НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
інформаційних систем та технологій

_____ Швиденко М.З., зав. каф., к.е.н. засл. проф
(підпис) (ПІБ, вчене звання і ступінь)

«___»_____2025 р

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка інформаційної системи підтримки та прийняття рішень при складанні меню здорового харчування»

Спеціальність 121 «Інженерія програмного забезпечення»

Гарант освітньої програми

д.е.н., професор _____ Руденський Р.А.
(Науковий ступень та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

д.філос.н(спец.122), ст.викл. _____ Золотуха Р.А.
(Науковий ступень та вчене звання) (підпис) (ПІБ)

Виконав _____ Пилипчук А.А.
(підпис) (ПІБ)

КИЇВ-2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ

Завідувач кафедри
інформаційних систем та технологій

_____/ Швиденко М.З., зав. каф., к.е.н. засл. проф /

підпис

“ _ ” _____ 2025 р.

ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

студент Пилипчук Анна Андріївна

Спеціальність 121 «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи: Розробка інформаційної системи підтримки та прийняття рішень при складанні меню здорового харчування

Затверджена наказом ректора НУБіП України від 16.12.2024 № 2246 “С”

Термін подання завершеної роботи на кафедру _____

(рік, місяць, число)

Вихідні дані до роботи: опис програмного забезпечення

Перелік питань , які потрібно розробити:

Аналіз проблемної області, вибір та обґрунтування засобів для розробки системи, проектування інформаційної системи.

Дата видачі завдання “16” 12 2024р.

Керівник бакалаврської кваліфікаційної роботи

д.філос.н(спец.122), ст.викл. _____ Золотуха Р.А.
(Науковий ступень та вчене звання) (підпис) (ПІБ)

Виконав _____ Пилипчук А.А.
(підпис) (ПІБ)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

– ІСППР — інформаційна система підтримки прийняття рішень

– БД — база даних

– API — прикладний програмний інтерфейс (Application Programming Interface)

– GUI — графічний інтерфейс користувача (Graphical User Interface)

– HTTP — протокол передавання гіпертексту (HyperText Transfer Protocol)

– JSON — формат обміну даними JavaScript Object Notation

– JS — мова програмування JavaScript

– Node.js — програмна платформа для виконання JavaScript-коду на стороні сервера

– Express.js — фреймворк для створення серверних веб-додатків на основі Node.js

– REST — архітектурний стиль побудови API (Representational State Transfer)

– MVC — архітектурна модель «Model–View–Controller»

– SQL — мова структурованих запитів (Structured Query Language)

– UI — інтерфейс користувача (User Interface)

– UX — користувацький досвід (User Experience)

– HTML — мова розмітки гіпертексту (HyperText Markup Language)

– CSS — каскадні таблиці стилів (Cascading Style Sheets)

– NPM — менеджер пакетів для середовища Node.js (Node Package Manager)

– CRUD — базові операції з даними: створення (Create), читання (Read), оновлення (Update), видалення (Delete)

– SPA — односторінковий вебзастосунок (Single Page Application)

– JWT — стандарт маркерів доступу (JSON Web Token)

ВСТУП

У сучасних умовах цифровізації охорони здоров’я та підвищеної уваги до превентивної медицини особливої значущості набувають інформаційні технології, спрямовані на підтримку здорового способу життя. Зокрема, формування збалансованого харчового раціону є однією з ключових складових профілактики хронічних захворювань і підтримання життєдіяльності організму. Традиційні підходи до складання раціону передбачають участь дієтолога або самостійне опрацювання значних обсягів інформації користувачем, що є трудомістким процесом, не завжди ефективним і доступним широкому загалу. Це зумовлює необхідність створення автоматизованих рішень, які спрощують прийняття рішень у галузі здорового харчування на основі об’єктивних критеріїв та медичних рекомендацій.

Актуальність теми бакалаврської кваліфікаційної роботи полягає в зростаючій потребі населення у персоналізованих цифрових інструментах, які забезпечують науково обґрунтоване формування раціону з урахуванням індивідуальних фізіологічних потреб, харчових обмежень, рівня активності та цілей користувача (нормалізація ваги, профілактика захворювань тощо). У зв’язку з цим постає потреба у впровадженні інформаційних систем, які забезпечують не лише рекомендаційні функції, а й можливість адаптації до динамічних змін параметрів користувача. З огляду на сучасні тенденції в галузі ІТ та здоров’я, особливої ваги набуває поєднання технологій розробки вебзастосунків з принципами доказової дієтології, що відкриває нові перспективи для розвитку галузі на перетині медицини, програмної інженерії та когнітивних технологій.

Метою бакалаврської кваліфікаційної роботи є розробка веборієнтованої інформаційної системи підтримки та прийняття рішень при складанні меню здорового харчування, яка надає користувачам можливість формувати

персоналізований раціон відповідно до добових норм споживання макро- і мікронутрієнтів, енергетичної цінності та дієтичних вимог.

Об’єктом дослідження є процес прийняття рішень при формуванні індивідуального харчового раціону.

Предметом дослідження виступають методи проєктування та реалізації інформаційних систем підтримки прийняття рішень, що базуються на вебтехнологіях, базах даних харчових продуктів і алгоритмах оптимізації харчових планів.

