це спеціалізована онлайн-платформа, яка дозволяє користувачам контролювати раціон, підраховувати калорії, складати плани харчування відповідно до цілей і переваг, а також відстежувати фізіологічні показники. Такі застосунки надають можливість вести здоровий спосіб життя, керувати дієтами, слідкувати за динамікою ваги та формувати корисні харчові звички.

Основна мета подібних вебплатформ — зробити процес харчування більш усвідомленим, персоналізованим та ефективним. Завдяки сучасним інструментам (зокрема API для розрахунку харчової цінності та засобам штучного інтелекту[5]), користувачі можуть автоматизувати рутинні дії, отримуючи точні й персоналізовані рекомендації.

Основні компоненти вебзастосунків для харчового планування:

– Профіль користувача: збереження персональних даних (вік, вага, зріст, цілі).

– База продуктів: доступ до великої кількості продуктів із зазначеними калоріями, білками, жирами, вуглеводами.

– Генерація рецептів: створення страв на основі заданих критеріїв (дієта, кухня, алергени).

– Облік КБЖВ: автоматичне підрахування калорійності кожного прийому їжі[2].

– Візуальна аналітика: графіки змін ваги, макронутрієнтів, активності.

– Синхронізація з пристроями: інтеграція з фітнес-трекерами або календарем.

Цілі подібних систем:

– створення здорового раціону;

– персоналізація харчування;

– контроль дотримання дієти;

– допомога у досягненні цілей (схуднення, набір ваги тощо);

– формування звичок здорового способу життя.

– Переваги вебплатформ для харчового планування:

– Доступність: використання з будь-якого пристрою з доступом до Інтернету.

– Гнучкість: можливість налаштувати харчування відповідно до способу життя.

– Автоматизація: підрахунок калорій[3] і складання раціону без ручного введення формул.

– Персоналізація: адаптація плану до фізіологічних параметрів і цілей.

– Аналітика: візуалізація прогресу та показників у зрозумілій формі.

– Недоліки подібних сервісів:

– Залежність від Інтернету: для повної функціональності потрібне стабільне з'єднання.

– Не всі сервіси безкоштовні: багато функцій доступні лише за підпискою.

– Надлишок функцій: для початківців інтерфейс може бути перевантаженим.

– Обмежена база локальних продуктів: часто потрібен англомовний пошук.

– Відсутність AI: багато сервісів не підтримують генерацію рецептів за допомогою ШІ.

1.2 Специфіка персоналізованого планування харчування

Планування харчування має низку особливостей, що відрізняють його від звичайного підрахунку калорій або спонтанного вибору страв. У сучасному підході

велика увага приділяється персоналізації, аналітиці та інтеграції з харчовими базами даних, що дозволяє адаптувати раціон до індивідуальних цілей користувача. Нижче розглянуто ключові аспекти.

Роль фізіологічних параметрів:

– Вік, стать, вага, зріст та рівень фізичної активності мають прямий вплив на добову потребу в калоріях і нутрієнтах.

– Платформа повинна враховувати ці параметри автоматично, щоб забезпечити коректні рекомендації.

– КБЖВ – основа раціону

– Розрахунок калорій, білків, жирів, вуглеводів (КБЖВ) є ключовим для формування збалансованого плану.

– Похибка в підрахунках або невірне визначення порції може суттєво вплинути на результат.

Штучний інтелект у складанні меню

На відміну від статичних баз даних, ШІ дозволяє генерувати рецепти динамічно — під конкретні цілі користувача (наприклад: тип кухні, список інгрідієнтів).

Це дає змогу формувати унікальні страви, які не зберігаються в готових бібліотеках.

Кухні світу та дієтичні обмеження

– Люди обирають страви не лише за калорійністю, а й за культурними або релігійними вподобаннями.

– Програма має враховувати тип кухні (японська, середземноморська, українська тощо), алергени, тип дієти (веганська, безглютенова тощо).

– Аналітика та зворотний зв'язок

– Для формування звички користувач має бачити свій прогрес у зміні КБЖВ, ваги, частоті використання певних продуктів.

– Візуалізація даних допомагає утримати мотивацію та адаптувати план харчування у динаміці.

1.3 Цільова аудиторія

Застосунок Planner-Calorie орієнтований на широке коло користувачів, які мають різні цілі щодо харчування, здоров’я та способу життя.

Основні категорії користувачів включають:

– Люди, які контролюють вагу — користувачі, які прагнуть схуднути або набрати масу тіла, відстежуючи калорійність раціону, склад страв та прогрес змін. Їх мета — ефективне досягнення фізіологічних цілей за допомогою персоналізованого плану харчування.

– Користувачі, які дотримуються спеціальних дієт — вегани, люди з харчовими алергіями, непереносністю глютену або лактози. Вони потребують рецепти та меню, які відповідають конкретним дієтичним обмеженням.

– Люди, які прагнуть здорового способу життя — користувачі, які не мають специфічних обмежень, але хочуть покращити якість харчування, вести облік КБЖВ і зберігати здоровий баланс нутрієнтів у щоденному раціоні.

– Активні користувачі та спортсмени — ті, хто займається фітнесом або спортом і хоче слідкувати за точним споживанням білків, жирів та вуглеводів для досягнення оптимальної продуктивності та відновлення.

1.4 Дослідження існуючих аналогів

У процесі створення власного застосунку Planner-Calorie було здійснено глибокий аналіз популярних цифрових рішень у сфері планування харчування, відстеження нутрієнтів та персоналізованого підходу до здорового способу життя. Сучасний ринок програмного забезпечення пропонує користувачам велику кількість застосунків, які, так чи інакше, дозволяють контролювати свій раціон, аналізувати споживання калорій, формувати страви та відслідковувати зміни у масі тіла.

Серед широкого переліку наявних інструментів, найбільш помітними стали такі продукти як MyFitnessPal, Samsung Food та MealPrepPro — кожен з яких має

свої сильні сторони, певні недоліки та обмеження, що стали відправною точкою для проектування функціоналу унікального програмного рішення.

1.4.1 MyFitnessPal

MyFitnessPal — багатофункціональний застосунок, який дозволяє вести харчовий щоденник, відслідковувати КБЖВ, інтегрувати фізичну активність і формувати базове уявлення про щоденне споживання їжі. Сервіс має багаторічну репутацію та підтримку широкої спільноти користувачів.

Переваги:

– Найбільша база продуктів у світі: Сотні тисяч продуктів і готових страв уже внесені до бази, що значно спрощує введення даних. Присутні як міжнародні, так і локальні бренди.

– Сканування штрихкодів: Це значно економить час — користувач просто сканує упаковку, і застосунок автоматично заповнює всі КБЖВ.

– Інтеграція з фітнес-пристроями: MyFitnessPal підтримує з’єднання з Garmin, Fitbit, Apple Watch тощо, дозволяючи коригувати харчові цілі відповідно до фізичної активності.

– Гнучка система цілей: Користувач може вручну задати бажану вагу, швидкість змін і отримає персональний план споживання калорій.

– Аналітика та статистика: Застосунок візуалізує динаміку ваги, розподіл макронутрієнтів, середнє споживання — усе зручно, у вигляді графіків.

Недоліки:

– Платні функції в Premium-версії: Більшість цікавих можливостей (наприклад, облік макроелементів по прийомах їжі, розширена аналітика, заміна продуктів тощо) доступні лише у платному тарифі.

– Відсутність інтелектуальної генерації страв: Застосунок не пропонує рецепти автоматично. Усі комбінації продуктів і страв користувач повинен створювати вручну.

– Застарілий інтерфейс: Незважаючи на регулярні оновлення, інтерфейс виглядає перевантаженим — новачки можуть відчувати складність при першому використанні.

– Мінімальна підтримка локальних кухонь: Хоч база велика, відсутні функції фільтрації рецептів за національними кухнями (українська, японська тощо).

– Немає персонального підбору страв: Користувач самостійно шукає та додає їжу — штучний інтелект чи рекомендаційна система не використовується.

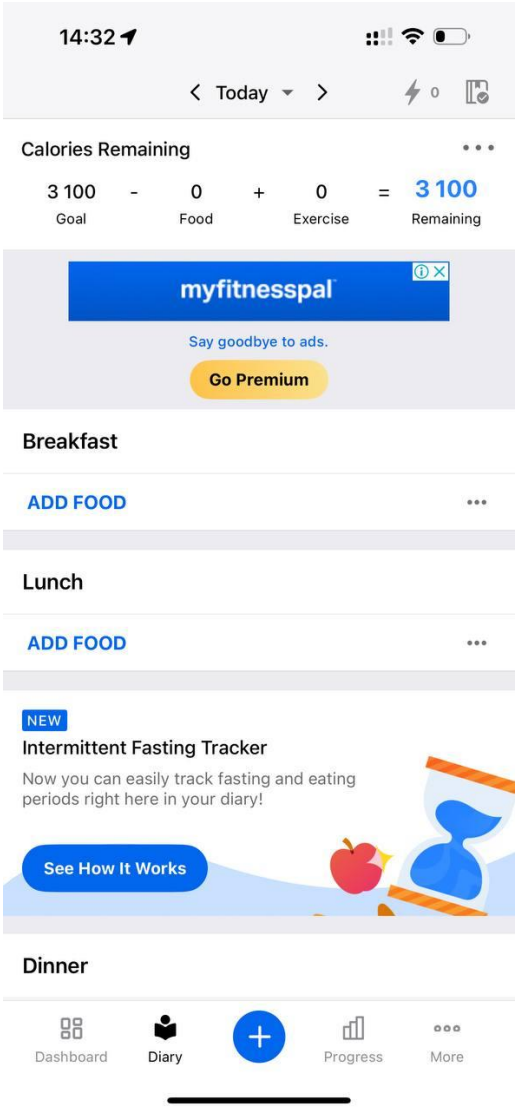


Рис. 1.1 Приклад додання страв до щоденного раціону

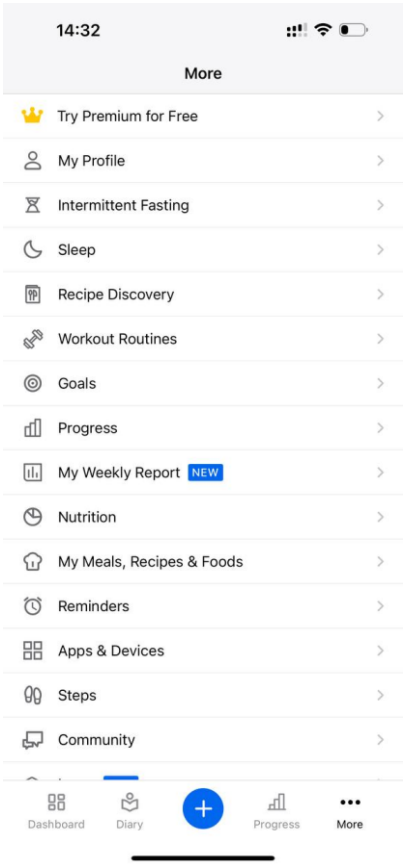


Рис. 1.2 Всі доступні функції в додатку

1.4.2 Samsung Food

Samsung Food — інструмент для зручного збереження рецептів, формування списків покупок і складання меню на тиждень. Він є частиною екосистеми Samsung і орієнтований передусім на користувачів побутової техніки цієї компанії, хоча доступний усім охочим.

Переваги:

– Сучасний, зручний інтерфейс: Мінімалістичний дизайн, легка навігація, адаптація під мобільні пристрої — усе це створює комфортне враження від використання.

– Бібліотека з понад 160 тис. рецептів: Рецепти структуровані, мають фото, опис кроків, список інгредієнтів та позначення дієтичних властивостей.

– Підтримка дієт та фільтрів: Користувач може фільтрувати страви за типом дієти (вегетаріанська, кето, безглютенова), складом або алергенами.

– Формування списку покупок: Обрані рецепти автоматично додаються до списку продуктів, який можна використовувати в магазині.

– Інтеграція з технікою Samsung: Наприклад, можна надіслати рецепт прямо на "розумну" плиту або холодильник.

Недоліки:

– Обмежені можливості персоналізації КБЖВ: Застосунок не дозволяє встановити свої калорійні цілі або бачити макронутрієнти у динаміці.

– Відсутність AI-генерації страв: Весь контент уже створений — користувач не може скомпонувати страву з доступних інгредієнтів під свої параметри.

– Відсутність індивідуальної аналітики: У системі немає розділу з ваговим прогресом, статистикою чи відображенням динаміки харчування.

– Часткова підтримка української мови: Частина інтерфейсу перекладена, але рецепти, інгредієнти та діалоги часто залишаються англійською.

– Орієнтація на готування, а не на здоров’я: Сервіс більше підходить тим, хто шукає цікаві рецепти, ніж тим, хто слідкує за нутрієнтами.

Рис. 1.3 Приклад збереження рецепту з різних сайтів

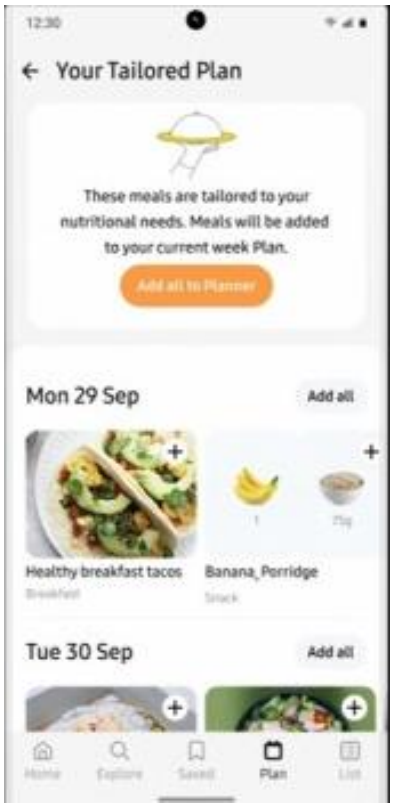


Рис. 1.4 Створення щоденного раціону

1.4.3 MealPrepPro

MealPrepPro — застосунок для тих, хто хоче структурувати своє харчування по днях і прийомах їжі. Особливу увагу сервіс приділяє спортсменам, людям на дієті та зайнятим користувачам, які хочуть заздалегідь підготувати їжу на тиждень.

Переваги:

– Готові плани харчування: Сервіс пропонує тижневі меню з рецептами, враховуючи мету (схуднення, набір м’язової маси, підтримка ваги).

– Класифікація за типами дієт: Наприклад, можна обрати кето, веганську або безмолочну дієту — і отримати повний план на день.

– Зручний тайм-менеджмент: Усі рецепти оптимізовані за часом приготування, що дозволяє швидко готувати на декілька днів наперед.

– Розрахунок макронутрієнтів: Кожна страва має повний КБЖВ-аналіз, і користувач бачить свій добовий баланс.

– Висока якість вмісту: Рецепти красиво оформлені, мають покрокові інструкції та гарні фотографії.

Недоліки:

– Закритий контент — лише з підпискою: Безплатна версія майже не функціональна. Щоб отримати доступ до рецептів, потрібна щомісячна оплата.

– Немає можливості генерувати рецепти: Усі страви вже створені, і не можна змінювати інгредієнти або комбінувати нові.

– Обмежена база продуктів: Немає гнучкого пошуку по інгредієнтах чи локальних продуктах.

– Відсутність обліку фізіологічних параметрів: Немає модуля для зберігання ваги, зросту або динаміки змін — лише харчування.

– Немає повноцінної візуальної аналітики: Користувач не бачить історію змін або графіків — тільки поточні значення.

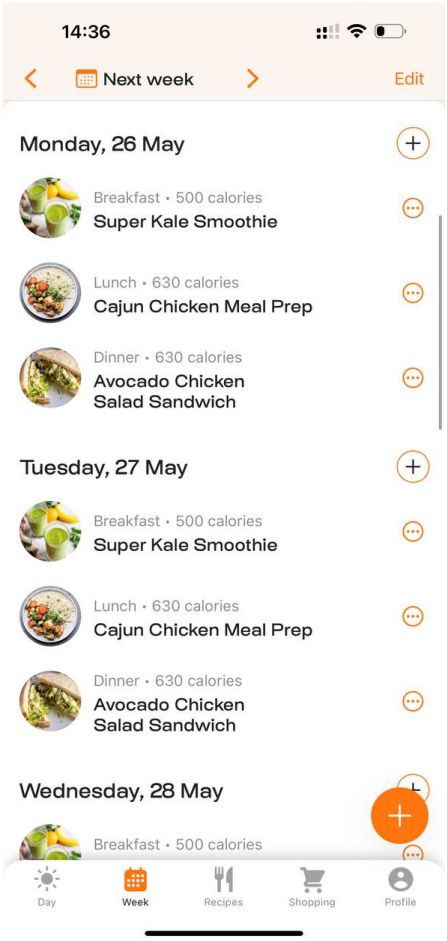


Рис. 1.4 Створення щоденного раціону

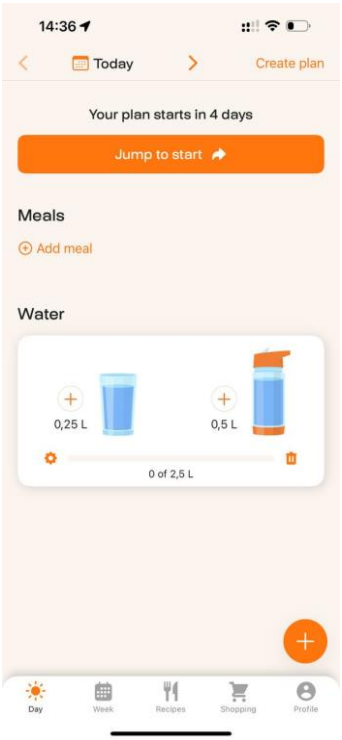


Рис. 1.4 Основне меню додатку

В таблиці 1.1 зведено результати аналізу існуючих застосунків для вивчення японської мови та їх порівняння.