Для досягнення поставленої мети в роботі вирішуються такі **завдання:**

- виконати аналіз предметної області та оцінити вимоги до функціональності фітнес-застосунку;
- розробити UML-модель, що описує структуру та взаємодію програмних компонентів;
- побудувати архітектуру програмного забезпечення із розподілом на логічні підсистеми;
- реалізувати адаптивний інтерфейс користувача для мобільних платформ;
- розробити алгоритми формування тренувальних сесій на основі вхідних параметрів;
- забезпечити зберігання даних про параметри користувача та історію занять;

Наукова новизна роботи полягає у реалізації інструменту автоматизованого складання раціону, який поєднує бази харчових даних з алгоритмічними методами прийняття рішень, що робить можливим персоналізований підхід до харчування без участі медичного спеціаліста.

Практичне значення проєкту полягає у створенні готового до використання застосунку, який може бути адаптований для потреб медичних установ, фітнес-центрів, освітніх платформ з превентивної медицини або особистого використання громадянами для контролю харчування.

Структура бакалаврської роботи включає чотири розділи: перший розділ містить огляд теоретичних і нормативних засад побудови інформаційних систем у галузі харчування; другий розділ присвячений проєктуванню системи, побудові її моделі та вибору засобів реалізації; третій розділ висвітлює практичну реалізацію, приклади коду, структуру бази даних та результати тестування. У завершенні подано висновки щодо досягнутих результатів і перспектив подальшого вдосконалення.

.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Раціональне харчування є однією з ключових складових підтримання фізичного та психічного здоров’я людини. У контексті зростаючої захворюваності на хронічні недуги, пов’язані з порушенням метаболізму, надмірною масою тіла та дефіцитом поживних речовин, питання формування здорового щоденного меню набуває пріоритетного значення як у персональній практиці, так і в роботі медичних, спортивних та освітніх установ. Проте самостійне складання меню потребує спеціальних знань у сфері дієтології, обчислення калорійності, балансу макро- і мікроелементів, урахування харчових обмежень, алергенів та супутніх захворювань.

Предметна область даного дослідження охоплює розробку інтерактивної інформаційної системи, яка дозволяє автоматизувати процес створення збалансованого меню відповідно до індивідуальних параметрів користувача (вік, стать, фізична активність, цілі харчування) та добових норм нутрієнтів. Система виконує функцію інструменту підтримки прийняття рішень, надаючи користувачу науково обґрунтовані рекомендації щодо раціону з урахуванням встановлених медичних стандартів і баз даних продуктів харчування.

Особливістю предметної області є висока варіативність вхідних параметрів: кількість прийомів їжі, доступні продукти, рівень енерговитрат, стан здоров’я, що вимагає реалізації гнучкої системи конфігурації профілю користувача та підтримки фільтрації продуктів за категоріями, складом, алергенами та калорійністю. Також необхідно врахувати взаємозалежності між продуктами (наприклад, несумісність або взаємне посилення засвоюваності) та рекомендації щодо режиму харчування.

На відміну від типових мобільних трекерів калорій, що лише фіксують спожиті продукти, розроблювана система виконує активну функцію генерації

меню, тобто формує пропозиції щодо прийомів їжі, що відповідають заданим критеріям, забезпечуючи користувачу вибір між кількома варіантами раціону на день або тиждень.

Для реалізації програмного продукту використовується стек вебтехнологій:

– JavaScript — основна мова для реалізації логіки як на клієнтській, так і на серверній стороні;

– Node.js — середовище виконання JavaScript-коду на сервері;

– Express.js — вебфреймворк для створення RESTful API та обробки запитів;

– JSON — формат обміну структурованими даними між модулями та клієнтською частиною;

– HTML/CSS — для побудови інтерфейсу користувача, що є доступним, адаптивним та простим у використанні.

Ключовими компонентами предметної області є:

– база даних харчових продуктів, що містить інформацію про склад, енергетичну цінність, категорії, протипоказання;

– модуль профілю користувача, де зберігаються фізіологічні параметри, цілі та обмеження;

– алгоритм генерації раціону, який на основі заданих умов формує збалансоване меню з урахуванням норм добового споживання;

– інтерфейс прийняття рішень, що дозволяє користувачу обрати меню, відредагувати його або зберегти обраний варіант.

Предметна область передбачає створення системи, що виконує не лише обчислювальні функції, але й має інформаційно-аналітичну природу, забезпечуючи підтримку прийняття індивідуалізованих рішень у сфері харчування.

1.2 Огляд інформаційних джерел та існуючих рішень

На сучасному етапі розвитку інформаційних технологій ринок вебзастосунків, орієнтованих на надання кулінарних рецептів та підтримку планування раціону, активно розширюється. Проте більшість таких платформ зосереджуються на загальній аудиторії та не забезпечують належного рівня персоналізації для користувачів із медичними показаннями чи специфічними дієтичними потребами. В рамках даного дослідження було проаналізовано функціональність чотирьох провідних сервісів: AllRecipes, KitchenAid (Yummly), BBC Good Food та Eat This Much, з метою порівняння їх можливостей із планованою інформаційною системою.

Сервіс AllRecipes є одним із наймасовіших кулінарних порталів з великою базою рецептів, які супроводжуються відео- та фотоінструкціями, системою оцінювання та коментарями користувачів (рис. 1.1). Позитивною стороною є наявність фільтрів за типом дієт (вегетаріанська, безглютенова тощо), однак ці фільтри реалізовані поверхнево — вони не забезпечують багатофакторної перевірки інгредієнтів на сумісність, а інформація про алергени подається неструктуровано або відсутня взагалі [15].

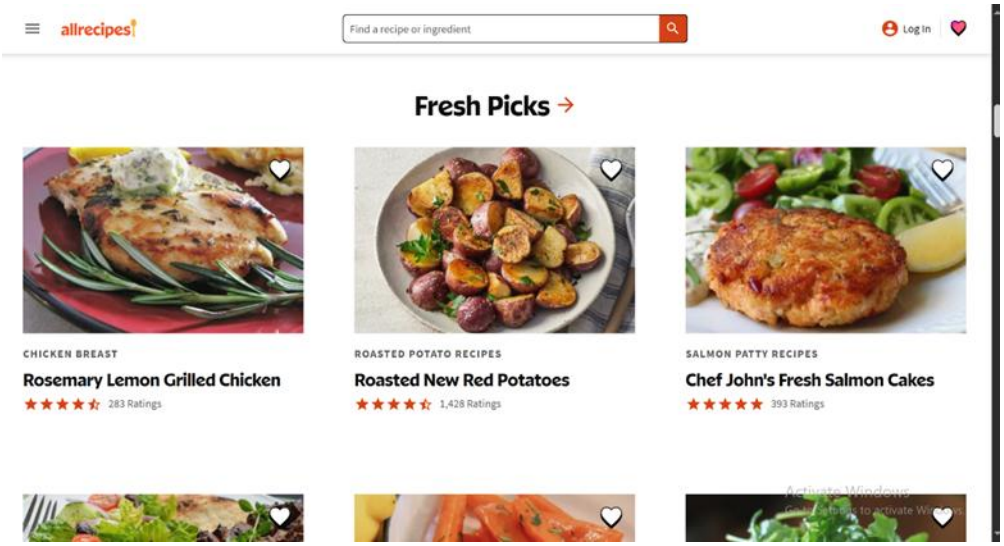


Рисунок 1.1 – Інтерфейс сервісу AllRecipes

Система KitchenAid (Yummly) демонструє вищий рівень персоналізації, дозволяючи створювати профілі з указанням алергенів, типу дієти та небажаних продуктів (рис. 1.2). Платформа також підтримує генерацію списків покупок і синхронізацію з розумною кухонною технікою. Недоліками є складність інтерфейсу, обмеження безкоштовного функціоналу та відсутність україномовної локалізації, що знижує її доступність для вітчизняних користувачів [16].

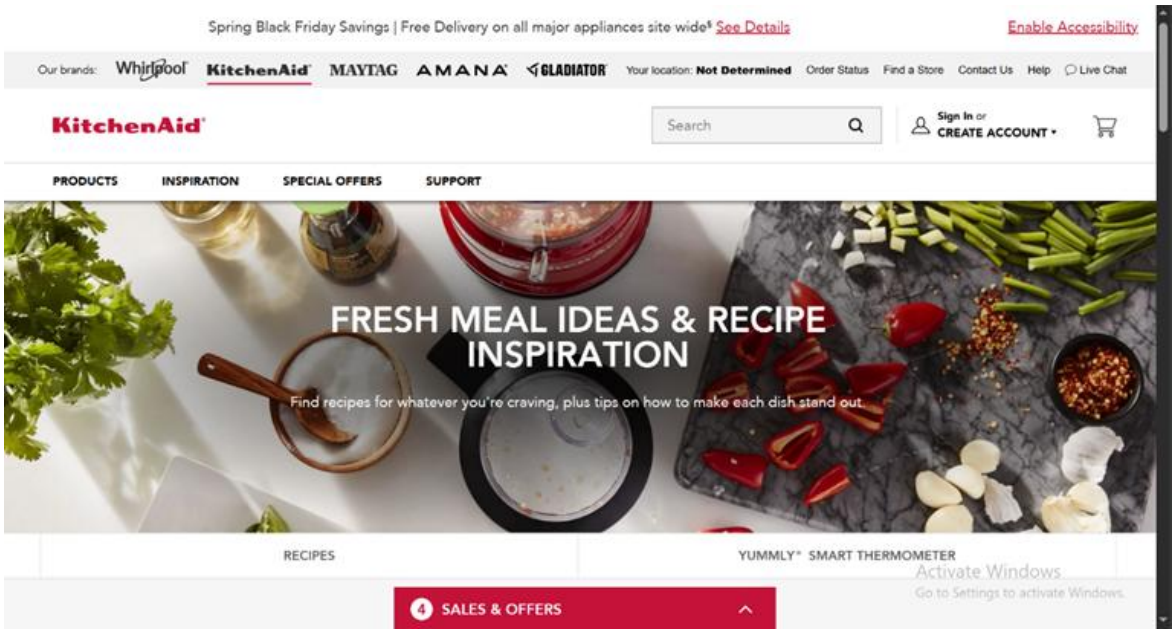


Рисунок 1.2 – Головна сторінка KitchenAid (Yummly)

Сервіс BBC Good Food пропонує високу якість контенту та надійність рецептів, які проходять редакторську перевірку. Категорії здорового харчування присутні (рис. 1.3), але можливості автоматизованого врахування алергенів, як і фільтрація за складними комбінаціями дієт, суттєво обмежені. Такий підхід є прийнятним для здорових користувачів, однак недостатнім у контексті медичних дієт [17].