Таблиця 1.1

Зведена таблиця аналізу існуючих застосунків для планування харчування та їх порівняння

Показник	**MyFitnessPal**	**Samsung Food**	**MealPrepPro**
Планування раціону	Харчовий щоденник, введення вручну або сканером	Меню на тиждень за готовими рецептами	Автоматизований тижневий план харчування
Рецепти	Відсутні AI-рецепти, лише вручну/створені	Велика бібліотека готових рецептів	Готові рецепти з розрахованим КБЖВ

Продовження таблиці 1.1

Зведена таблиця аналізу існуючих застосунків для планування харчування та їх порівняння

Показник	**MyFitnessPal**	**Samsung Food**	**MealPrepPro**
Розрахунок КБЖВ	Автоматичний	Присутній частково (тільки для страв)	Вбудований для кожного рецепта
Персоналізація	Є, залежно від цілей і фізичних даних	Мінімальна, лише дієтичні фільтри	Обмежена — обирається тип дієти
Генерація страв	Відсутня	Відсутня	Відсутня
Облік фізіології	Вага, зріст, активність	Відсутній	Відсутній
Інтеграція з пристроями	Fitbit, Apple Watch, інші трекери	Пристрої Samsung (плити, холодильники)	Відсутня
Аналітика і графіки	Розгорнуті графіки ваги, нутрієнтів	Немає графіків, лише рецепти	Статистика добового КБЖВ без істроії
Підтримка української мови	Відсутня	Частково	Відсутня
Мобільність	Web, iOS, Android	Web, iOS, Android	iOS
Вартість	Безкоштовна версія, Premium із розширеним функціоналом	Безкоштовний	Платна підписка

Продовження таблиці 1.1

Зведена таблиця аналізу існуючих застосунків для планування харчування та їх порівняння

Показник	**MyFitnessPal**	**Samsung Food**	**MealPrepPro**
Цільова аудиторія	Широка аудиторія, всі бажаючі схуднути чи набрати вагу	Люди, які люблять готувати	Активні користувачі, спортсмени
Унікальні особливості	Найбільша база продуктів, синхронізація з фітнесом	Інтеграція з кухонною технікою	Професійні рецепти, тайм-менеджмент їжі

1.5 Визначення вимог до застосунку

Проаналізувавши функціональне призначення застосунку, його специфіку, очікування цільової аудиторії та існуючі аналоги, було сформульовано основні вимоги до вебзастосунку для персонального харчового планування:

– Авторизація користувача: можливість реєстрації, входу та виходу із системи, збереження сесій за допомогою JWT-токенів, захист даних на рівні бекенду (Express + Supabase).

– База продуктів: доступ до великої кількості харчових продуктів з API (Nutritionix), з можливістю пошуку за назвою, перегляду нутрієнтів (калорійність, білки, жири, вуглеводи) та додавання у комбінацію.

– Модуль комбінацій продуктів: створення користувачем власних страв/комбінацій з обраних інгредієнтів, автоматичний підрахунок загального КБЖВ та можливість збереження таких комбінацій у профілі.

– Генерація рецептів на основі штучного інтелекту (AI): надсилання запиту до AI-моделі (Gemini API), отримання текстового рецепта з інструкцією, інгредієнтами та орієнтовною калорійністю.

– Аналітика змін у харчуванні: побудова графіків змін ваги, статистика за збереженими рецептами, підрахунок середнього КБЖВ, типів кухонь.

– Модуль вимірювань (Measurements): фіксація ваги та зросту користувача у динаміці з прив’язкою до дати, збереження історії змін та відображення в графічному вигляді.

– Улюблені страви: можливість додавання рецептів та комбінацій до списку улюблених для швидкого повторного використання.

– Швидкий, адаптивний та зрозумілий інтерфейс користувача: зручна навігація між вкладками (Profile, Analytics, Recipes, MealPlanner тощо), мінімалістичний дизайн із використанням Tailwind CSS та бібліотеки компонентів shadcn/ui, підтримка мобільних пристроїв (PWA-формат).

2 ЗАСОБИ РЕАЛІЗАЦІЇ

2.1 Середовище розробки

Інтегроване середовище розробки (IDE) – це комплексне програмне забезпечення, що надає програмістам різноманітні інструменти для розробки програмного коду. Інтегровані середовища розробки зазвичай включають текстовий редактор, інструменти для автоматизації збірки коду, дебагер та, часто, інтерфейс для системи контролю версій. Це допомагає розробникам ефективно писати, тестувати і відлагоджувати свої програми в одному середовищі.

2.1.1 Visual Studio Code

Visual Studio Code (VS Code) — це передовий текстовий редактор від Microsoft, який широко використовується програмістами для розробки програмного забезпечення. Хоча VS Code сам по собі не є повноцінним інтегрованим середовищем розробки (IDE), його можна ефективно використовувати як IDE завдяки гнучкості та численним розширенням, які можуть значно розширити його функціональність.

VS Code підтримує багато мов програмування "з коробки", включаючи JavaScript, TypeScript, Python, PHP, C++, та інші. Це досягається через використання розширень, які можна легко встановити та налаштувати, адаптуючи редактор до специфічних вимог розробки.

Однією з ключових переваг VS Code є його система плагінів. Користувачі можуть встановлювати розширення для майже всього — від підтримки додаткових мов програмування і фреймворків до інтеграції з іншими інструментами та сервісами, такими як контейнери Docker, системи управління версіями (наприклад, Git) і бази даних.

Хоча VS Code не є IDE в класичному розумінні цього слова, його можливості та широка адаптованість роблять його відмінним вибором для розробників, які

шукають легкий, але потужний інструмент для кодування.

2.2 Мови програмування

Мова програмування — це формалізована система інструкцій, яку використовують для написання програм, що керують поведінкою комп’ютера або реалізують алгоритми. Сучасні мови дозволяють розробникам створювати ефективні, масштабовані та зручні для підтримки програмні рішення. У контексті веброзробки найпоширенішими є мови JavaScript, TypeScript та Python.

2.2.1 TypeScript

TypeScript — це мова програмування з відкритим вихідним кодом, розроблена компанією Microsoft. Вона є надмножиною JavaScript, що забезпечує підтримку статичної типізації. TypeScript дозволяє розробникам явно задавати типи змінних, функцій, об'єктів, що значно знижує кількість помилок і полегшує підтримку коду.

TypeScript транслюється у JavaScript, тому він сумісний із будь-якими платформами, що підтримують останній: браузерами, серверами на Node.js тощо. Статична типізація особливо корисна для командної роботи над великими проєктами, оскільки дозволяє уникнути труднощів, пов’язаних з динамічною природою JavaScript.

Мова підтримує об’єктно-орієнтовані підходи, включаючи класи, інтерфейси, наслідування, дженерики. Вона інтегрується з багатьма сучасними фреймворками, включаючи React, Angular та Vue.js.

2.2.2 JavaScript

JavaScript — мова програмування, яка є основою для створення інтерактивних вебінтерфейсів. Вона підтримується всіма сучасними браузерами без потреби в додаткових плагінах. JavaScript використовується як на стороні клієнта (frontend), так і на стороні сервера (з використанням Node.js).

Мова є динамічно типізованою, прототипно-орієнтованою та мультипарадигмальною — вона підтримує функціональний, об'єктно-орієнтований та імперативний підходи. Завдяки постійній еволюції стандарту ECMAScript JavaScript отримує нові можливості щороку, включаючи асинхронне програмування через async/await, модулі, деструктуризацію, замикання тощо.

2.3 Веб-фреймворки

Веб-фреймворки — це програмні платформи, призначені для прискорення процесу розробки вебзастосунків. Вони забезпечують структуру проєкту, стандартизацію архітектурних підходів, а також надають набір готових функціональних компонентів (наприклад, маршрутизацію, керування станом, роботу з API).

Фреймворки діляться на клієнтські (frontend) та серверні (backend). У клієнтських фреймворках головна мета — створення інтерфейсу користувача та взаємодія з даними. У серверних — обробка запитів, доступ до баз даних, реалізація бізнес-логіки.

2.3.1 React

React[9] — це JavaScript-бібліотека для створення користувацьких інтерфейсів, розроблена компанією Meta (раніше Facebook). Вона базується на компонентній архітектурі, що дозволяє розбивати інтерфейс на незалежні, повторно використовувані частини. Кожен компонент може містити власну логіку та стан.

React активно використовує концепції Virtual DOM, що підвищує продуктивність шляхом мінімізації кількості змін у реальному DOM. У поєднанні з TypeScript, React дозволяє створювати масштабовані та стабільні односторінкові застосунки (SPA).

2.3.2 Express

Express.js[14] — це мінімалістичний серверний фреймворк для Node.js, що використовується для створення веб-серверів та API. Його гнучкість і простота зробили його популярним вибором для розробників серверної логіки. Express забезпечує маршрутизацію запитів, підтримку middleware, обробку форм, роботу з базами даних, автентифікацію тощо.

2.3.3 Tailwind CSS

Tailwind CSS[11] — це сучасний фреймворк для стилізації інтерфейсів, що реалізує утилітарний підхід до CSS. На відміну від традиційних фреймворків, які пропонують готові компоненти з наперед заданим виглядом (наприклад, Bootstrap або Materialize), Tailwind надає велику кількість низькорівневих CSS-класів для створення інтерфейсу за принципом “utility-first” — тобто шляхом послідовного додавання атомарних стилів без потреби написання кастомного CSS.

Класи Tailwind охоплюють майже всі аспекти верстки: розміри, відступи, кольори, типографіку, гнучкі сітки, адаптивність, псевдокласи (hover, focus тощо) та багато іншого. Завдяки цьому, розробники можуть швидко будувати й змінювати інтерфейс без створення окремих CSS-файлів або написання ручного стилю.

Tailwind CSSактивно застосовується у сучасних frontend-фреймворках (React, Vue, Svelte тощо) та підтримує роботу з компонентним підходом, що робить його популярним вибором для побудови швидких, адаптивних і візуально цілісних вебзастосунків.[12]

2.4 Системи управління базами даних

Система управління базами даних (СУБД) — це програмне забезпечення, яке дозволяє створювати, зберігати, змінювати та управляти структурованими даними. У сучасній веброзробці особливо важливо використовувати СУБД, що забезпечує швидку взаємодію з API, підтримує масштабованість і має вбудовані механізми безпеки.

Supabase[6] — це хмарна платформа з відкритим кодом, яка забезпечує повноцінний backend для веб- і мобільних застосунків. Вона побудована на базі PostgreSQL і поєднує в собі функціональність бази даних, аутентифікації, хмарного зберігання та генерації API в єдиному інтерфейсі.

Оскільки основою Supabase є PostgreSQL[7], він успадковує всі можливості цієї надійної реляційної СУБД: підтримку складних SQL-запитів, транзакцій, зовнішніх ключів, індексів, а також обробку структурованих даних у форматі JSON.

Основні функціональні можливості Supabase:

– Хмарна реляційна база даних (PostgreSQL);

– Генерація RESTful та Realtime API автоматично на основі таблиць;

– Аутентифікація та авторизація користувачів (email, OAuth, magic links тощо)[8];

– JWT-токени — для безпечного доступу до даних та реалізації захисту маршрутизованих запитів;

– Рольова система доступу (RLS) — на рівні бази даних можна налаштовувати, хто які дані може бачити, змінювати або створювати;

– Зберігання файлів (Storage) — для роботи з зображеннями, документами тощо;

– Інтеграція з клієнтськими фреймворками на JavaScript/TypeScript.

Supabase є зручною альтернативою класичному підходу із самостійним розгортанням серверної частини. Завдяки простій інтеграції та потужним можливостям, платформа дозволяє розробникам зосередитись на логіці застосунку, мінімізуючи витрати часу на налаштування backend-інфраструктури.

Завдяки наявності в Supabase аутентифікації з JWT-токенами, дані користувачів у застосунку залишаються захищеними, а доступ до ресурсів — контрольованим і безпечним.

2.5 Система контролю версій

Система контролю версій дозволяє відслідковувати зміни в коді, координувати командну роботу, відновлювати попередні версії програмного забезпечення. Найпоширенішою є система Git у поєднанні з сервісами GitHub, GitLab або Bitbucket.

GitHub — веб-платформа, яка надає інструменти для хостингу репозиторіїв, управління версіями коду, відслідковування змін та спільної розробки. Розробники можуть створювати pull-request'и, здійснювати рев'ю коду, вести документацію та використовувати CI/CD інтеграцію. GitHub підтримує відкриті та приватні проєкти, а також інтегрується з багатьма іншими інструментами DevOps.

2.6 Інструменти проєктування інтерфейсу користувача

Інтерфейс користувача — ключова частина вебзастосунку, що безпосередньо впливає на досвід користувача. Для створення дизайну сучасних UI використовуються спеціалізовані інструменти.

Figma[4] — це веборієнтований інструмент для UI/UX-дизайну, що дозволяє створювати макети, прототипи та дизайн-системи в режимі реального часу. Figma підтримує спільну роботу, анімацію, компоненти, автолейаути та інтерактивні прототипи. Унікальною особливістю Figma є можливість роботи без встановлення локального ПЗ — усі функції доступні через браузер.

3 ПРОЄКТУВАННЯ

3.1 Проєктування архітектури застосунку

Загальна архітектура веб-застосунку Planner-Calorie побудована на основі класичної клієнт-серверної моделі, яка забезпечує чітке розділення обов'язків між клієнтською та серверною частинами. Такий підхід широко застосовується в сучасній веброзробці завдяки своїй масштабованості, модульності та зрозумілості.

Інтерфейс реалізовано за SPA-підходом (Single Page Application) з використанням бібліотеки React. Це означає, що зміна вмісту (зокрема, навігація між сторінками особистого кабінету: комбінації, рецепти, цілі, аналітика тощо) відбувається без повного перезавантаження сторінки, а за допомогою клієнтської маршрутизації. Такий підхід забезпечує кращий користувацький досвід та високу продуктивність інтерфейсу.

Фронтенд побудований з використанням React, Tailwind CSS та бібліотеки компонентів shadcn/ui. Клієнтська частина надсилає HTTP-запити до серверної частини та зовнішніх API для отримання або надсилання даних.

Серверна частина реалізована на Express (Node.js). Вона відповідає за обробку запитів, авторизацію користувача за допомогою JWT, збереження та оновлення даних у базі, а також виступає проміжною ланкою між клієнтом і сторонніми сервісами.[13]

Supabase використовується як основна база даних (побудована на PostgreSQL). Вона забезпечує:

– зберігання комбінацій, цілей, історії змін, вимірів;

– управління ролями доступу через RLS;

– аутентифікацію та перевірку токенів JWT.

– Крім цього, застосунок інтегрується з:

– Gemini API — генерація AI-рецептів за обраними інгредієнтами та ціллю користувача;

– Nutritionix API — підрахунок калорійності та КБЖВ на основі введених продуктів.

Обрана архітектура забезпечує кілька суттєвих для проєкту переваг:

– Масштабованість – додавання нових функцій без переробки всієї системи;

– Модульність – фронтенд, сервер і база даних розділені логічно;

– Гнучкість – легко замінити окремі сервіси або бібліотеки;

– Інтеграція – використання сучасних API (Gemini, Nutritionix);

– Безпечність – авторизація через JWT і контроль доступу через Supabase;

– Продуктивність – оновлення інтерфейсу без перезавантаження сторінки, асинхронна взаємодія з сервером.

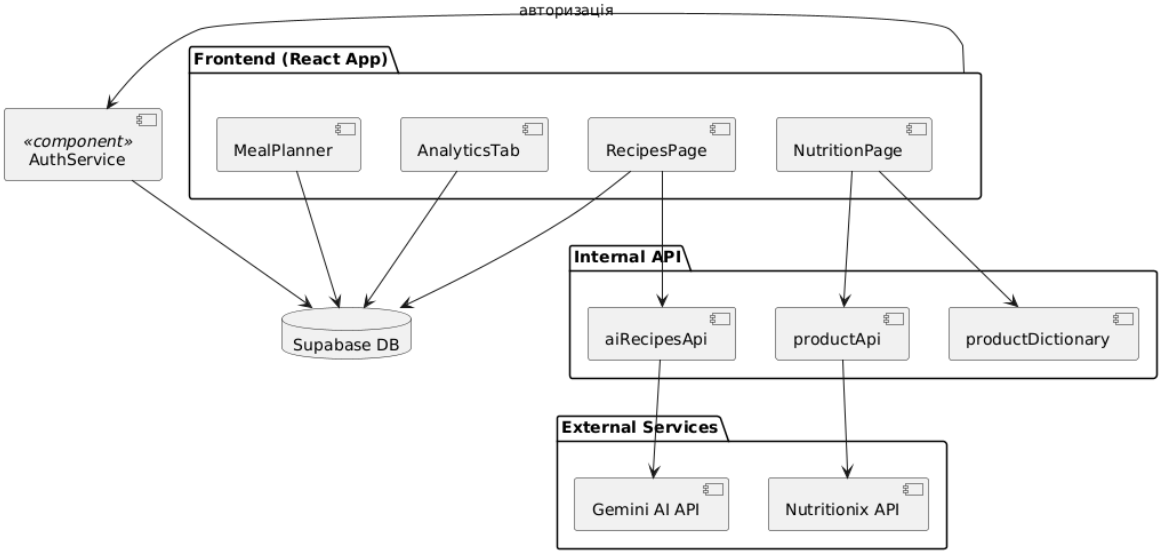


Рис. 3.1 Діаграма компонентів

Для ілюстрації наведено діаграму компонентів, яка відображає логічну структуру взаємодії між клієнтською частиною, API, зовнішніми сервісами та базою даних.

3.2 Структура клієнтської частини

Фронтенд реалізовано за допомогою бібліотеки React, покращеної TypeScript для безпеки типів та кращої підтримки. Ця частина системи розроблена відповідно до компонентно-орієнтованої парадигми, яка заохочує модульність, повторне використання коду та чіткий розподіл обов'язків. Це дозволяє створювати інтерактивні та адаптивні інтерфейси, які динамічно реагують на дії користувача без необхідності перезавантаження всієї сторінки.