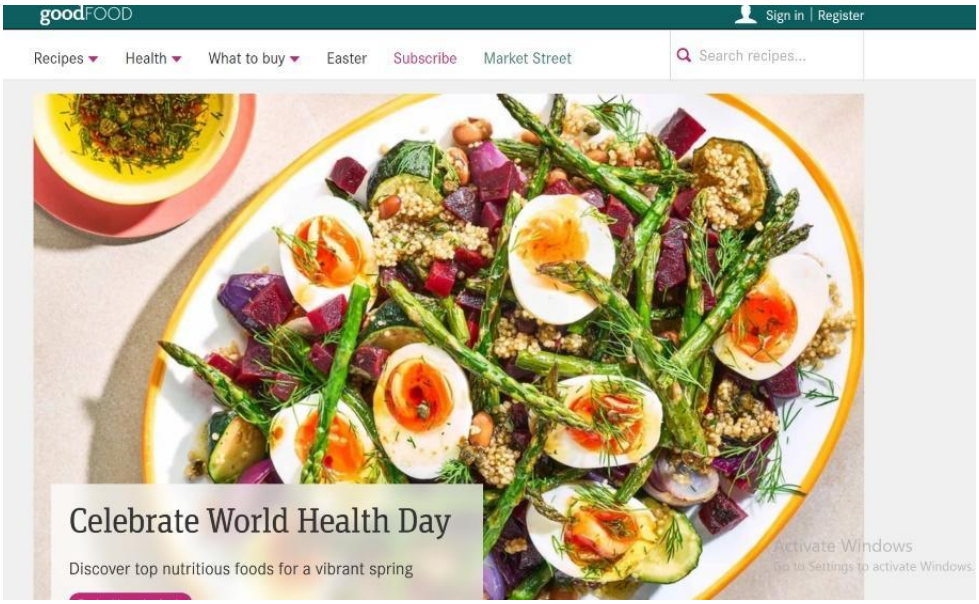


Рисунок 1.3– Категорія здорового харчування на BBC Good Food

Платформа Eat This Much вирізняється серед аналогів акцентом на автоматичне планування харчування, з урахуванням цілей користувача (наприклад, схуднення або набір маси), енергетичних витрат і бюджету (рис. 1.4). Незважаючи на наявність налаштувань дієт (кето, веган, палео), система не передбачає медично орієнтованої персоналізації — наприклад, вона не розпізнає комбінацію діабету та алергії на глютен [18].

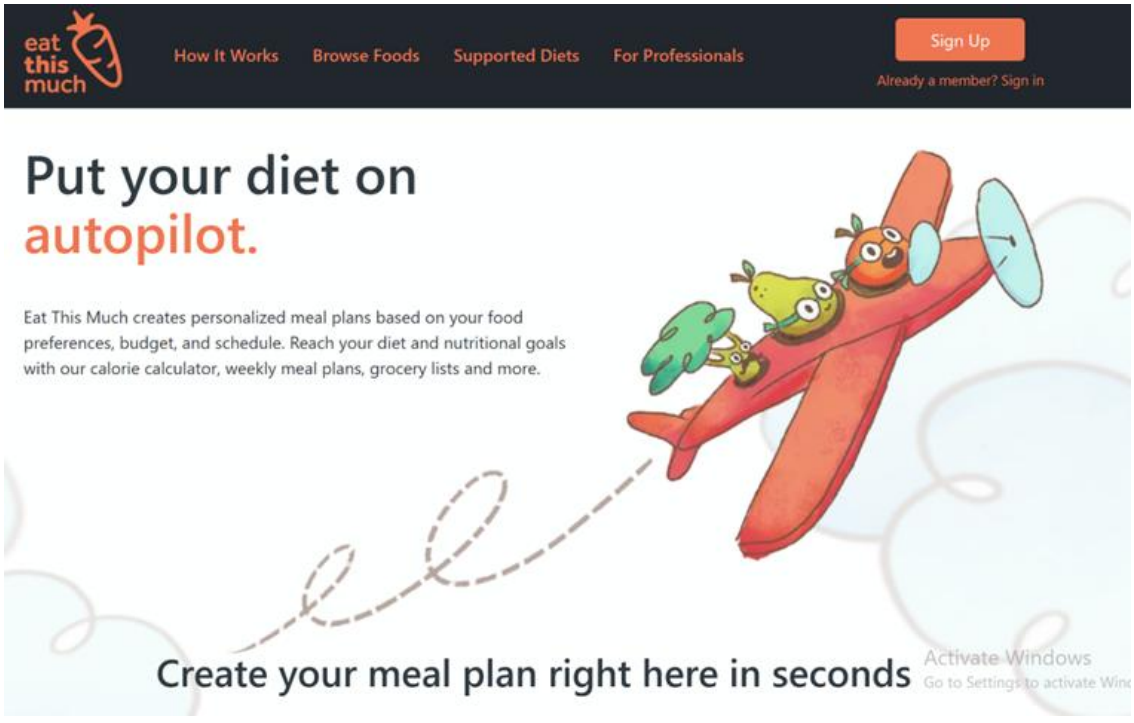


Рисунок 1.4 – Персоналізація дієти на Eat This Much

Зведена порівняльна характеристика цих сервісів наведена у таблиці 1.1:

Таблиця 1.1

Порівняння популярних вебсервісів з розроблюваною системою

Сервіс	Переваги	Недоліки	Порівняння з розроблюваною системою
AllRecipes [15]	Велика база рецептів, відеоінструкції, рейтинги	Неструктурована фільтрація, ручна перевірка алергенів	У нашій системі передбачено автоматичну перевірку несумісних інгредієнтів та розширену систему фільтрації
KitchenAid / Yummly [16]	Персоналізація профілю, алергенні фільтри, список покупок	Обмеження у функціоналі, складний інтерфейс, без локалізації	Запропонована система буде україномовною, доступною та адаптованою до користувачів з медичними обмеженнями
BBC Good Food [17]	Висока якість рецептів, категоризація здорового харчування	Відсутність алергенної валідації, обмежена гнучкість	У розроблюваній системі буде реалізовано автоматичну діагностику алергенів і дієтичних конфліктів
Eat This Much [18]	Автоматичне планування, дієтичні цілі, бюджетування	Не враховує медичні обмеження, спрощений підбір	Наша система охоплює медико-нутріціологічні сценарії, включаючи складні комбінації (діабет + алергія + дієта)

Аналіз функціональних можливостей сучасних вебплатформ свідчить, що хоча персоналізація харчування активно розвивається, наявні сервіси не забезпечують повноцінну підтримку користувачів із множинними чи медичними обмеженнями. Бракує глибокої перевірки складу продуктів, підтримки комбінованих профілів, а також локалізованих інтерфейсів.

У зв’язку з цим обґрунтовується необхідність створення спеціалізованої інформаційної системи, яка реалізує:

– підтримку багатофакторної персоналізації;
– верифікацію алергенів на рівні інгредієнтів;
– автоматичне формування раціону з медичним підходом;
– україномовний та інклюзивний інтерфейс.

Саме такий підхід і реалізується в рамках розробки даного вебзастосунку, орієнтованого на підтримку користувачів із особливими харчовими потребами.

1.3 Аналіз вимог до програмної системи

Аналіз вимог до інформаційної системи підтримки та прийняття рішень при складанні меню здорового харчування є визначальним етапом у процесі проєктування, оскільки він забезпечує чітке формулювання очікувань користувачів, технічних параметрів функціонування, а також вимог до зручності, адаптивності та безпеки системи. Для забезпечення структурованого підходу до опису вимог застосовано класифікацію відповідно до міжнародних стандартів IEEE 830–1998 та ISO/IEC/IEEE 29148:2018, які передбачають поділ на функціональні, нефункціональні, системні вимоги та обмеження [12].

У межах проведеного дослідження визначено базові функціональні вимоги, які охоплюють основні сценарії взаємодії користувача із системою. Вони охарактеризовані в таблиці 1.2.

Таблиця 1.2

Функціональні вимоги до системи складання меню здорового харчування

№	Назва функції	Опис функціональності
1	Створення профілю користувача	Введення особистих параметрів (вік, стать, вага, активність, алергії, харчові цілі)
2	Формування меню	Генерація збалансованого меню згідно індивідуальних вимог користувача
3	Збереження та редагування раціону	Можливість зберігати, копіювати та модифікувати добові плани харчування

Продовження таблиці 1.2

4	Фільтрація продуктів	Вибір продуктів за категоріями, алергенами, поживною цінністю
5	Розрахунок нутрієнтів	Автоматичний підрахунок калорій, білків, жирів, вуглеводів, вітамінів тощо
6	Генерація списку покупок	Створення списку необхідних інгредієнтів відповідно до обраного меню

Визначення функціональних вимог ґрунтувалося на результатах аналізу аналогічних сервісів (див. підпункт 1.2), а також ураховувало специфіку потреб цільової аудиторії — осіб із особливими дієтичними обмеженнями, медичними показаннями або багатофакторними харчовими запитами [3, с. 74].

Окрім функціональної поведінки, важливим аспектом є нефункціональні вимоги, що визначають ключові характеристики системи в контексті продуктивності, зручності, безпеки та адаптивності. Їх деталізовано в таблиці 1.3.

Таблиця 1.3

Нефункціональні вимоги до інформаційної системи

№	Категорія	Вимога
1	Надійність	Збереження даних користувача у випадку збоїв у роботі клієнтського середовища
2	Продуктивність	Формування меню та розрахунок нутрієнтів повинні виконуватись за час не більше 2 секунд
3	Безпека	Обмеження доступу до профілів через автентифікацію та авторизацію
4	Зручність використання	Адаптивний інтерфейс із логічною структурою, можливість використання на ПК і мобільних пристроях
5	Локалізація	Підтримка української мови для всіх елементів інтерфейсу

Установлення нефункціональних вимог спрямоване на досягнення високого рівня користувацького досвіду при щоденному використанні системи, що є критичним для її впровадження в побут або клінічну практику [10].

Також були сформульовані системні вимоги, які відображають технічні умови розгортання та сумісності програмного продукту. Їх подано в таблиці 1.4.

Таблиця 1.4

Системні вимоги до середовища виконання

№	Компонент	Вимога
1	Операційна система	Windows 10+, Linux Ubuntu 20.04+, macOS 11+
2	Браузер	Підтримка Chrome 100+, Firefox 100+, Safari 14+, Edge Chromium 100+
3	Серверна платформа	Node.js версії 18 або новішої
4	Фреймворк	Express.js (для обробки запитів і маршрутизації)
5	Формат обміну	JSON (для клієнт-серверної взаємодії)
6	База даних	Реляційна або JSON-орієнтована локальна БД (наприклад, SQLite або lowdb)
7	Клієнтська частина	HTML5, CSS3, JavaScript з адаптивною верстkою

Окрему увагу приділено обмеженням, які накладаються як технологічними, так і етичними факторами. Серед них — відсутність залежності від зовнішніх API у базовій версії, робота в офлайн-режимі (для забезпечення конфіденційності), вимога щодо збереження даних лише на локальному пристрої користувача, а також обов’язкова підтримка української мови в усіх елементах інтерфейсу.

Комплексне урахування зазначених вимог дозволяє забезпечити відповідність розроблюваної системи потребам користувача, вимогам нормативно-правових актів та принципам програмної інженерії сучасного рівня.