Кожна логічна частина інтерфейсу розроблена як незалежний компонент. Це означає, що різні екрани застосунку — зокрема генерація рецептів, перегляд КБЖВ, аналітика або збережені комбінації — реалізовані як окремі сторінки та модулі, які виводять відповідну інформацію користувачу й дозволяють взаємодіяти з нею.

Компонент Recipes забезпечує інтерфейс для введення параметрів рецепта та перегляду результатів, згенерованих за допомогою штучного інтелекту. Обрані рецепти можуть бути збережені у вкладці MealPlanner, яка виконує функцію колекції обраних страв. Компонент Nutrition надає можливість переглядати дані про калорійність та склад продуктів, а також формувати харчові комбінації, які користувач може додати до обраного через FavoriteCombos. Аналітична частина реалізована у AnalyticsTab, де відображається візуалізована інформація про фізичні показники користувача та статистику споживання.

Процеси автентифікації реалізовані через відповідні компоненти, що забезпечують вхід та реєстрацію, обробку помилок і зберігання токена JWT. Вся візуальна частина побудована з використанням Tailwind CSS та бібліотеки інтерфейсних компонентів shadcn/ui, що забезпечує швидку розробку й адаптивну верстку.

Загалом, клієнтська архітектура побудована таким чином, щоб забезпечити зручний, гнучкий та масштабований користувацький інтерфейс. Модульна структура фронтенду дозволяє незалежну розробку й обслуговування окремих частин застосунку, а також полегшує інтеграцію з серверною частиною системи.

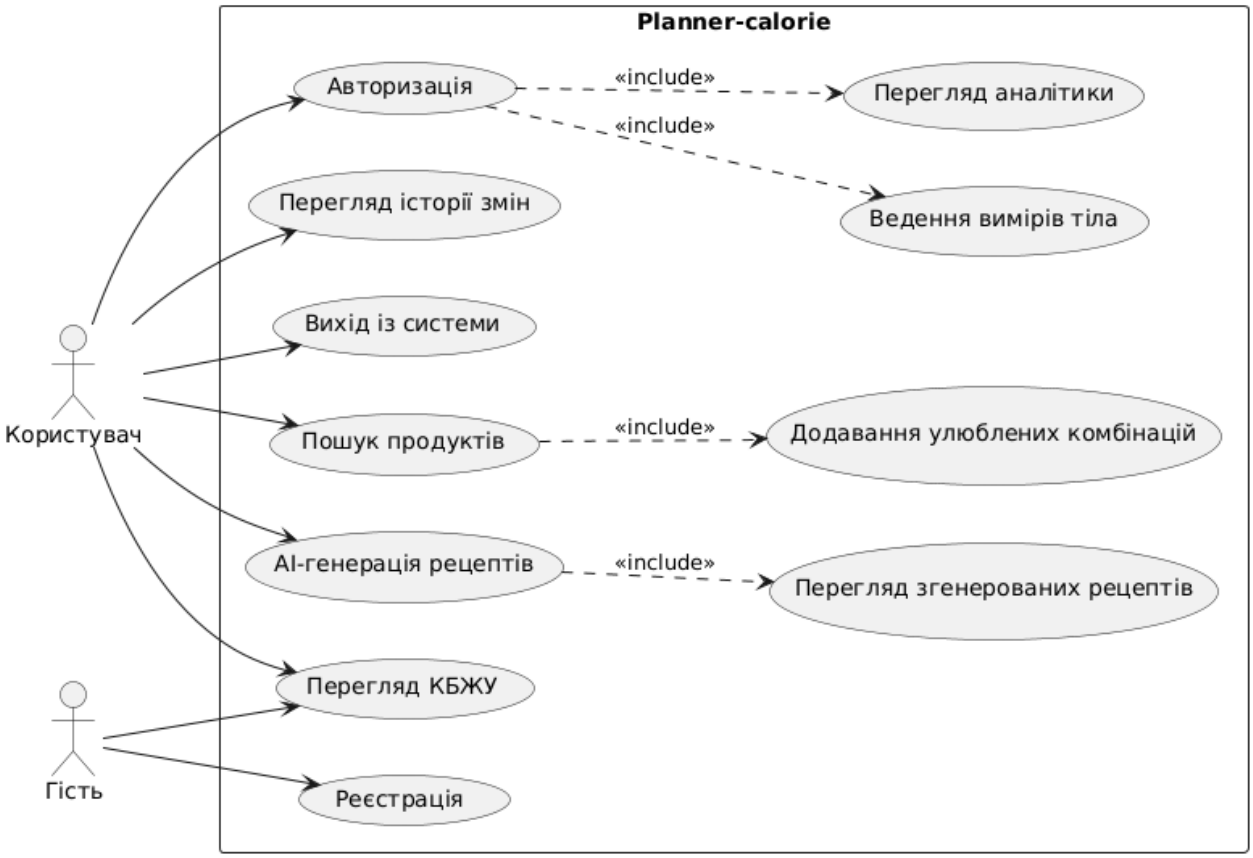


Рис. 3.2 Діаграма варіантів використання

Наведена діаграма варіантів використання демонструє всі можливі основні сценарії взаємодії між користувачем,гостем та клієнтською частиною застосунку.

3.3 Структура серверної частини

Серверна частина застосунку Planner-Calorie реалізована за допомогою Express.js — легкого та продуктивного фреймворку для створення RESTful API. Вона забезпечує обробку запитів від клієнтської частини, взаємодію з зовнішніми сервісами та базою даних Supabase, а також реалізацію логіки автентифікації та авторизації користувачів через JWT.

Архітектура серверної частини дотримується принципів модульності: кожен функціональний блок винесено в окремий модуль або сервіс (наприклад, aiRecipesApi, productApi, authService), що забезпечує ізоляцію логіки, зручність тестування та масштабованість системи.

Взаємодія з зовнішніми сервісами:

– Gemini API: Для генерації AI-рецептів використовується зовнішній сервіс Gemini. Серверна частина отримує запит з параметрами (наприклад, бажані інгредієнти або тип страви), надсилає його до API Gemini та повертає відповідь у вигляді згенерованого рецепта.

– Nutritionix API[1]: Пошук продуктів та підрахунок харчових значень (КБЖВ) здійснюється через Nutritionix API. Користувач може шукати продукти, переглядати калорійність та формувати власні комбінації.

Уся інформація користувача, включаючи рецепти, харчові комбінації, виміри тіла, цілі та історію споживання, зберігається у базі даних Supabase (PostgreSQL). Express-сервер виконує взаємодію з цією базою через REST-запити та Supabase SDK. Для кожної сутності (наприклад, Recipe, FavoriteCombo, Measurement, MealPlan) реалізовано окремі маршрути для створення, читання, оновлення та видалення (CRUD-операції).

Система автентифікації реалізована на основі JSON Web Token (JWT). Після успішної авторизації Supabase видає токен, який передається в кожному запиті до захищених маршрутів. Сервер перевіряє валідність токена перед доступом до персоналізованих даних користувача. Для спрощення керування доступом реалізовано middleware-шар перевірки токена.

Особливості реалізації

– Всі маршрути API побудовані за принципами REST.

– Асинхронна обробка запитів забезпечує продуктивність при взаємодії з зовнішніми сервісами.

– Серверна частина повністю сумісна з клієнтською, дозволяючи їй швидко оновлювати дані в режимі реального часу.

– Завдяки модульності та розділенню логіки, застосунок легко підтримується та розширюється новим функціоналом.

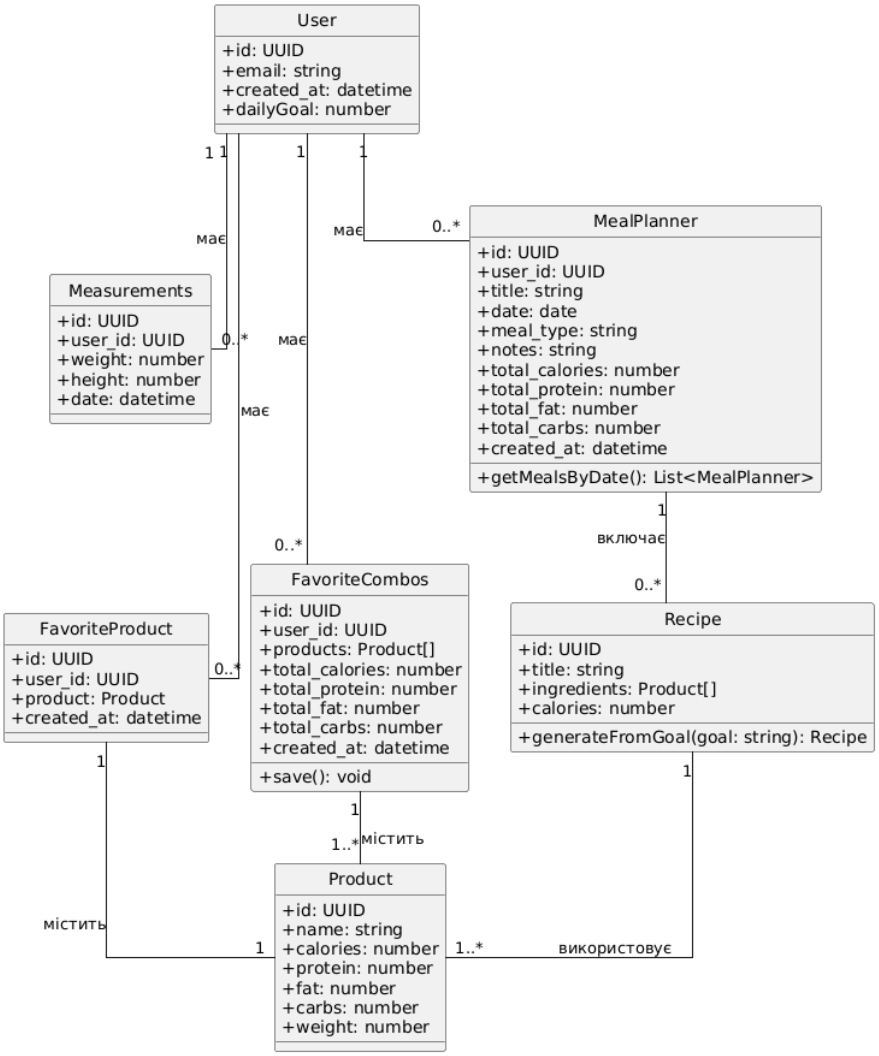


Рис. 3.3 Діаграма класів

Наведено діаграму класів, яка ілюструє головні сутності на сервері та зв’язки між ними, зокрема користувача, рецептів, харчових комбінацій, вимірів, цілей та токена авторизації.

3.4 Модель даних та база даних

Для реалізації функціональності веб-застосунку Planner-Calorie було спроєктовано реляційну модель даних, яка дозволяє організовано зберігати всі ключові сутності та забезпечує логічну взаємодію між ними. Обрана модель є основою для зберігання персоналізованої інформації про користувача, його цілі, споживання продуктів, рецепти, аналітику харчування та інші дії в системі.

У структурі даних визначено декілька типів сутностей: користувачі, вимірювання, продукти, рецепти, комбінації, обране, а також планування харчування. Кожна з цих сутностей відображає окремий аспект роботи додатку. Наприклад, сутність користувача містить загальну інформацію про обліковий запис, а також виступає центральним вузлом у багатьох взаємозв'язках з іншими таблицями. Через зовнішні ключі до неї прив'язуються такі елементи, як історія вимірювань, обрані продукти, створені рецепти та сформовані комбінації.

Сутності, пов'язані з харчуванням, — рецепти, продукти, комбінації — взаємодіють між собою для формування повної картини спожитих або запланованих страв. Кожен рецепт, наприклад, може складатися з набору продуктів, кожна комбінація також формується з окремих продуктів, а заплановані прийоми їжі включають рецепти, підібрані користувачем або запропоновані системою. Таким чином, реалізується багаторівнева система зв'язків, яка дає змогу ефективно структурувати дані для подальшого аналізу та візуалізації.

Сутність вимірювань використовується для фіксації фізичних показників (ваги, зросту тощо) у часовій динаміці. Це забезпечує відстеження прогресу користувача відносно поставлених цілей і може використовуватись як візуальна аналітика або як основа для адаптації раціону.

Реляційна модель дає змогу забезпечити високу узгодженість та достовірність збережених даних. Використання первинних та зовнішніх ключів дозволяє чітко визначати залежності між таблицями, уникати дублювання даних, а також реалізовувати каскадне оновлення або видалення, що є важливим при роботі з великою кількістю пов'язаних об'єктів.

Завдяки чітко визначеним зв'язкам та логічній структурі база даних не лише обслуговує поточні потреби додатку, а й закладає основу для майбутнього масштабування. У разі розширення функціональності можна безболісно додати нові таблиці (наприклад, історію споживання, графік водного балансу, спортивну активність), не змінюючи існуючу модель. Це робить систему гнучкою до змін та стійкою до зростання обсягів інформації.

Загалом, модель даних у Planner-Calorie дозволяє комплексно відображати поведінку користувача у сфері харчування, підтримує персоналізацію, зберігає історичні дані та забезпечує ґрунт для глибшого аналізу споживчих звичок.

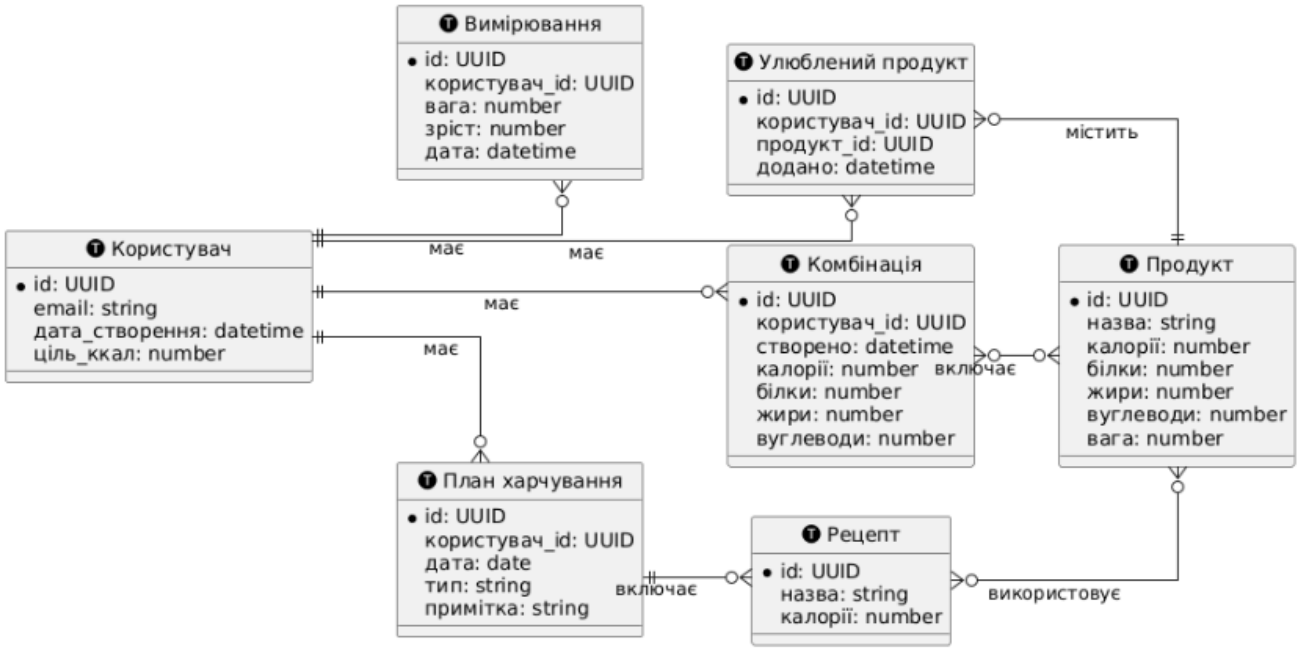


Рис. 3.4 ER-діаграма

ER-діаграма відображає логічну структуру даних системи: від користувача як центральної сутності — до взаємопов’язаних елементів, таких як рецепти, продукти, вимірювання, планування меню та обране. Взаємозв’язки між сутностями базуються на зовнішніх ключах, що гарантує цілісність та відповідність даних на всіх рівнях.

4 РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

4.1 Реалізація пошуку КБЖВ продуктів

Однією з ключових функцій застосунку Planner-Calorie є пошук поживної цінності продуктів — калорій, білків, жирів і вуглеводів (КБЖВ) — на основі введеного користувачем текстового запиту. Особливістю реалізації є повна підтримка української мови при взаємодії з англомовним API.

Оскільки використовувана версія Nutritionix API підтримує лише англійську мову, у системі реалізовано власний словник перекладів PRODUCT_DICTIONARY, який дозволяє користувачам вводити продукти українською, а система — автоматично трансформує ці терміни для запиту. Наприклад, введений користувачем запит "рис, курка" буде перетворено на "rice, chicken" перед відправкою.

Функція searchProduct, розташована у файлі productApi.js, виконує переклад і формує запит:

```
const translated = PRODUCT_TRANSLATIONS[product] || product;
const response = await axios.post(
  "https://trackapi.nutritionix.com/v2/natural/nutrients",
  { query: `1 ${translated}` },
  {
    headers: {
      "x-app-id": "...",
      "x-app-key": "...",
      "Content-Type": "application/json"
    }
  }
);
```

Відповідь API містить англомовні назви продуктів і значення КБЖВ на 100

г. Дані потім виводяться у користувацький інтерфейс, де користувач може ввести бажану вагу продукту у грамах, а система перерахує значення:

```
const ratio = grams / 100;
{
  calories: +(product.calories * ratio).toFixed(2),
  protein: +(product.protein * ratio).toFixed(2),
  fat: +(product.fat * ratio).toFixed(2),
  carbs: +(product.carbs * ratio).toFixed(2),
}
```

Інтерфейс Nutrition.jsx дозволяє:

– вводити продукти через кому (українською),

– бачити таблицю результатів з КБЖВ для заданої ваги,

– формувати комбіновані страви або раціони,

– зберігати їх як КБЖВ-комбінації.