1.4 Постановка завдання

На основі виконаного аналізу предметної області, формалізації типових сценаріїв користувацької взаємодії та визначення функціональних і нефункціональних вимог сформульовано технічне завдання на розробку інформаційної системи підтримки та прийняття рішень при складанні меню здорового харчування. Розроблювана система має забезпечити автоматизоване формування добового або тижневого раціону з урахуванням індивідуальних

характеристик користувача, дієтичних обмежень, алергенів, нутрієнтного балансу та фінансових параметрів.

Передбачається реалізація вебзастосунку на основі технологій JavaScript, Node.js, Express та клієнтської частини з використанням HTML, CSS та стандартних вебінтерфейсів. Уся обробка даних та генерація меню мають виконуватись на локальному рівні або у рамках обмеженої серверної логіки без залучення зовнішніх API. Це забезпечує конфіденційність персональної інформації та підвищує стабільність функціонування системи в автономному режимі.

До вхідних параметрів, які необхідно враховувати при побудові раціону, належать персональні фізіологічні показники користувача, його цілі, переваги та обмеження, а також доступні продукти. Структура основних вхідних даних подана в таблиці 1.5.

Таблиця 1.5

Вхідні параметри для формування раціону

Категорія	Параметри
Користувач	Вік, стать, вага, ріст, рівень фізичної активності
Ціль	Підтримка ваги, схуднення, набір маси, лікувальна дієта
Обмеження	Алергії (глютен, лактоза, горіхи тощо), заборонені продукти
Дієтичні уподобання	Вегетаріанство, веганство, кето, безцукрова дієта тощо
Продуктова база	Перелік інгредієнтів із зазначенням калорійності, макронутрієнтів, алергенів
Бюджет	Максимальна вартість добового раціону, обмеження на дорогі продукти

Результатом функціонування системи має бути створення персоналізованого плану харчування, що відповідає заданим критеріям і забезпечує збалансоване надходження поживних речовин. Основні результати, які система повинна формувати, наведено в таблиці 1.6.

Таблиця 1.6

Очікувані результати роботи системи

Компонент результату	Опис

Персоналізоване меню	Список страв із зазначенням інгредієнтів, обсягу порцій та часу прийому
Нутрієнтний звіт	Підсумкові значення калорійності, білків, жирів, вуглеводів, вітамінів
Алергенний контроль	Повідомлення про наявність заборонених інгредієнтів у рецептах
Список покупок	Згенерований перелік продуктів із розрахунком необхідної кількості
Експорт результатів	Можливість збереження меню та звітів у форматі PDF або JSON
Повідомлення користувачу	Інформування про успішність генерації меню, помилки, нестачу даних

Архітектура інформаційної системи має бути модульною та адаптивною. Основні програмні компоненти включають: модуль обробки профілю користувача, генератор меню на основі фільтрації та алгоритмів добору, модуль обрахунку нутрієнтів, система валідації алергенів, модуль збереження результатів та інтерфейс виведення інформації.

З технічного погляду проєкт рекомендується реалізовувати з дотриманням архітектурного шаблону MVC (Model–View–Controller), що дозволить розмежувати бізнес-логіку, візуальний інтерфейс та структуру даних, а також забезпечить можливість автономного тестування та масштабування системи в майбутньому.

Загальна постановка задачі визначає вимоги до кожного з функціональних блоків системи, формує межі технічної реалізації та забезпечує цілісну концепцію, яка буде покладена в основу наступного етапу — створення концептуальної моделі інформаційної системи.

1.5 Моделювання предметної області

Моделювання предметної області є ключовим етапом у процесі розробки інформаційної системи, оскільки дозволяє формалізувати основні функціональні взаємодії між користувачами та програмною системою, а також визначити межі відповідальності окремих компонентів. Застосування засобів об’єктно-орієнтованого аналізу забезпечує структуроване подання сценаріїв роботи системи, що підвищує прозорість архітектурних рішень та спрощує процес проєктування програмної логіки [14].

Для моделювання предметної області використано нотацію UML (Unified Modeling Language), яка є міжнародним стандартом для візуалізації, опису та документування програмних систем. Основна увага приділяється побудові діаграми прецедентів та діаграми послідовності, які є найдоцільнішими на ранньому етапі розробки.

На діаграмі прецедентів (рис. 1.5) відображено основні сценарії взаємодії користувачів з інформаційною системою складання меню. Користувач має можливість проходити процедуру реєстрації, редагувати профіль, генерувати персоналізоване меню, фільтрувати продукти, розраховувати нутрієнтний склад та експортувати результати. Окремі дії, такі як фільтрація та обрахунок нутрієнтів, є невід’ємними складовими сценарію генерації меню, що позначено відношенням включення (include). Адміністратор має доступ до функцій керування базою даних продуктів та перегляду статистики використання.

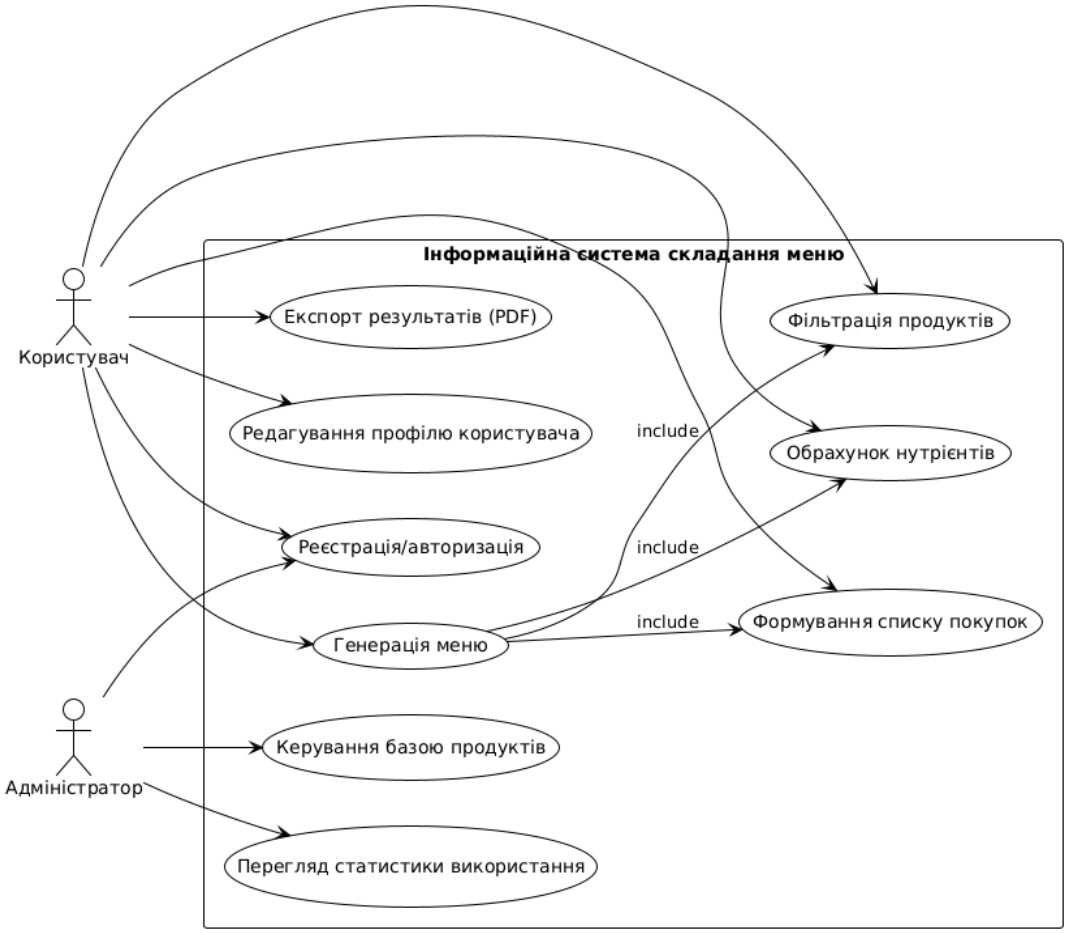


Рисунок 1.5 – UML-діаграма прецедентів для інформаційної системи складання меню

Для деталізації одного з ключових прецедентів – «Генерація меню» – побудовано діаграму послідовності, яка демонструє часову взаємодію між компонентами системи (рис. 1.6). Користувач ініціює запит на формування меню через інтерфейс. Менеджер меню звертається до модуля фільтрації продуктів, який у свою чергу здійснює запит до бази даних. Після отримання списку дозволених продуктів система виконує обрахунок нутрієнтів, використовуючи відповідні дані з бази. Отримані результати повертаються до інтерфейсу користувача для відображення у вигляді меню з підсумковими значеннями калорій, білків, жирів і вуглеводів.

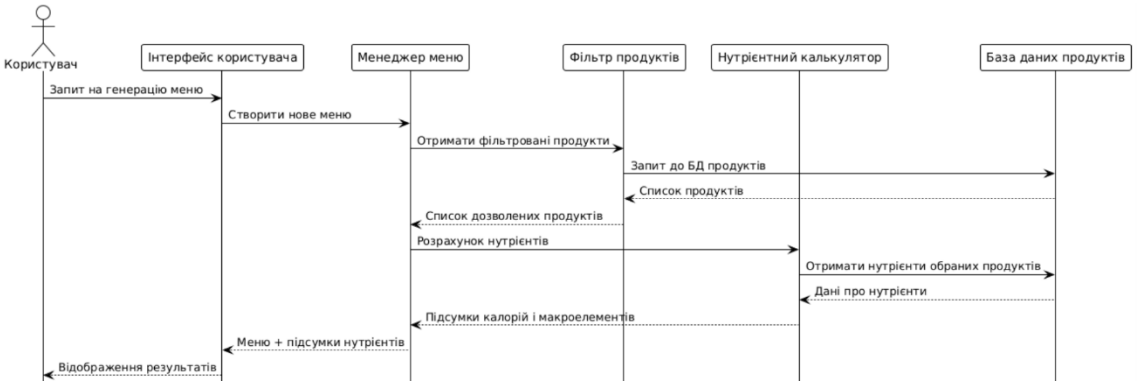


Рисунок 1.6 – Діаграма послідовності процесу генерації персоналізованого меню

Для подальшого уточнення динаміки взаємодії користувача із системою, побудовано діаграму активності, яка відображає логіку послідовних дій у процесі роботи з вебзастосунком (рис. 1.7). На діаграмі представлено повний сценарій типового використання: від ініціалізації системи до здійснення базових функцій, таких як перегляд рецепту, коментування та експорт списку інгредієнтів.

Процес розпочинається з запуску вебзастосунку, після чого користувач обирає дію — вхід у систему або реєстрацію. Якщо користувач вже має обліковий запис, система переходить до етапу авторизації. У протилежному випадку відбувається створення нового облікового запису з подальшим формуванням харчового профілю, що включає зазначення дієтичних параметрів, алергенів, цілей і вподобань.