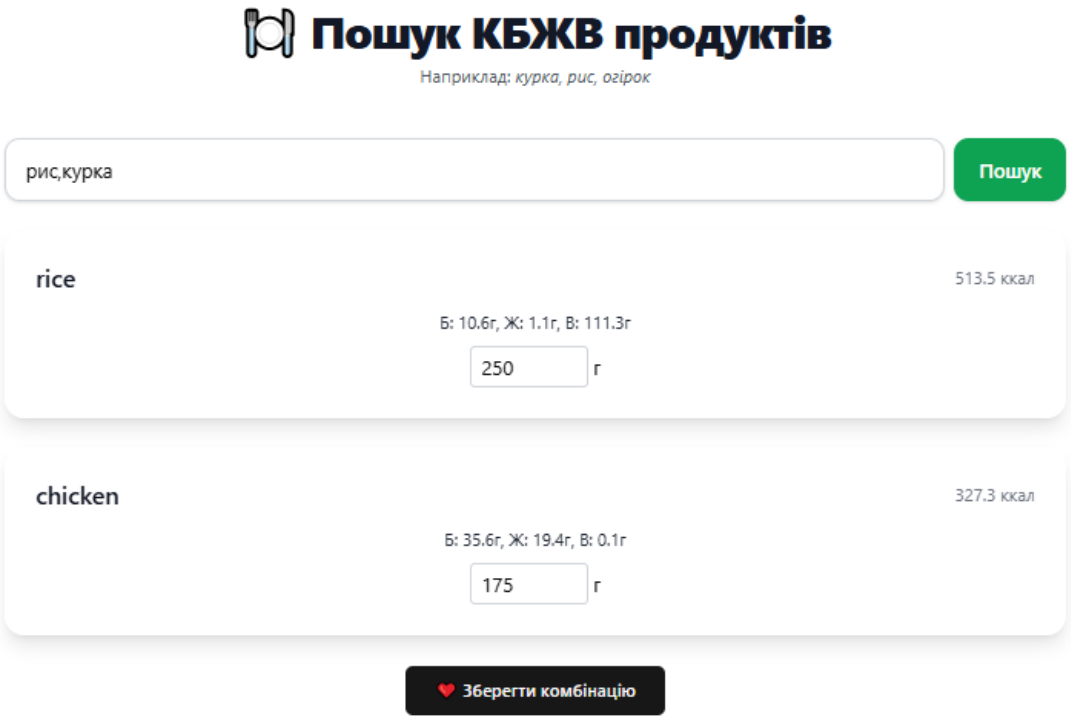


Рис. 4.1 Екранна форма пошуку продуктів, введених українською мовою, з результатами від Nutritionix API

Таким чином, реалізована система поєднує точність зовнішнього API з зручністю використання української мови. Це дозволяє не лише розширити аудиторію, але й зробити процес підбору раціону інтуїтивно зрозумілим для користувачів без знання англійської.

4.2 Комбінації продуктів: збереження та перегляд

Вкладка «Комбінації продуктів» є окремим функціональним модулем у застосунку Planner-Calorie, що дозволяє користувачеві переглядати збережені раніше комбінації харчових продуктів із підрахованими поживними значеннями. Ця функція розширює зручність використання системи, забезпечуючи швидкий доступ до персоналізованих раціонів.

Кожна комбінація формується у вкладці Nutrition.jsx, де користувач вводить назви продуктів, їхню вагу, а система автоматично розраховує поживну цінність. Після натискання кнопки «Зберегти комбінацію» в компоненті SaveCombinationButton.jsx дані зберігаються в таблицю favorite_combinations у Supabase:

```
{
  user_id: user.id,
  combination: [ /* масив продуктів з weight та КБЖВ */ ],
  total_calories: ...,
  total_protein: ...,
  total_fat: ...,
  total_carbs: ...,
}
```

У вкладці FavoriteCombos.jsx збережені комбінації відображаються як картки, кожна з яких містить:

- номер комбінації (id);
- перелік інгредієнтів (назва, вага, КБЖВ на вагу);
- сумарні значення по калоріях, білках, жирах, вуглеводах;

– кнопку «Видалити комбінацію».

```
{combo.combination.map((item) => (
  <div key={item.name}>
    {item.name} — {item.weight}г, {item.calories.toFixed(2)} ккал,
    Б: {item.protein.toFixed(2)}г, Ж: {item.fat.toFixed(2)}г, В: {item.carbs.toFixed(2)}г
  </div>
))}
```

Після натискання кнопки "Оновити комбінації" викликається функція оновлення, яка перезапитує актуальні дані з бази Supabase.

Комбінації відображаються за допомогою анімованих компонентів Card та motion.div, що забезпечує плавний візуальний ефект появи. Візуально кожна картка структурована так, щоб інформація була читабельною, із кольоровим виділенням макронутрієнтів.

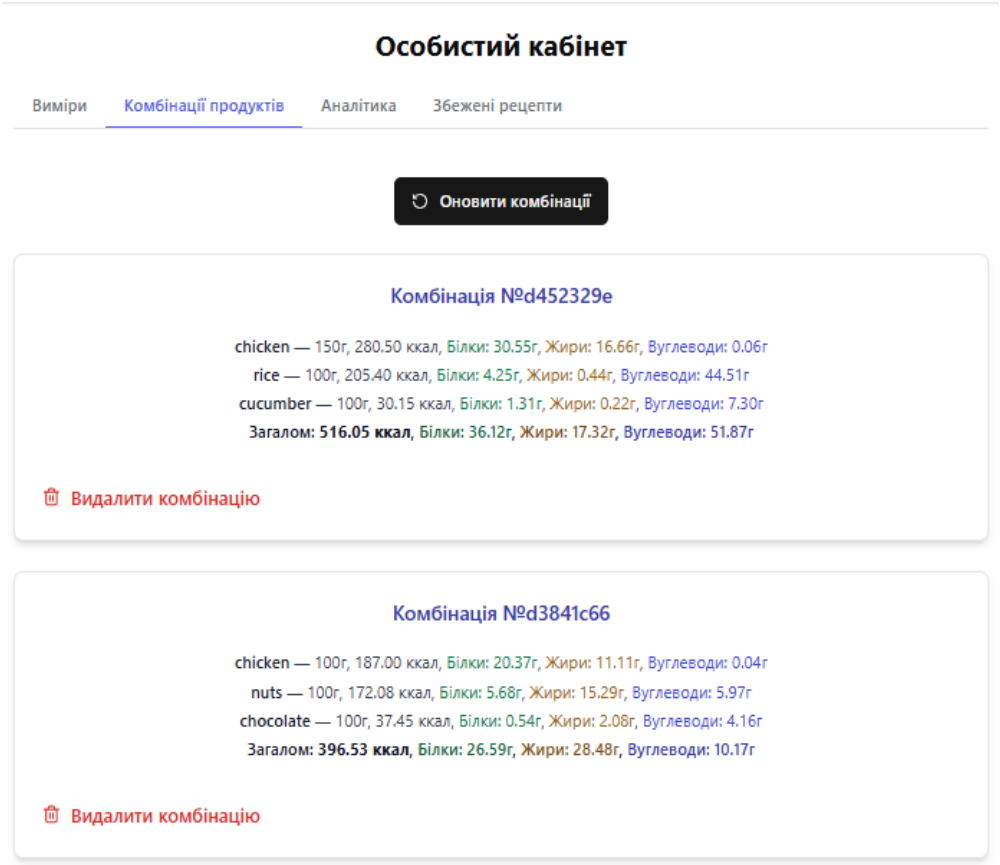


Рис. 4.2 Екранна форма збережених комбінацій

Цей модуль є ядром ручного підбору харчування в системі: замість створення рецепта кожного разу з нуля, користувач формує власну бібліотеку комбінацій, яку може переглядати, аналізувати, видаляти або повторно використовувати при плануванні.

4.3 Генерація AI-рецептів

У застосунку Planner-Calorie реалізовано можливість створення кулінарних рецептів за допомогою штучного інтелекту. Ця функція дозволяє користувачеві вводити бажані інгредієнти та тип кухні, після чого система генерує повноцінний рецепт, включаючи назву, список інгредієнтів, інструкцію з приготування та розраховану харчову цінність (КБЖВ).

Генерація здійснюється через інтеграцію з Gemini API, який отримує сформований prompt і повертає текст рецепту українською мовою. Запит формується у файлі aiRecipesApi.ts:

const prompt = `

Склади повноцінний і граматично правильний рецепт у стилі "${cuisine}".

Використай інгредієнти: ${Array.isArray(ingredients) ? ingredients.join(", ") : ingredients}.

Вивід має включати:

1. Назву страви (на окремому рядку).

2. Список інгредієнтів з одиницями.

3. Покрокову інструкцію.

4. КБЖВ (на 100 г).

```
const response = await fetch(GEMINI_URL, {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({
    contents: [{ parts: [{ text: prompt }] }]
  }),
```

```
});
```

На клієнтській частині користувач взаємодіє з інтерфейсом сторінки Recipes.jsx, вводячи інгредієнти та обираючи тип кухні:

```
<input
  value={ingredients}
  onChange={(e) => setIngredients(e.target.value)}
  placeholder="наприклад: курка, рис, морква"
/>
```

Після натискання кнопки "Згенерувати рецепт", запит надсилається до Gemini API. Отриманий текст рецепту зберігається в стані компонента й відображається користувачу. Вивід форматовано за блоками: назва, інгредієнти, інструкція, КБЖВ. Для цього реалізовано функцію renderRecipePreview, яка розділяє текст на логічні частини.

Після ознайомлення з рецептом користувач може зберегти страву у свій MealPlanner, натиснувши кнопку:

```
const plan = {
  user_id: user.id,
  date: new Date().toISOString().slice(0, 10),
  meal_type: "Страва",
  title: `Рецепт (${ingredients})`,
  notes: generatedText,
  ...
};
const { error } = await supabase.from("meal_plans").insert([plan]);
```

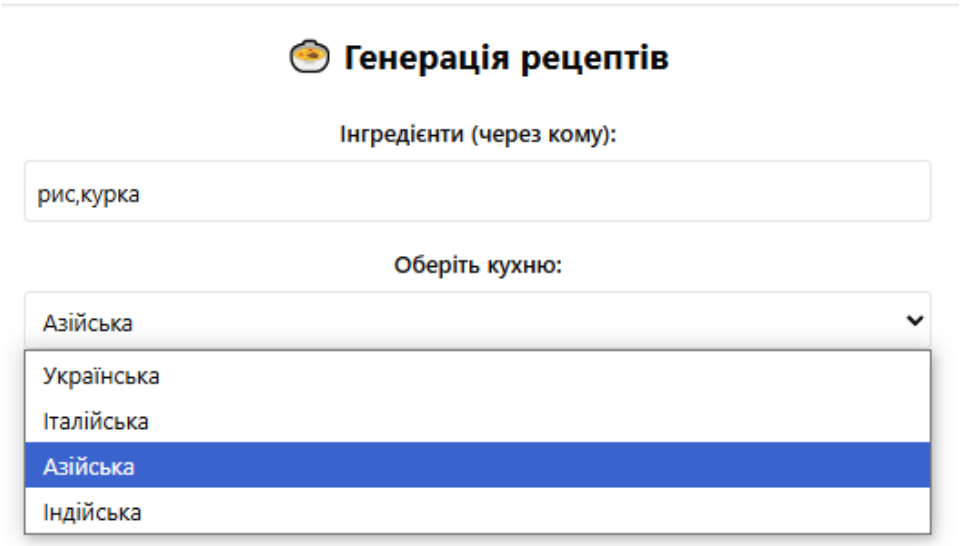


Рис. 4.3 Екранна форма генерації рецепту з полем введення інгредієнтів, типом кухні, результатом та кнопкою збереження

Рис. 4.4 Екранна форма згенерованого рецепту та кнопкою збереження

Таким чином, AI-рецепти є не лише зручним інструментом для швидкого створення страв, але й логічно інтегровані в загальну систему планування харчування, дозволяючи користувачу формувати власну кулінарну базу на основі інтелектуального підбору.

4.4 Фіксація фізичних вимірів користувача

У застосунку Planner-Calorie реалізовано функціональність для зберігання й візуалізації фізичних показників користувача — зокрема, ваги та зросту. Це дає змогу відстежувати динаміку змін тіла в часі та корелювати ці зміни з харчовими звичками.

Інтерфейс реалізовано у компоненті Measurements.jsx, де користувач може ввести свою поточну вагу (у кг) та зріст (у см) через текстові поля:

```
<input
  type="number"
  placeholder="Вага (кг)"
  value={weight}
  onChange={(e) => setWeight(e.target.value)}
/>
<input
  type="number"
  placeholder="Зріст (см)"
  value={height}
  onChange={(e) => setHeight(e.target.value)}
/>
```

Після заповнення натискається кнопка “Зберегти вимір”, яка викликає функцію onSave. Зазвичай ця функція зберігає дані в таблицю measurements бази даних Supabase разом із міткою часу (created_at).

Нижче форми введення виводиться історія попередніх вимірів, де кожен запис містить дату, вагу та зріст:

```
<li key={idx}>
  {formatDate(item.date)} — <strong>{item.weight} кг</strong>, {item.height} см
</li>
```

Також відображається динаміка змін ваги між двома останніми записами — із кольоровим індикатором зменшення (червоний) або зростання (зелений) та відповідним значком.

Якщо збережено більше одного запису, система автоматично генерує лінійний графік змін ваги на основі recharts:

```
<LineChart data={history.slice().reverse()}>
  <XAxis dataKey="date" tickFormatter={formatDate} />
  <YAxis unit=" кг" />
  <Tooltip labelFormatter={formatDate} />
  <Line type="monotone" dataKey="weight" stroke="#2563eb" />
</LineChart>
```

Графік відображається адаптивно завдяки компоненту ResponsiveContainer і забезпечує швидку візуальну оцінку прогресу.

Виміри не потребують додаткових підтверджень або складного інтерфейсу — достатньо ввести числові значення та натиснути кнопку. Це дозволяє оперативно фіксувати дані одразу після зважування. Такий підхід формує мотивацію до регулярного моніторингу та підвищує залучення до процесу самоконтролю.

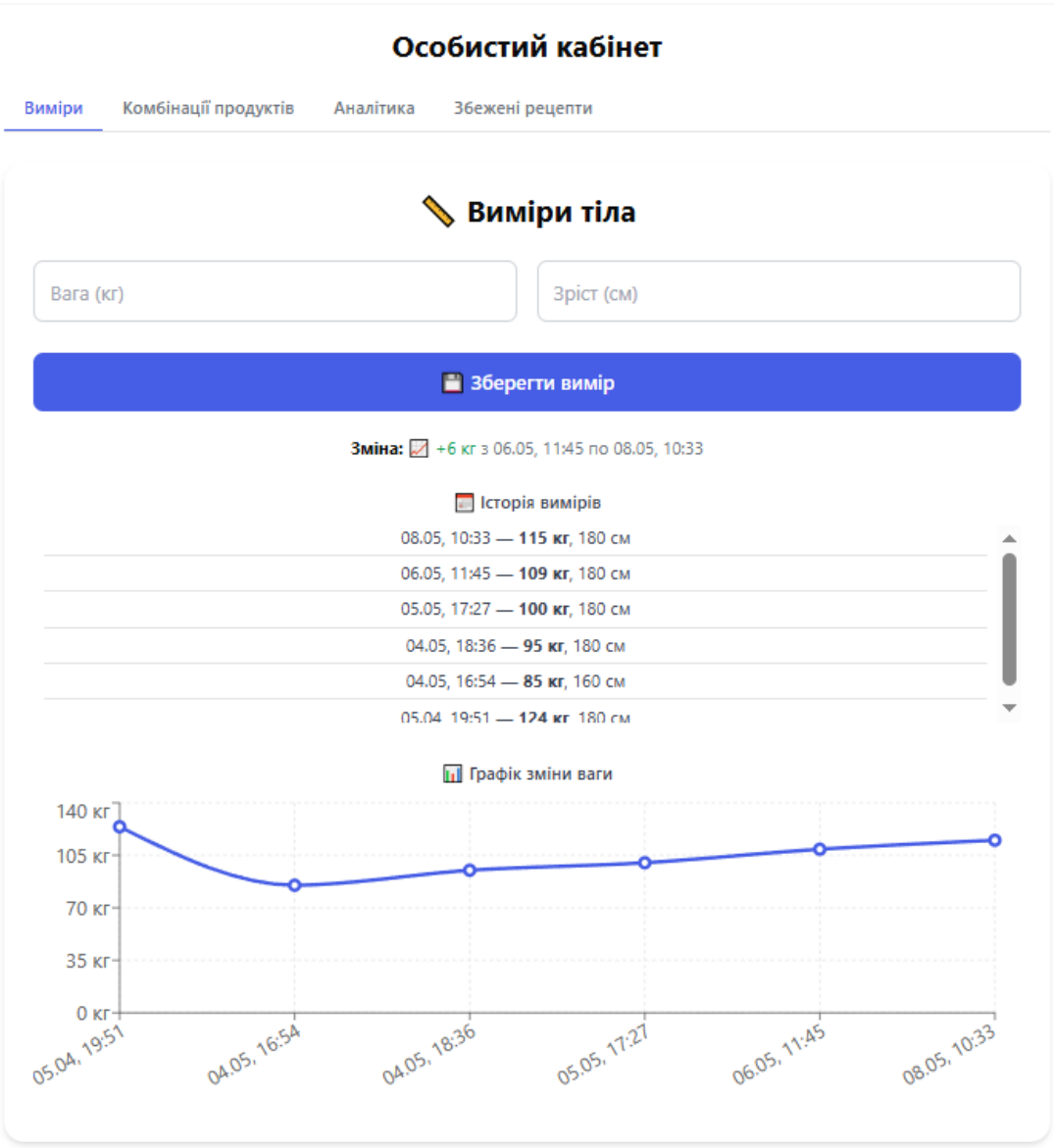


Рис. 4.5 Введення вимірів та графік змін ваги користувача

Таким чином, система вимірів у Planner-Calorie надає користувачам ефективний інструмент для моніторингу фізичних змін, підвищуючи мотивацію та контроль у досягненні цілей здорового способу життя.

4.5 Аналітика харчування та прогресу

Аналітична вкладка в застосунку Planner-Calorie надає користувачам можливість переглядати узагальнену інформацію про їхні харчові звички та зміни фізичних показників у динаміці.

Основна мета — надати зворотний зв’язок, який допомагає адаптувати раціон та оцінювати ефективність прийнятих рішень.

Інтерфейс вкладки реалізовано у компоненті AnalyticsTab.jsx. Після авторизації користувача система виконує два основні запити:

Отримання збережених прийомів їжі (meal_plans) із бази Supabase;

Запит до аналітичного бекенду /api/analytics/weight-summary для розрахунку середньої ваги за останні 30 днів:

```
const { data: recipes } = await supabase
  .from("meal_plans")
  .select("*")
  .eq("user_id", user.id);
const res = await fetch("http://localhost:5000/api/analytics/weight-summary", {
  headers: { Authorization: `Bearer ${token}` },
});
```

Після обробки відповіді користувач бачить блок із підсумком:

Середня вага за останні 30 днів: 70.2 кг (записів: 12)

Це дозволяє відслідковувати довготривалу динаміку змін без потреби вручну аналізувати історію вимірів.

Другий важливий блок — компонент RecipeAnalytics.jsx.

Його основна мета — візуалізувати структуру харчування користувача за збереженими стравами. Для цього використовуються графіки на базі бібліотеки recharts.

Секторна діаграма типів кухонь показує, які стилі харчування переважають у користувача (українська, італійська тощо). Дані агрегуються за полем kitchen у масиві archive.

```
const kitchenData = Object.entries(kitchenCounts).map(([k, v]) => ({ name: k, value: v }));
```

Стовпчаста діаграма калорійності виводить кількість калорій для кожного рецепта, що дозволяє швидко оцінити розподіл енергетичної цінності:

```
<Bar dataKey="calories" fill="#3b82f6" />
```

Ключові статистики подаються у вигляді окремих блоків:

– кількість збережених рецептів;

– середня калорійність;

– кількість унікальних кухонь у вибірці

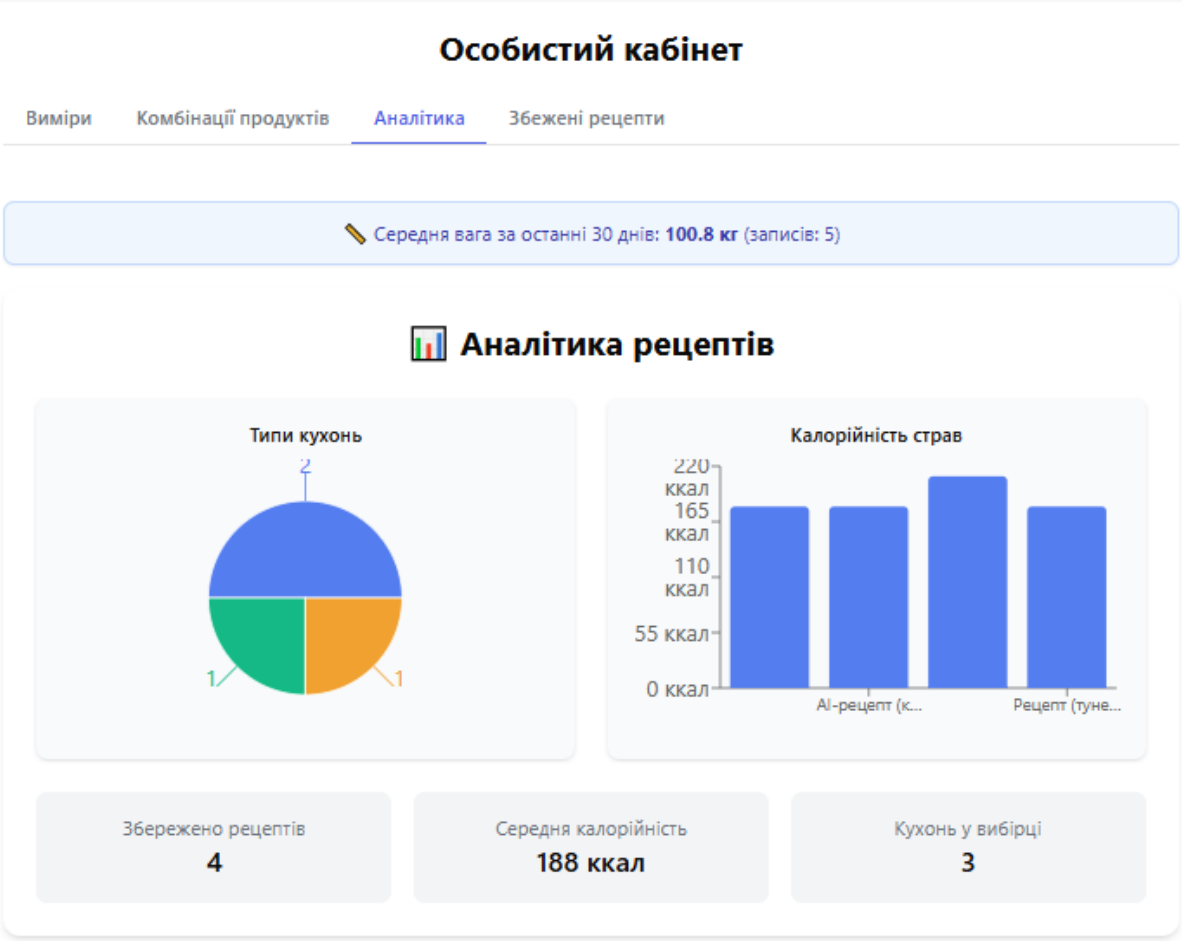


Рис. 4.6 Вкладка аналітики з діаграмою типів кухонь, графіком калорійності страв і ключовими метриками

4.6 Інтерфейс користувача

Інтерфейс користувача застосунку Planner-Calorie реалізовано з урахуванням принципів мінімалізму, адаптивності та доступності. Дизайн орієнтований на зручність використання як на комп’ютерах, так і на мобільних пристроях.

Основна мета — створити мінімалістичне середовище для перегляду КБЖВ продуктів, генерації рецептів та роботи з особистими даними користувача.

Загальна структура сторінки побудована на основі компонента Layout[10],

який об’єднує такі частини:

– Header — фіксована верхня панель з логотипом, навігацією, перемикачем теми (світлої/темної), кнопкою виходу з облікового запису.

– Outlet — динамічна частина, яка підвантажує відповідну сторінку залежно від маршруту (Home, Recipes, Nutrition, Profile тощо).

– Footer — нижня частина інтерфейсу з інформаційним блоком.

Функція перемикання теми реалізована за допомогою стану isDarkTheme, який зберігається у localStorage:

```
const toggleTheme = () => {
  setIsDarkTheme((prev) => {
    const next = !prev;
    localStorage.setItem("theme", next ? "dark" : "light");
    return next;
  });
};
```

Це дозволяє користувачеві вільно змінювати вигляд інтерфейсу відповідно до особистих уподобань. Всі компоненти підтримують Tailwind класи dark: для коректного виводу темної теми.

У Header передбачено два режими відображення меню:

– Десктопний — посилання на сторінки відображаються у горизонтальному ряду.

– Мобільний — при малій ширині активується бургер-меню з анімацією відкриття за допомогою framer-motion.

Адаптивна верстка побудована на Flexbox і Grid, що дозволяє застосунку виглядати коректно на різних пристроях. Крім того, Tailwind CSS забезпечує швидке оформлення з урахуванням світлої і темної тем.

Інтерфейс оформлений в сучасному стилі з м’якими тінями, згладженими кутами (rounded-xl, shadow-md) та анімованими елементами (motion.div). Кольори відповідають асоціативному навантаженню: зелений — для підтвердження дій, червоний — для видалення, синій — для навігації.

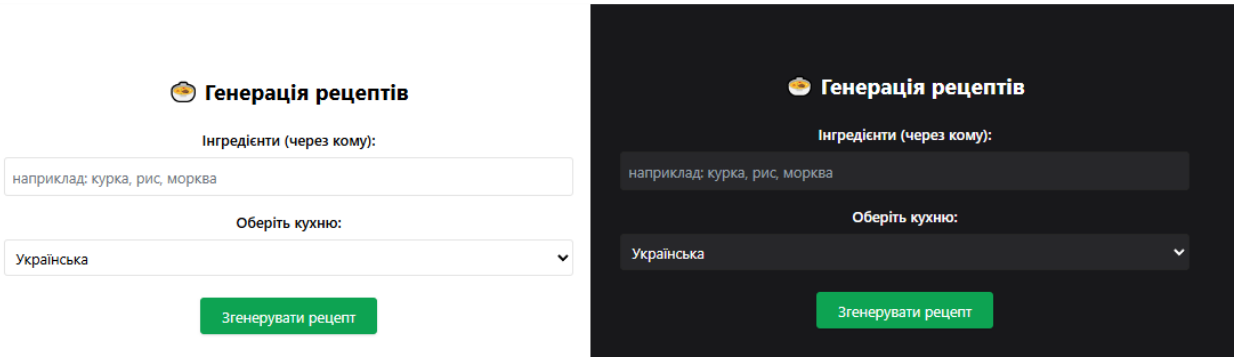


Рис. 4.7 Порівняння світлої та темної теми на головній сторінці

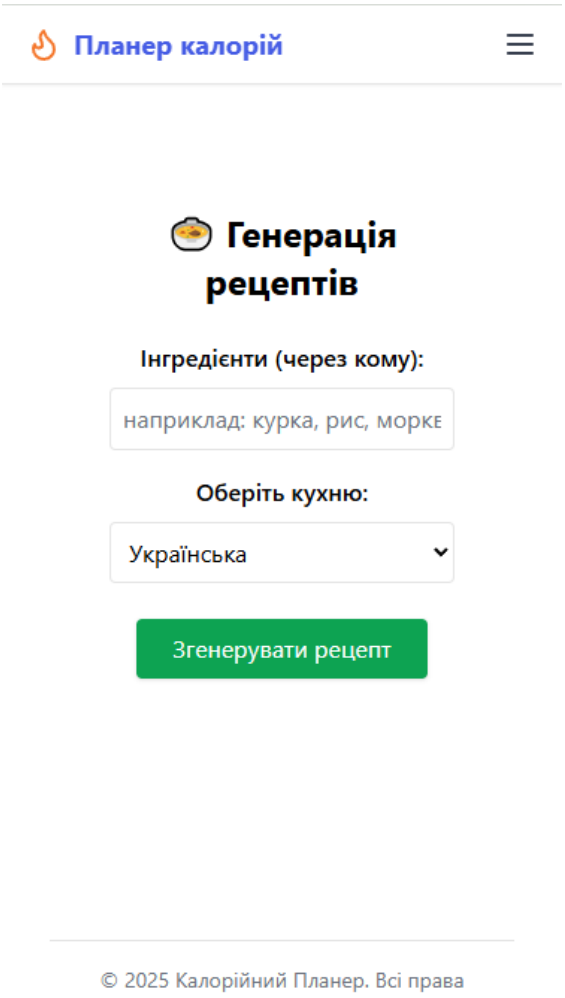


Рис. 4.8 Адаптивна версія інтерфейсу на мобільному пристрої

4.7 Реалізація реєстрації та логіну

У вебзастосунку Planner-Calorie реалізовано повноцінну систему автентифікації користувача за допомогою Supabase Auth, яка базується на email-паролі та підтримує OAuth-авторизацію (наприклад, через GitHub). Це дозволяє

користувачу захищати персональні дані й працювати з індивідуальним кабінетом.

Форма авторизації об'єднує функції входу та реєстрації в одному компоненті Auth.jsx. Перемикання між режимами реалізовано через змінну стану isLogin, а логіка відправки запиту виглядає наступним чином:

```
if (isLogin) {
  result = await supabase.auth.signInWithPassword({ email, password });
} else {
  result = await supabase.auth.signUp({ email, password });
}
```

Після успішної автентифікації користувача система автоматично перенаправляє його до головної сторінки:

```
navigate("/");
```

Для зручності інтерфейс розроблений у сучасному стилі з використанням Tailwind CSS та Framer Motion — з плавними анімаціями, адаптивною верстkою та підтримкою темної теми.[15] Приклад перемикання теми реалізовано через localStorage та React-хук useState:

```
const toggleTheme = () => {
  setIsDarkTheme((prev) => {
    const next = !prev;
    localStorage.setItem("theme", next ? "dark" : "light");
    return next;
  });
};
```

Користувач також бачить повідомлення про помилку при неправильному введенні або у випадку невдалої спроби входу:

```
{error && <p className="text-red-600 text-sm">{error}</p>}
```

Інтерфейс виконано в жовто-сірій гамі (в темному режимі — на базі кольору zinc), що створює комфортне середовище для роботи. Крім того, використання Supabase дозволило уникнути серверної реалізації логіки входу та значно спростити інтеграцію.

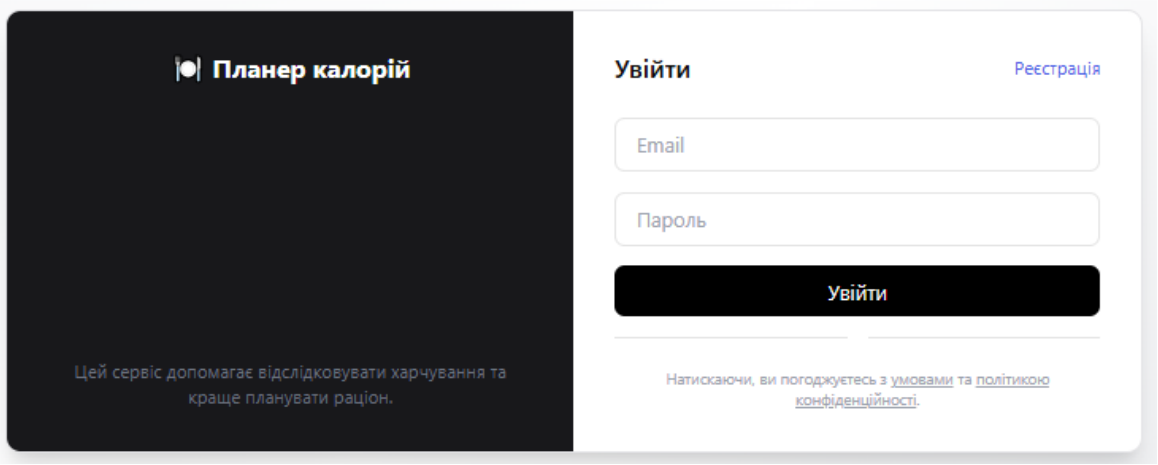


Рис. 4.9 Сторінка авторизації користувача в Planner-Calorie

4.8 Взаємодія з сервером та збереження даних

У проєкті Planner-Calorie клієнтська частина (React) активно взаємодіє з хмарною базою даних, реалізованою за допомогою Supabase — сучасного сервісу, що поєднує функціональність бази PostgreSQL, аутентифікацію, авторизацію та RESTful API.

Усі операції з даними — такі як збереження вимірів, обраних комбінацій, архіву рецептів, цілей користувача — виконуються за допомогою методів, визначених у файлі supabaseClient.js. У ньому ініціалізується клієнтська логіка взаємодії з Supabase через:

export const supabase = createClient(supabaseUrl, supabaseKey);

Основні типи запитів:

Зчитування користувача:

const { data } = await supabase.auth.getUser();

Додавання нового запису (наприклад, виміри):

await supabase.from("user_stats").insert([record]);

Отримання даних з фільтрацією:

```
await supabase.from("favorite_combinations")
  .select("*")
  .eq("user_id", userId)
```

```
  .order("created_at", { ascending: false });
```

Видалення запису:

```
await supabase.from("favorites").delete().eq("id", id);
```

Використовується також функція upsert() для збереження цілей користувача:

```
await supabase.from("user_goals").upsert({ user_id: userId, goal, ... });
```

Завдяки використанню асинхронного підходу (async/await), зміни даних моментально синхронізуються з інтерфейсом, що реалізовано через хук useEffect у відповідних компонентах (Measurements.jsx, MealPlanner.jsx, FavoriteCombos.jsx тощо).

У проєкті не застосовуються локальні тимчасові сховища, як-от глобальні змінні або in-memory списки. Усі зміни (навіть тимчасові) зберігаються безпосередньо у Supabase, що дозволяє забезпечити повну цілісність даних і роботу на будь-якому пристрої після авторизації.

4.9 Тестування роботи застосунку

Для забезпечення стабільної та коректної роботи веб-застосунку Planner-Calorie було проведено базове тестування клієнтської частини з використанням як ручних, так і автоматизованих підходів.

Для модульного тестування було використано фреймворк Vitest спільно з бібліотекою @testing-library/react, що дозволило перевірити рендеринг та функціональність ключових компонентів інтерфейсу[17][18].Тести охоплюють базові UI-взаємодії, перевірку відображення вмісту, зміна станів компонентів та відповідь на події користувача. В якості середовища тестів використовувався емулятор DOM - happy-dom, що забезпечує достатню точність без потреби в реальному браузері. Тести були зосереджені на таких компонентах:

– Recipes.jsx — перевірка введення інгредієнтів, генерації рецепта та виводу КБЖВ.

– Auth.jsx — перевірка вводу email/пароля, перемикання між реєстрацією та входом, валідації та переходу після входу

– Measurements.jsx — тестування форми введення ваги та зросту, натискання кнопки збереження.

– FavoriteCombos.jsx — тестування рендерингу збережених комбінацій та функції видалення (в процесі розширення).

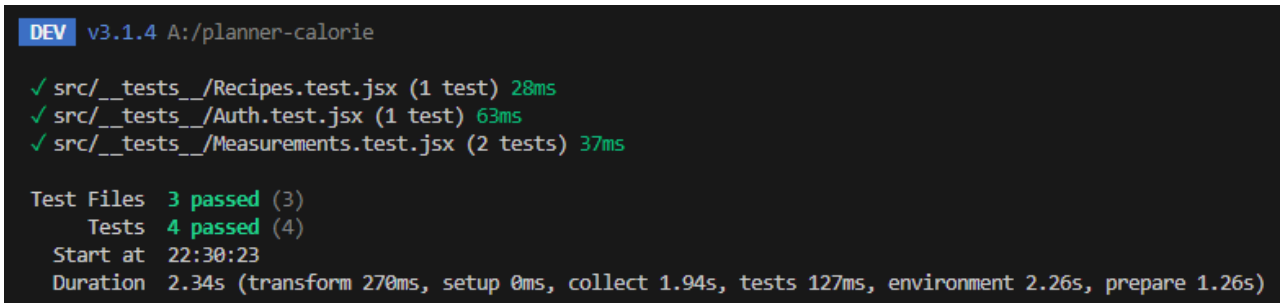


Рис. 4.10 Результат виконання тестів

Окрім автоматизованих тестів, було проведено ручне тестування в середовищі Google Chrome, Mozilla Firefox, з використанням DevTool і реального введення даних. Були перевірені ключові сценарії:

– Окрім автоматизованих тестів, було проведено ручне тестування інтерфейсу та основних сценаріїв використання:

– Успішна реєстрація, авторизація та вихід із системи через Supabase.

– Збереження та вивід згенерованих AI-рецептів через Gemini API.

– Введення та перегляд вимірів користувача (вага/зріст) з відображенням графіку.

– Темна/світла тема перемикається та зберігається між сесіями.

– Коректне відображення даних у вкладках «Рецепти», «Профіль», «Планування меню», «Аналітика».

– Усі запити до Supabase API завершуються без помилок, зміни одразу синхронізуються з інтерфейсом.

Згідно з результатами тестування, функціональність основних компонентів застосунку відповідає очікуванням. Веб-застосунок працює стабільно як у світлій, так і в темній темі, забезпечує правильну обробку користувацьких даних та взаємодію з API, а також має адаптивний і зручний інтерфейс.

ВИСНОВКИ

У результаті виконання дипломної роботи було розроблено веб-застосунок Planner-Calorie, призначений для зручного ведення обліку харчування, збереження КБЖВ-комбінацій, генерації рецептів та відстеження фізичних показників користувача.

Під час роботи над проєктом було виконано повний цикл розробки: від дослідження предметної області до створення, тестування та документування готової системи. У рамках аналізу проблематики було розглянуто існуючі аналоги на ринку, визначено їх переваги та недоліки, що дозволило сформулювати власні вимоги до функціоналу застосунку. На основі цього було реалізовано такі ключові функції: генерація AI-рецептів, розрахунок КБЖВ з використанням Nutritionix API, збереження комбінацій і страв, аналітика змін у фізичних показниках та аналітика збережених AI-рецептів.

Інтерфейс користувача був спроєктований у Figma з урахуванням принципів адаптивності, мінімалізму. Після цього дизайн було реалізовано у середовищі React із використанням Tailwind CSS та бібліотеки компонентів shadcn/ui. Було реалізовано підтримку темної та світлої теми, а також сучасну анімацію для покращення користувацького досвіду. Уся логіка додатку структурована у вигляді окремих вкладок: Рецепти, Збережені рецепти, Комбінації продуктів, Виміри, Аналітика, Головна, Кабінет.

Для збереження даних і автентифікації користувачів використано Supabase, який також виступає як хмарне сховище для історії змін, архіву рецептів і улюбленних комбінацій. Для генерації AI-рецептів застосовується Gemini API, що дозволяє створювати персоналізовані страви відповідно до вподобань користувача.

З метою забезпечення стабільної роботи системи було проведено ручне та модульне тестування. Основні клієнтські компоненти — такі як Recipes, Auth, Measurements, FavoriteCombos — були протестовані з використанням бібліотек Vitest і @testing-library/react, що дозволило переконатися у коректності рендерингу

елементів, передачі даних і стабільності взаємодії з користувачем. Усі тести були успішно виконані, про що свідчать відповідні результати

Таким чином, проєкт Planner-Calorie повністю відповідає поставленим вимогам. Застосунок забезпечує мінімалістичний інтерфейс, стабільну роботу.

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

1. Шаповал С.О., Колодюк А.В. Визначення вимог до додатку-планера калорій з інтегруванням штучного інтелекту. Матеріали конференції "Застосування програмного забезпечення в ІКТ", 24.04.2025, Київ, Державний університет інформаційно-комунікаційних технологій, Збірник тез. К.:ДУІКТ, 2025. С.353-356.

2. Шаповал С.О., Колодюк А.В. Застосунок-планер калорій з інтегруванням штучного інтелекту. Матеріали V Всеукраїнської науково-практичної конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15.05.2025, Київ, Державний університет інформаційно-комунікаційних технологій, Збірник тез. К.:ДУІКТ, 2025. (подано до друку)

ПЕРЕЛІК ПОСИЛАНЬ

1. Nutritionix API Documentation. URL: https://developer.nutritionix.com/docs/v2 (date of access: 02.03.2025).
2. WebMD. Calorie and Nutrition Database. URL: https://www.webmd.com/diet/healthtool-food-calorie-counter (date of access: 13.03.2025).
3. Healthline. How to Count Macros: A Step-by-Step Guide. URL: https://www.healthline.com/nutrition/how-to-count-macros (date of access: 14.03.2025).
4. Figma. Collaborative Interface Design Tool. URL: https://figma.com (date of access: 19.03.2025).
5. Gemini API Documentation. Google AI. URL: https://ai.google.dev/gemini-api/docs (date of access: 24.03.2025).
6. Supabase Documentation. URL: https://supabase.com/docs (date of access: 25.03.2025).
7. PostgreSQL documentation. URL: https://www.postgresql.org/docs/ (date of access: 25.03.2025).
8. Schönig H.-J. Mastering PostgreSQL 13. 4th ed. Birmingham: Packt Publishing, 2020. 476 p.
9. React Documentation. Meta Platforms Inc. URL: https://react.dev (date of access: 27.03.2025).
10. Bertoli M. React Design Patterns and Best Practices. Birmingham: Packt Publishing, 2016. 300 p.
11. Tailwind CSS Documentation. URL: https://tailwindcss.com/docs (date of access: 30.03.2025).
12. Rappin N. Modern CSS with Tailwind. 2nd ed. Pragmatic Bookshelf, 2023. 200 p.

13. Express.js Official Website. URL: https://expressjs.com/ (date of access: 08.04.2025).
14. Holowaychuk T. Mastering Express.js. URL: https://visionmedia.github.io/masteringnode/book2/ (date of access: 12.04.2025).
15. Recharts — A composable charting library built on React components. URL: https://recharts.org/en-US (date of access: 13.04.2025).
16. Framer Motion Docs. URL: https://www.framer.com/motion/ (date of access: 18.04.2025).
17. Vitest Docs. URL: https://vitest.dev (date of access: 26.04.2025).
18. Testing Library Docs. URL: https://testing-library.com/docs/react-testing-library/intro/ (date of access: 27.04.2025).
19. Happy-dom – Node.js implementation of the DOM and HTML standards Docs. URL: https:/npmjs.com/package/happy-dom (date of access: 27.04.2025)
20. DOM-дерево. URL: https://uk.javascript.info/dom-nodes (date of access: 28.04.2025)

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка персоналізованого планеру калорій на основі меню кухонь світу з використанням штучного інтелекту

Виконав студент 4 курсу
групи ПД-42
Шаповал Сергій Олександрович
Керівник роботи
асистент кафедри, Колодюк Андрій Васильович

Київ – 2025

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: спрощення процесу планування персоналізованого раціону харчування користувача.

Об’єкт дослідження: процес підбору продуктів харчування з урахуванням їх харчової цінності та кулінарних вподобань користувача.

Предмет дослідження: веб-застосунок для підбору та рекомендації страв різних кухонь світу з урахуванням харчової цінності продуктів і особистих переваг користувача за допомогою методів штучного інтелекту.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати сучасні підходи до автоматизації харчового планування, визначити переваги та недоліки існуючих веб-застосунків для контролю КБЖВ.
2. Дослідити методи інтеграції зовнішніх API (Nutrition, Gemini) для отримання актуальної інформації про продукти та AI-генерації рецептів.
3. Визначити функціональні та нефункціональні вимоги до майбутнього додатку.
4. Визначити програмні засоби реалізації.
5. Сформувати архітектуру веб-застосунку, що поєднує пошук і підбір продуктів, AI-генерацію страв, ведення історії вимірів тіла та аналітику раціону.
6. Реалізувати захищену взаємодію клієнтської та серверної частин через API-запити, організувати надійне зберігання персональних даних на базі Supabase(PostgreSQL).
7. Провести тестування прототипу веб-застосунку, проаналізувати результати щодо стабільності, генерації AI-рецептів та точності розрахунків КБЖВ.

АНАЛІЗ АНАЛОГІВ

Показник	MyFitnessPal	MealPrepPro	Samsung Food	Planner-calorie
Платформи	Android, iOs, Web	iOs	Android, iOs, Web	Web(PWA: моб/десктоп)
Ведення харчового щоденника	+	+-(обмежено)	-	+
Розрахунок КБЖВ	+	+	+-(лише рецепти)	+(по інгрідієнтам)
Додавання рецептів	-	-	+	+(AI-генерація)
База продуктів	+(14млн.)	-	+-(рецепти з Web)	+(Nutrition API/UA)
Графік змін ваги	+-(лише у Premium версії)	-	-	+
AI-генерація рецептів	-	-	-	+(Gemini API)
Підтримка української мови	-	-	+-(частково)	+
Вартість	+-(є безкоштовна та Premium версія)	доступна лише платна версія -	безкоштовна +	безкоштовна +
Захист персональних даних	+(OAuth2/HTTPS)	+–(Apple ID)	+(Samsung Account/HTTPS)	+(JWT/HTTPS/Supabase)

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

1. Пошук КБЖВ (калорії, білки, жири, вуглеводи) конкретних продуктів.
2. Відображення таблиць харчової цінності.
3. Інтеграція ШІ для генерації ідей страв з різних кухонь світу, базуючись на інтересах користувача.
4. Виведення згенерованих страв у текстовому форматі з коротким описом або рецептом.
5. Ведення історії вимірів тіла(зріст,вага)

Нефункціональні вимоги:

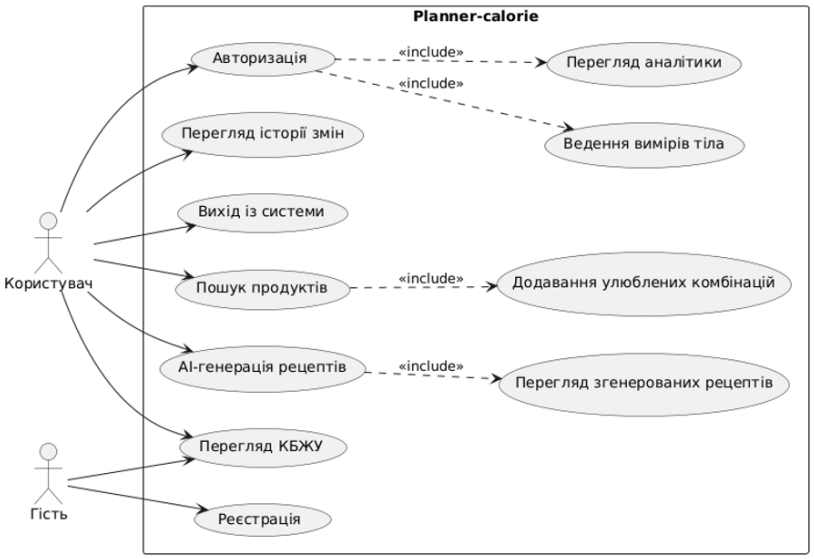
1. Час відповіді основних API (Nutritionix, Gemini) не повинен перевищувати 5 секунд.
2. Актуальність даних (автоматичне оновлення через API при додаванні нових продуктів).
3. Адаптивність до мобільних пристроїв.
4. Захист персональних даних користувача (JWT, захищені маршрути).

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

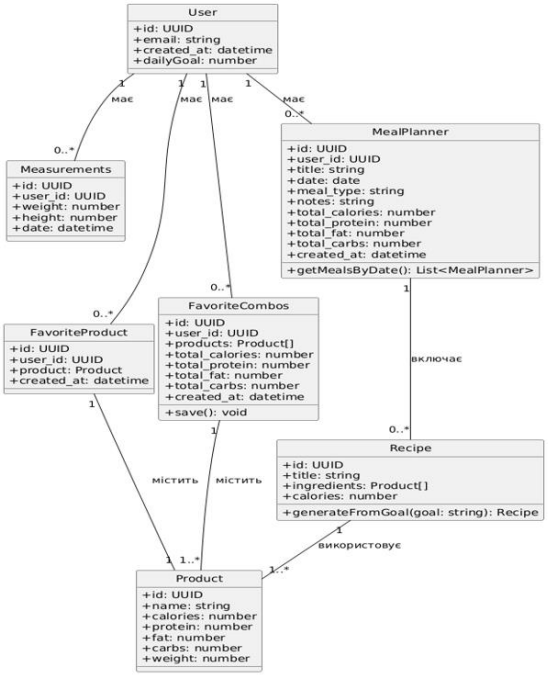
6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



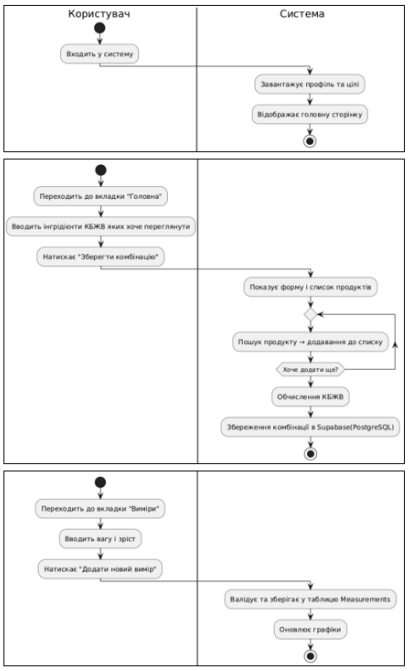
7

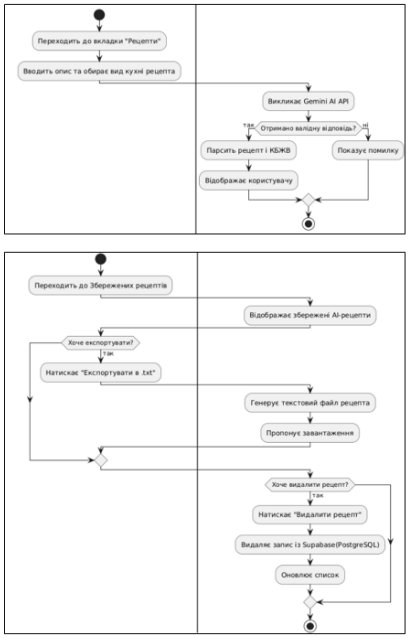
ДІАГРАМА КЛАСІВ



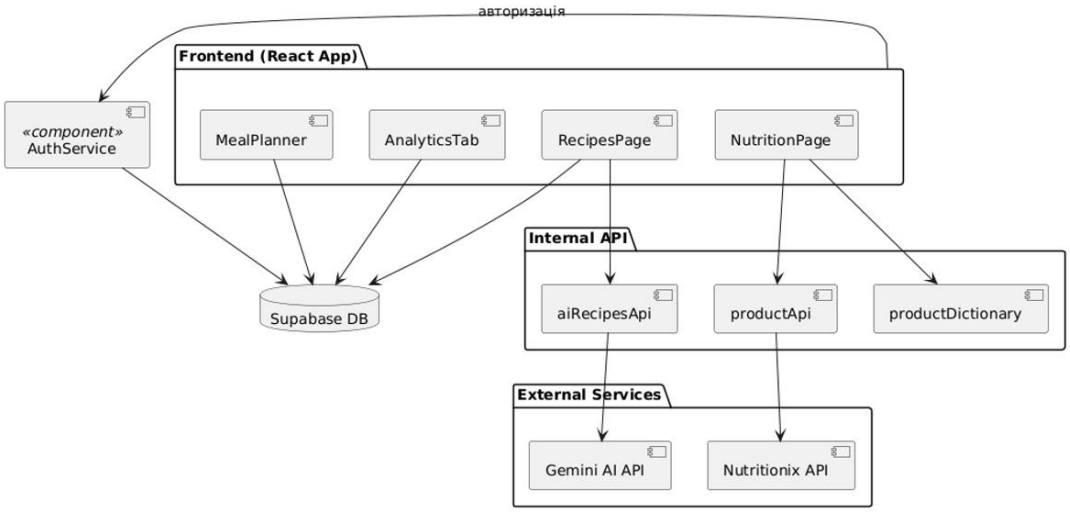
8

ДІАГРАМА ДІЯЛЬНОСТІ ПЛАНЕРУ-КАЛОРІЙ

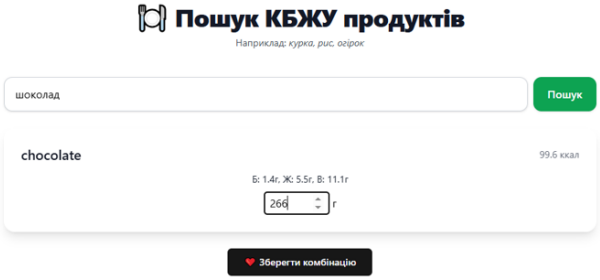




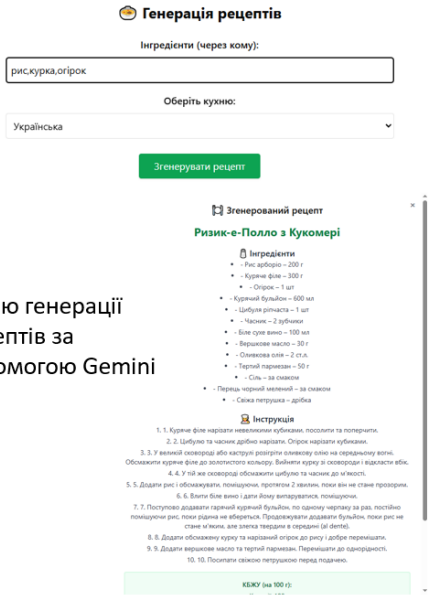
ДІАГРАМА КОМПОНЕНТІВ



ЕКРАННІ ФОРМИ



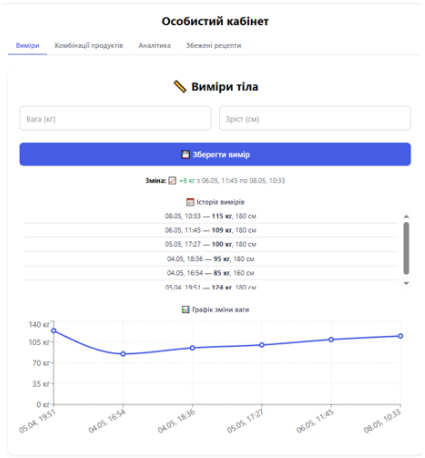
Основна частина сайту – пошук КБЖВ продуктів



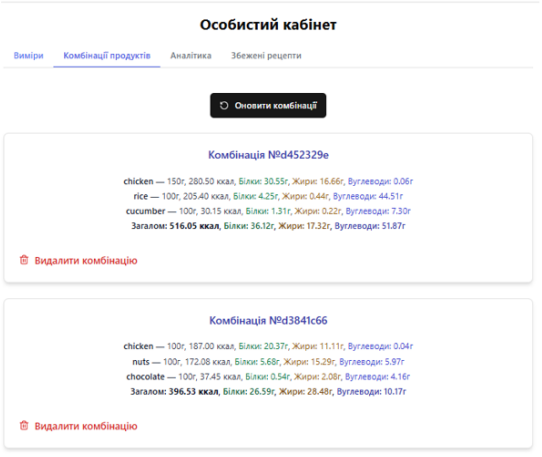
Меню генерації рецептів за допомогою Gemini

11

ЕКРАННІ ФОРМИ



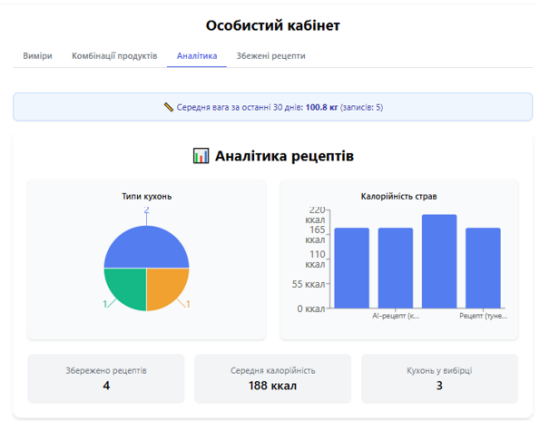
Форма введення та аналізу вимірів тіла



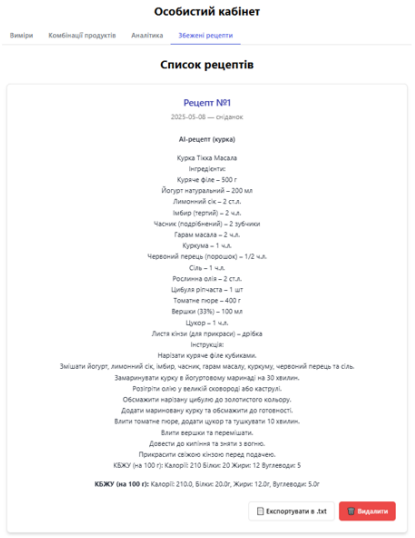
Список збережених комбінацій продуктів

12

ЕКРАННІ ФОРМИ

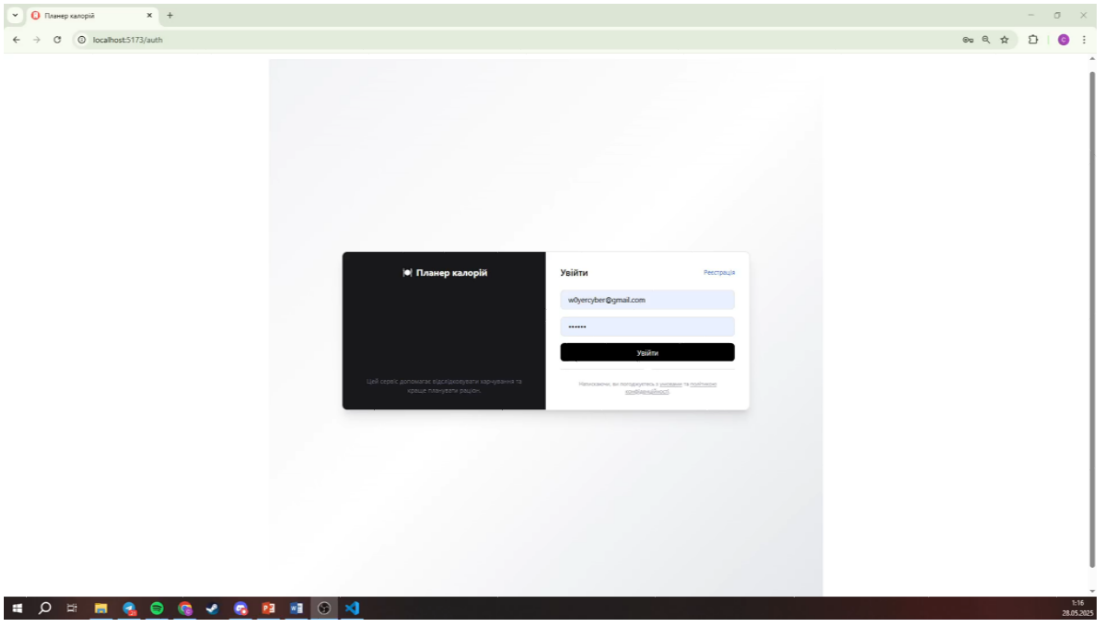


Вигляд аналітики збережених рецептів



Детальний перегляд згенерованих AI-рецептів

ДЕМОНСТРАЦІЯ РОБОТИ ПРОГРАМИ



АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Шаповал С.О., Колодюк А.В. Визначення вимог до додатку-планера калорій з інтегруванням штучного інтелекту. Матеріали конференції "Застосування програмного забезпечення в ІКТ", 24.04.2025, Київ, Державний університет інформаційно-комунікаційних технологій, Збірник тез. К.:ДУІКТ, 2025. С.353-356.
2. Шаповал С.О., Колодюк А.В. Застосунок-планер калорій з інтегруванням штучного інтелекту. Матеріали V Всеукраїнської науково-практичної конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15.05.2025, Київ, Державний університет інформаційно-комунікаційних технологій, Збірник тез. К.:ДУІКТ, 2025. (подано до друку)

15

ВИСНОВКИ

1. У ході дослідження проаналізовано сучасні рішення у сфері цифрового контролю харчування та інструментів для відстеження КБЖВ, що дозволило визначити основні функціональні вимоги до веб-платформи для ведення харчового щоденника.
2. Проведено аналіз можливостей і обмежень популярних API-сервісів (Gemini та Nutritionix), результати якого враховані при проектуванні архітектури системи та структури зберігання харчових даних.
3. Реалізовано функціональну веб-платформу, яка дозволяє генерувати AI-рецепти на основі заданих параметрів, зберігати індивідуальні комбінації продуктів, вести облік вимірів тіла та переглядати аналітику у вигляді графіків. Реалізовано аналітичні модулі, що візуалізують зміну ваги, калорійність страв та видом кухні.
4. Побудовано клієнтську та серверну частини застосунку з використанням архітектури, що базується на модульності, REST-запитах та взаємодії з реляційною базою даних Supabase(PostgreSQL).
5. Проведено тестування функціоналу, що підтвердило стабільність, узгодженість даних і коректність взаємодії між модулями.

16

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Код pages/Auth.jsx

```
import React, { useState } from "react";
import { supabase } from "../services/supabaseClient";
import { useNavigate } from "react-router-dom";
import { Github } from "lucide-react";
import { motion } from "framer-motion";

const Auth = () => {
const navigate = useNavigate();
const [email, setEmail] = useState("");
const [password, setPassword] = useState("");
const [isLogin, setIsLogin] = useState(true);
const [error, setError] = useState("");

const handleAuth = async () => {
setError("");
let result;
if (isLogin) {
result = await supabase.auth.signInWithPassword({ email, password });
} else {
result = await supabase.auth.signUp({ email, password });
}
const { error } = result;
if (error) setError(error.message);
else navigate("/");
};
const handleOAuth = async () => {
const { error } = await supabase.auth.signInWithOAuth({ provider: "github" });
if (error) setError(error.message);
};

return (
<div className="min-h-screen flex items-center justify-center bg-gradient-to-br from-gray-100 via-white to-gray-200 dark:from-zinc-900 dark:to-black px-4">
<motion.div
initial={{ opacity: 0, y: 20 }}
animate={{ opacity: 1, y: 0 }}
transition={{ duration: 0.5 }}
className="bg-white dark:bg-zinc-900 w-full max-w-4xl grid grid-cols-1 md:grid-cols-2 shadow-xl rounded-xl overflow-hidden border border-zinc-200 dark:border-zinc-700"
>
<div className="bg-zinc-900 text-white p-8 flex flex-col justify-between dark:bg-zinc-800">
<div>
<h1 className="text-xl font-bold mb-4">□ Планер калорій</h1>
<p className="text-gray-400">
</p>
</div>
<p className="mt-6 text-sm text-gray-500">Цей сервіс допомагає відслідковувати харчування та краще планувати раціон.</p>
</div>

<div className="p-8 dark:text-white">
<div className="flex justify-between mb-6">
<h2 className="text-xl font-semibold">
{isLogin ? "Увійти" : "Створити акаунт"}
</h2>
<button
onClick={() => setIsLogin(!isLogin)}
className="text-blue-600 dark:text-blue-400 text-sm hover:underline"
>
{isLogin ? "Реєстрація" : "Вхід"}
</button>
</div>

<div className="space-y-4">
<input
type="email"
placeholder="Email"
value={email}
onChange={(e) => setEmail(e.target.value)}
className="w-full px-4 py-2 border rounded-lg focus:ring-2 focus:ring-blue-500 dark:bg-zinc-800 dark:border-zinc-700"
/>
<input
type="password"
placeholder="Пароль"
value={password}
onChange={(e) => setPassword(e.target.value)}
className="w-full px-4 py-2 border rounded-lg focus:ring-2 focus:ring-blue-500 dark:bg-zinc-800 dark:border-zinc-700"
/>
{error && <p className="text-red-600 text-sm">{error}</p>}
<button
onClick={handleAuth}
className="w-full bg-black dark:bg-white text-white dark:text-black py-2 rounded-lg hover:opacity-90 transition"
>
{isLogin ? "Увійти" : "Зареєструватися"}
</button>

<div className="flex items-center gap-2 text-gray-400">
<div className="flex-1 border-t"></div>
<span className="text-sm"></span>
<div className="flex-1 border-t"></div>
</div>



</div>

<p className="text-xs text-gray-400 mt-6 text-center dark:text-gray-500">
Натискаючи, ви погоджуєтесь з <span className="underline">умовами</span> та <span className="underline">політикою конфіденційності</span>.
</p>
</div>
</motion.div>
</div>
);
};

export default Auth;
```

Код pages/ Nutrition.jsx

```
import React, { useState } from "react";
```

```
import SaveCombinationButton from
"../components/profile/SaveCombinationButton";
import { PRODUCT_TRANSLATIONS } from
"../services/productDictionary";

const Nutrition = () => {
  const [query, setQuery] = useState("");
  const [selected, setSelected] = useState([]);
  const [weight, setWeight] = useState("");
  const [results, setResults] = useState([]);

  const handleSearch = () => {
    const terms = query.split(",").map((q) =>
q.trim().toLowerCase());
    const found = terms
      .map((term) => PRODUCT_DICTIONARY.find((p) =>
p.name.toLowerCase() === term))
      .filter(Boolean);
    setResults(found);
  };

  const handleAdd = (product) => {
    const grams = parseFloat(weight);
    if (!grams || grams <= 0) return;

    const ratio = grams / 100;
    setSelected((prev) => [
      ...prev,
      {
        ...product,
        grams,
        calories: +(product.calories * ratio).toFixed(2),
        protein: +(product.protein * ratio).toFixed(2),
        fat: +(product.fat * ratio).toFixed(2),
        carbs: +(product.carbs * ratio).toFixed(2),
      },
    ]);
    setWeight("");
  };

  const totals = selected.reduce(
    (acc, p) => {
      acc.grams += p.grams;
      acc.calories += p.calories;
      acc.protein += p.protein;
      acc.fat += p.fat;
      acc.carbs += p.carbs;
      return acc;
    },
    { grams: 0, calories: 0, protein: 0, fat: 0, carbs: 0 }
  );

  return (
    <div className="p-6 max-w-4xl mx-auto">
      <h1 className="text-2xl font-bold mb-4">КБЖВ
Продуктів</h1>

      <div className="mb-4">
        <input
          type="text"
          value={query}
          onChange={(e) => setQuery(e.target.value)}
          placeholder="Введіть продукти через кому
(наприклад: рис, курка, огірок)"
          className="border p-2 w-full mb-2"
        />
        <button onClick={handleSearch} className="bg-blue-
500 text-white px-4 py-2">
          Пошук
        </button>
      </div>

      {results.map((product, i) => (
        <div key={i} className="border p-2 my-2 rounded">
          <p className="font-medium">Назва:
{product.name}</p>
          <input
            type="number"
            value={weight}
            onChange={(e) => setWeight(e.target.value)}
            placeholder="Вага (г)"
            className="border p-1 my-2 mr-2 w-32"
          />
          <button
            onClick={() => handleAdd(product)}
            className="bg-green-500 text-white px-3 py-1"
          >
            Додати
          </button>
        </div>
      ))}

      {selected.length > 0 && (
        <div className="mt-6">
          <h2 className="text-xl font-semibold mb-2">Підібрані
продукти</h2>
          <table className="table-auto w-full border">
            <thead>
              <tr>
                <th>Назва</th>
                <th>Вага (г)</th>
                <th>Калорії</th>
                <th>Білки</th>
                <th>Жири</th>
                <th>Вуглеводи</th>
              </tr>
            </thead>
            <tbody>
              {selected.map((item, i) => (
                <tr key={i}>
                  <td>{item.name}</td>
                  <td>{item.grams}</td>
                  <td>{item.calories}</td>
                  <td>{item.protein}</td>
                  <td>{item.fat}</td>
                  <td>{item.carbs}</td>
                </tr>
              ))}
              <tr className="font-bold border-t">
                <td>Разом:</td>
                <td>{totals.grams}</td>
                <td>{totals.calories.toFixed(2)}</td>
                <td>{totals.protein.toFixed(2)}</td>
                <td>{totals.fat.toFixed(2)}</td>
                <td>{totals.carbs.toFixed(2)}</td>
              </tr>
            </tbody>
          </table>

          <div className="mt-4">
            <SaveCombinationButton combination={selected} />
          </div>
        </div>
      )}
    </div>
  );
};
```

```
export default Nutrition;
```

Код services/productApi.js

```
import axios from "axios";
import { PRODUCT_TRANSLATIONS } from
"./productDictionary";

const APP_ID =
import.meta.env.VITE_NUTRITIONIX_APP_ID;
const APP_KEY =
import.meta.env.VITE_NUTRITIONIX_APP_KEY;

export const searchProduct = async (rawInput) => {
  const productList = rawInput
    .split(",")
    .map((p) => p.trim().toLowerCase())
    .filter(Boolean);

  const results = [];

  for (const product of productList) {
    const translated = PRODUCT_TRANSLATIONS[product] ||
product;

    try {
      const response = await axios.post(
        "https://trackapi.nutritionix.com/v2/natural/nutrients",
        { query: `1 ${translated}` },
        {
          headers: {
            "x-app-id": "776a978b",
            "x-app-key": "69f8463f227a98f9cc0740687261229b",
            "Content-Type": "application/json"
          }
        }
      );

      const items = response.data.foods.map((item) => ({
        name: item.food_name,
        calories: item.nf_calories,
        protein: item.nf_protein,
        fat: item.nf_total_fat,
        carbs: item.nf_total_carbohydrate
      }));

      results.push(...items);
    } catch (error) {
      console.warn(`Помилка продукту "${product}":`,
error.response?.data || error.message);
    }
  }

  return results;
};
```

Код profile/SaveCombinationButton.jsx

```
import React, { useState } from "react";
import { supabase } from "../../services/supabaseClient";
import { Button } from "@/components/ui/button";

const SaveCombinationButton = ({ user, inputValue = [],
onClear, refreshCombos }) => {
  const [saving, setSaving] = useState(false);
  const [saved, setSaved] = useState(false);

  const calculateTotals = (products) => {
    return products.reduce(
      (acc, item) => ({
        calories: acc.calories + (item.calories * item.weight) / 100,
        protein: acc.protein + (item.protein * item.weight) / 100,
        fat: acc.fat + (item.fat * item.weight) / 100,
        carbs: acc.carbs + (item.carbs * item.weight) / 100,
      }),
      { calories: 0, protein: 0, fat: 0, carbs: 0 }
    );
  };

  const handleSave = async () => {
    if (!user || !Array.isArray(inputValue) || inputValue.length
=== 0) return;

    setSaving(true);
    setSaved(false);

    try {
      const enriched = inputValue.map((item) => ({
        name: item.name,
        calories: (item.calories * item.weight) / 100,
        protein: (item.protein * item.weight) / 100,
        fat: (item.fat * item.weight) / 100,
        carbs: (item.carbs * item.weight) / 100,
        weight: item.weight,
      }));

      const totals = calculateTotals(inputValue);

      const { error } = await
supabase.from("favorite_combinations").insert([
        {
          user_id: user.id,
          combination: enriched,
          total_calories: totals.calories,
          total_protein: totals.protein,
          total_fat: totals.fat,
          total_carbs: totals.carbs,
        },
      ]);

      if (error) {
        console.error("❌Помилка вставки:", error);
      } else {
        console.log("✅Комбінація успішно збережена");
        setSaved(true);
        onClear?.();
        refreshCombos?.();
      }
    } catch (error) {
      console.error("❌Помилка при збереженні:", error);
    } finally {
      setSaving(false);
      setTimeout(() => setSaved(false), 2500);
    }
  };

  return (
    <div className="mt-6 flex flex-col items-center space-y-
2">
      <Button
        onClick={handleSave}
        disabled={saving}
        className="px-6 py-2"
      >
        {saving ? "Збереження..." : "□ Зберегти комбінацію"}
      </Button>
      {saved && (
```

```
      <p className="text-green-600 text-sm">Комбінацію збережено ✅</p>
    )}
   </div>
  );
```

```
};

export default SaveCombinationButton;
```

Код pages/Recipes.jsx

```
import React, { useState } from "react";
import { generateRecipe } from "../services/aiRecipesApi";
import { useNavigate } from "react-router-dom";
import { getCurrentUser, supabase } from "../services/supabaseClient";
const parseNutritionFromText = (text) => {
  const pattern = /КБЖВ.*?Калор[ії]ї[:\-\s]*([\d,.]+).*?Білк[иі][:\-\s]*([\d,.]+).*?Жир[и][:\-\s]*([\d,.]+).*?Вуглевод[и][:\-\s]*([\d,.]+)/is;
  const match = text.match(pattern);
  if (!match) return null;
  return {
    calories: parseFloat(match[1].replace(",", ".")) || 0,
    protein: parseFloat(match[2].replace(",", ".")) || 0,
    fat: parseFloat(match[3].replace(",", ".")) || 0,
    carbs: parseFloat(match[4].replace(",", ".")) || 0,
  };
};

const Recipes = () => {
  const [ingredients, setIngredients] = useState("");
  const [cuisine, setCuisine] = useState("Українська");
  const [generatedText, setGeneratedText] = useState("");
  const [loading, setLoading] = useState(false);
  const navigate = useNavigate();
  const handleGenerate = async () => {
    if (!ingredients.trim()) return;
    setLoading(true);
    const result = await generateRecipe(ingredients.split(","), cuisine);
    setGeneratedText(result);
    setLoading(false);
  };
  const handleSave = async () => {
    const user = await getCurrentUser();
    if (!user) return alert("Не вдалося визначити користувача");

    const nutrition = parseNutritionFromText(generatedText);

    const plan = {
      user_id: user.id,
      date: new Date().toISOString().slice(0, 10),
      meal_type: "Страва",
      title: `Рецепт (${ingredients})`,
      notes: generatedText,
      kitchen: cuisine,
      total_calories: nutrition?.calories || 0,
      total_protein: nutrition?.protein || 0,
      total_fat: nutrition?.fat || 0,
      total_carbs: nutrition?.carbs || 0,
      items: [],
    };
    const { error } = await supabase.from("meal_plans").insert([plan]);
    if (error) {
      console.error("❌Supabase помилка:", error);
      alert("Не вдалося зберегти рецепт ❌");
    } else {
      alert("✅Рецепт збережено в MealPlanner!");
      setGeneratedText("");
      navigate("/profile");
    }
  };
  const renderRecipePreview = () => {
    const lines = generatedText.split("\n").filter(Boolean);
    const title = lines[0] || "";
    const ingredientsStart = lines.findIndex((line) => line.toLowerCase().includes("інгредієнти"));
    const instructionStart = lines.findIndex((line) => line.toLowerCase().includes("інструкц"));
    const kbjuStart = lines.findIndex((line) => line.toLowerCase().includes("кбжв"));

    return (
      <div className="space-y-4">
        <h2 className="text-2xl font-bold text-center text-green-700">{title}</h2>

        {ingredientsStart !== -1 && instructionStart !== -1 && (
          <div>
            <h3 className="text-lg font-semibold text-gray-800 dark:text-gray-100">🧂 Інгредієнти</h3>
            <ul className="list-disc list-inside text-sm text-gray-700 dark:text-gray-300 space-y-1">
              {lines.slice(ingredientsStart + 1, instructionStart).map((line, idx) => (
                <li key={idx}>{line}</li>
              ))}
            </ul>
          </div>
        )}

        {instructionStart !== -1 && (
          <div>
            <h3 className="text-lg font-semibold text-gray-800 dark:text-gray-100">👩‍🍳 Інструкція</h3>
            <ol className="list-decimal list-inside text-sm text-gray-700 dark:text-gray-300 space-y-1">
              {lines.slice(instructionStart + 1, kbjuStart !== -1 ? kbjuStart : undefined).map((line, idx) => (
                <li key={idx}>{line}</li>
              ))}
            </ol>
          </div>
        )}

        {kbjuStart !== -1 && (
          <div className="bg-green-50 dark:bg-green-900 border border-green-200 dark:border-green-700 rounded p-4 text-sm text-green-800 dark:text-green-200">
            <h3 className="font-semibold mb-1">КБЖВ (на 100 г):</h3>
            {lines.slice(kbjuStart + 1).map((line, idx) => (
              <div key={idx}>{line}</div>
            ))}
          </div>
        )}
      </div>
    );
  };

  return (
```

```
   <div className="max-w-2xl mx-auto p-6 text-black
dark:text-white">
     <h1 className="text-2xl font-bold mb-6 text-center">
Генерація рецептів</h1>
     <label className="block mb-2 font-medium text-black
dark:text-white">Інгредієнти (через кому):</label>
     <input
       value={ingredients}
       onChange={(e) => setIngredients(e.target.value)}
       placeholder="наприклад: курка, рис, морква"
       className="border p-2 rounded w-full mb-4
                  bg-white text-black placeholder-gray-500
                  dark:bg-zinc-800 dark:text-white dark:placeholder-
gray-400"
     />

     <label className="block mb-2 font-medium text-black
dark:text-white">Оберіть кухню:</label>
     <select
       value={cuisine}
       onChange={(e) => setCuisine(e.target.value)}
       className="border p-2 rounded w-full mb-6
                  bg-white text-black
                  dark:bg-zinc-800 dark:text-white"
     >
       <option>Українська</option>
       <option>Італійська</option>
       <option>Азійська</option>
       <option>Індійська</option>
     </select>

     <button
       onClick={handleGenerate}
       className="bg-green-600 text-white px-6 py-2 rounded
shadow hover:bg-green-700 transition"
       disabled={loading}
     >
       {loading ? "Генерація..." : "Згенерувати рецепт"}
     </button>

     {generatedText && (
       <div className="fixed inset-0 bg-black bg-opacity-50
flex items-center justify-center z-50">
         <div className="bg-white dark:bg-zinc-900 max-w-2xl
w-full rounded-lg p-6 shadow-lg relative overflow-y-auto max-
h-[90vh] text-black dark:text-white">
           <button
             onClick={() => setGeneratedText("")}
             className="absolute top-3 right-3 text-gray-500
hover:text-red-500 text-lg font-bold"
           >
             ×
           </button>

           <h3 className="font-semibold mb-4 text-xl text-
center">Згенерований рецепт</h3>
           <div className="text-sm leading-relaxed space-y-4">
             {renderRecipePreview()}
           </div>

           <button
             onClick={handleSave}
             className="mt-6 w-full bg-blue-600 hover:bg-blue-
700 text-white py-2 rounded"
           >
             +Додати у MealPlanner
           </button>
         </div>
       </div>
     )}
   </div>
  );
};

export default Recipes;
```

Код services/aiRecipesApi.ts

```
const GEMINI_API_KEY =
import.meta.env.VITE_GEMINI_KEY;
const GEMINI_URL =
`https://generativelanguage.googleapis.com/v1beta/models/gem
ini-2.0-flash:generateContent?key=${GEMINI_API_KEY}`;

export const generateRecipe = async (ingredients, cuisine) => {
  const ingredientsList = Array.isArray(ingredients) ?
ingredients.join(", ") : ingredients;


  const ukrainianHint = cuisine === "Українська"
    ? " (врахуй традиційні страви — борщ, вареники,
деруни, печеня, узвар, голубці)"
    : "";

  const prompt = `
Склади детальний і граматично правильний рецепт у стилі
"${cuisine}"${ukrainianHint}.
Використай інгредієнти: ${ingredientsList}.

Результат має містити:
1. Назву страви (на окремому рядку).
2. Список інгредієнтів з одиницями (г, мл, шт тощо), у
форматі:
Інгредієнти:
- Продукт 1 – 100 г
- Продукт 2 – 2 шт
3. Покрокову інструкцію у вигляді списку.
4. КБЖВ у форматі (на окремому рядку кожен):
КБЖВ (на 100 г):
Калорії: 350
Білки: 20
Жири: 15
Вуглеводи: 30

❗ Не додавай пояснень, прикрас, символів чи додаткових
коментарів.
Увесь текст — українською мовою.
`;

  try {
    const response = await fetch(GEMINI_URL, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        contents: [{ parts: [{ text: prompt }] }],
      }),
    });

    const data = await response.json();
    const content =
data?.candidates?.[0]?.content?.parts?.[0]?.text;

    if (content) return content.trim();
    throw new Error(data.error?.message || "Порожній
результат");
  } catch (error) {
    console.error("Gemini API error:", error);
    return `Помилка генерації: ${error.message}`;
  }
};
```

Код profile/Measurements.jsx

```
import React from "react";
import {
  LineChart,
  Line,
  XAxis,
  YAxis,
  Tooltip,
  ResponsiveContainer,
  CartesianGrid,
} from "recharts";
const formatDate = (isoString) => {
  const date = new Date(isoString);
  const dd = String(date.getDate()).padStart(2, "0");
  const mm = String(date.getMonth() + 1).padStart(2, "0");
  const hh = String(date.getHours()).padStart(2, "0");
  const min = String(date.getMinutes()).padStart(2, "0");
  return `${dd}.${mm}, ${hh}:${min}`;
};
const Measurements = ({ weight, setWeight, height, setHeight,
history, onSave }) => {
  const latest = history[0];
  const previous = history[1];
  const diff = latest && previous ? (latest.weight -
previous.weight).toFixed(1) : 0;
  const isGain = diff > 0;

  return (
    <div className="bg-white dark:bg-zinc-900 text-gray-900
dark:text-gray-100 p-6 rounded-2xl shadow-md">
      <h2 className="text-2xl font-bold mb-6 text-center">
Виміри тіла</h2>
      <div className="grid grid-cols-1 md:grid-cols-2 gap-4
mb-6">
        <input
          type="number"
          placeholder="Вага (кг)"
          value={weight}
          onChange={(e) => setWeight(e.target.value)}
          className="border border-gray-300 dark:border-zinc-
700 bg-white dark:bg-zinc-800 text-gray-900 dark:text-gray-
100 p-3 rounded-lg w-full focus:outline-none focus:ring-2
focus:ring-blue-500 transition"
        />
        <input
          type="number"
          placeholder="Зріст (см)"
          value={height}
          onChange={(e) => setHeight(e.target.value)}
          className="border border-gray-300 dark:border-zinc-
700 bg-white dark:bg-zinc-800 text-gray-900 dark:text-gray-
100 p-3 rounded-lg w-full focus:outline-none focus:ring-2
focus:ring-blue-500 transition"
        />
      </div>

      <button
        onClick={onSave}
        className="w-full bg-blue-600 hover:bg-blue-700 text-
white font-semibold py-3 rounded-lg transition"
      >
         Зберегти вимір
      </button>
      {latest && previous && (
        <div className="mt-5 text-sm text-center">
          <strong>Зміна:</strong>{" "}
```

```
        <span className={isGain ? "text-green-600" : "text-red-
600"}>
          {isGain ? "□ +" : "□ -"}{Math.abs(diff)} кг
        </span>
        <span className="text-gray-500 dark:text-gray-400"> з
{formatDate(previous.date)} по
{formatDate(latest.date)}</span>
      </div>
    )}
    {history.length > 0 && (
      <div className="mt-6">
        <h3 className="font-semibold text-sm text-gray-700
dark:text-gray-300 mb-2">□ Історія вимірів</h3>
        <ul className="max-h-40 overflow-y-auto text-sm
space-y-1 text-gray-700 dark:text-gray-200 px-2">
          {history.map((item, idx) => (
            <li key={idx} className="border-b border-gray-200
dark:border-zinc-700 pb-1">
              {formatDate(item.date)} — <strong>{item.weight}
кг</strong>, {item.height} см
            </li>
          ))}
        </ul>
      </div>
    )}
    {history.length > 1 && (
      <div className="mt-8">
        <h3 className="text-sm font-semibold mb-2 text-gray-
700 dark:text-gray-300">□ Графік зміни ваги</h3>
        <div className="w-full h-64">
          <ResponsiveContainer width="100%" height="100%">
            <LineChart
              data={history.slice().reverse()}
              margin={{ top: 5, right: 20, left: 10, bottom: 30 }}
            >
              <CartesianGrid strokeDasharray="3 3"
stroke="#e5e7eb" />
              <XAxis
                dataKey="date"
                tickFormatter={formatDate}
                angle={-30}
                textAnchor="end"
                height={50}
              />
              <Tooltip
                labelFormatter={formatDate}
                contentStyle={{ backgroundColor: "#1f2937",
border: "none" }}
                labelStyle={{ color: "#ffffff" }}
                itemStyle={{ color: "#3b82f6" }}
              />
              <YAxis unit=" кг" />
              <Tooltip labelFormatter={formatDate} />
              <Line
                type="monotone"
                dataKey="weight"
                stroke="#2563eb"
                strokeWidth={3}
                dot={{ r: 4 }}
              />
            </LineChart>
          </ResponsiveContainer>
        </div>
      </div>
    )}
  </div>
 );
};

export default Measurements
```

Код profile/RecipeAnalytics.jsx

```
import React from "react";
import {
 PieChart,
 Pie,
 Cell,
 Tooltip,
 ResponsiveContainer,
 BarChart,
 Bar,
 XAxis,
 YAxis,
} from "recharts";

const COLORS = ["#3b82f6", "#10b981", "#f59e0b",
"#ef4444", "#6366f1"];

const RecipeAnalytics = ({ archive }) => {
 if (!archive || archive.length === 0) {
  return <p className="text-gray-500 text-sm">Недостатньо
даних для аналітики.</p>;
 }

 const kitchenCounts = archive.reduce((acc, rec) => {
  const kitchen = rec.kitchen || "Невідомо";
  acc[kitchen] = (acc[kitchen] || 0) + 1;
  return acc;
 }, {});
 const kitchenData = Object.entries(kitchenCounts).map(([k,
v]) => ({ name: k, value: v }));

 const calorieData = archive.filter((r) => r.total_calories >
0).map((r) => ({
  name: r.title.length > 12 ? r.title.slice(0, 12) + "…" : r.title,
  calories: r.total_calories,
 }));

 const total = archive.length;
 const avgCalories =
  archive.reduce((sum, r) => sum + r.total_calories, 0) /
Math.max(1, total);
 const uniqueKitchens = Object.keys(kitchenCounts).length;

 return (
  <div className="bg-white dark:bg-zinc-900 p-6 rounded-xl
shadow-md space-y-6">
   <h2 className="text-2xl font-bold mb-4">□ Аналітика
рецептів</h2>

   <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
    <div className="bg-gray-50 dark:bg-zinc-800 p-4
rounded-lg shadow">
     <h3 className="text-sm font-medium mb-2">Типи
кухонь</h3>
     <ResponsiveContainer width="100%" height={200}>
      <PieChart>
       <Pie
        data={kitchenData}
        dataKey="value"
        nameKey="name"
        cx="50%"
        cy="50%"
        outerRadius={70}
        label
       >
```

```
            {kitchenData.map((_, index) => (
              <Cell key={index} fill={COLORS[index %
COLORS.length]} />
            ))}
          </Pie>
          <Tooltip />
        </PieChart>
      </ResponsiveContainer>
    </div>

    <div className="bg-gray-50 dark:bg-zinc-800 p-4
rounded-lg shadow">
      <h3 className="text-sm font-medium mb-
2">Калорійність страв</h3>
      <ResponsiveContainer width="100%" height={200}>
        <BarChart data={calorieData}>
          <XAxis dataKey="name" tick={{ fontSize: 12 }} />
          <YAxis unit=" ккал" />
          <Tooltip />
          <Bar dataKey="calories" fill="#3b82f6" radius={[4, 4,
0, 0]} />
        </BarChart>
      </ResponsiveContainer>
    </div>
  </div>

  <div className="grid grid-cols-1 sm:grid-cols-3 gap-4
text-sm text-center">
    <div className="bg-gray-100 dark:bg-zinc-800 rounded-
lg p-4">
      <p className="text-gray-500">Збережено
рецептів</p>
      <p className="text-xl font-semibold">{total}</p>
    </div>
    <div className="bg-gray-100 dark:bg-zinc-800 rounded-
lg p-4">
      <p className="text-gray-500">Середня
калорійність</p>
      <p className="text-xl font-
semibold">{avgCalories.toFixed(0)} ккал</p>
    </div>
    <div className="bg-gray-100 dark:bg-zinc-800 rounded-
lg p-4">
      <p className="text-gray-500">Кухонь у вибірці</p>
      <p className="text-xl font-
semibold">{uniqueKitchens}</p>
    </div>
  </div>
 );
};

export default RecipeAnalytics
```