Після автентифікації або реєстрації відбувається перехід до каталогу рецептів, де запускається процедура автоматичної фільтрації на основі профілю користувача. Це дозволяє миттєво адаптувати інтерфейс до потреб конкретної особи та виводити лише релевантні страви. Кінцеві дії включають перегляд рецепту, оцінювання чи залишення коментаря, а також експорт списку інгредієнтів у форматі, придатному для закупівель чи подальшого аналізу.

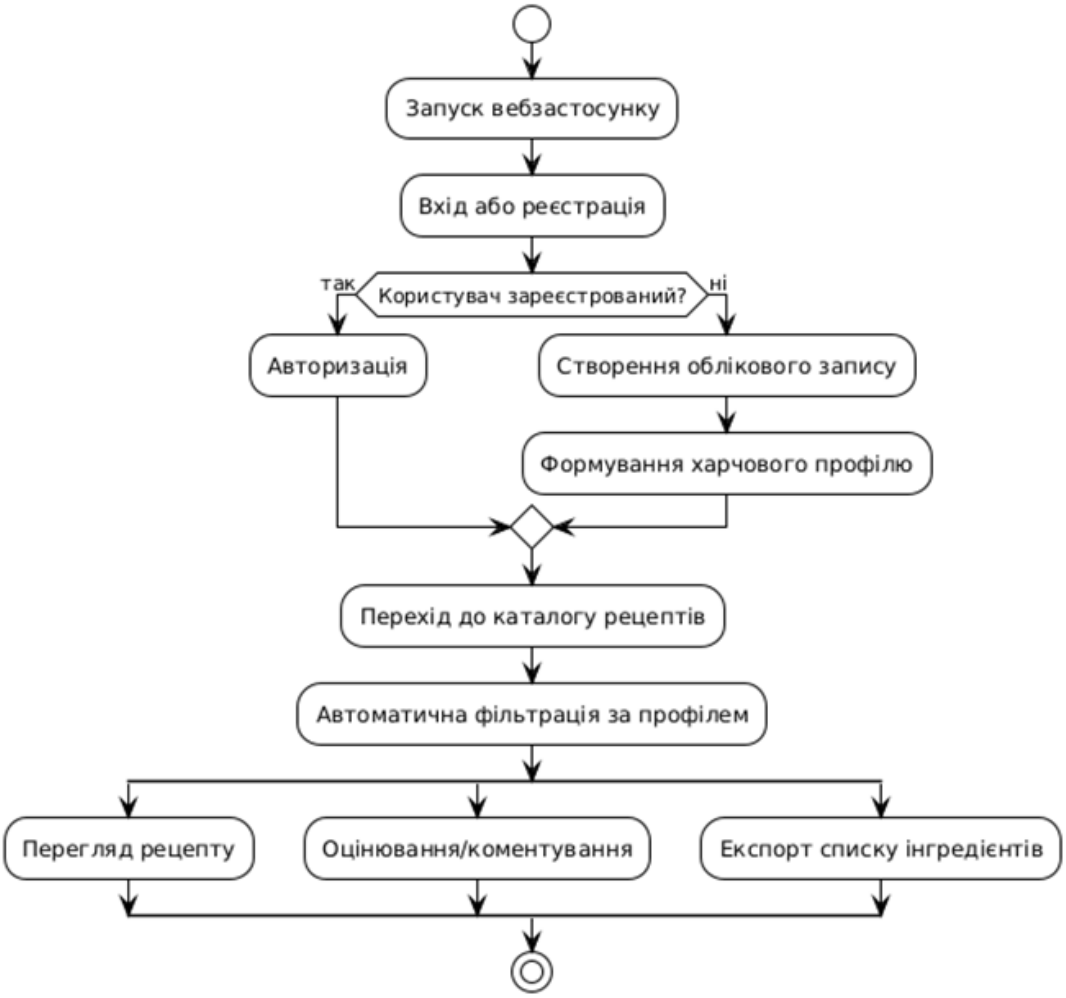


Рисунок 1.7 – UML-діаграма активності користувача у вебзастосунку для складання меню

Представлена діаграма забезпечує візуальне уявлення про логіку роботи інформаційної системи та дозволяє проаналізувати послідовність виконання ключових функцій. Такий підхід є критично важливим для виявлення потенційних розривів у логіці взаємодії, визначення точок перевірки введених даних та закладання основ для побудови архітектурного каркасу застосунку

Поєднання діаграм прецедентів, послідовності та активності дозволяє отримати цілісне бачення функціональної моделі предметної області, що формує підґрунтя для проєктування структури компонентів, бази даних і алгоритмів логіки застосунку.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Проєктування логічної моделі бази даних є одним із ключових етапів реалізації інформаційної системи, оскільки саме на цьому рівні відбувається відображення структурованих елементів предметної області у вигляді сутностей, їхніх атрибутів та взаємозв’язків між ними. Побудована модель відображає всі функціональні компоненти системи: користувачів, профілі, рецепти, алергени, інгредієнти, коментарі та зв’язки між ними. Логічна модель забезпечує концептуальну основу для побудови фізичної структури бази даних та формує уніфікований підхід до обробки інформації в системі.

Для формалізації структури даних використано ER-модель у класичній реляційній формі з відображенням первинних і зовнішніх ключів. На рисунку 2.1 представлено логічну модель, яка охоплює ключові сутності: User, Profile, Recipe, Ingredient, Allergen, Comment та зв’язкові таблиці Profile_Allergen і Recipe_Ingredient, що реалізують зв’язки N:M.

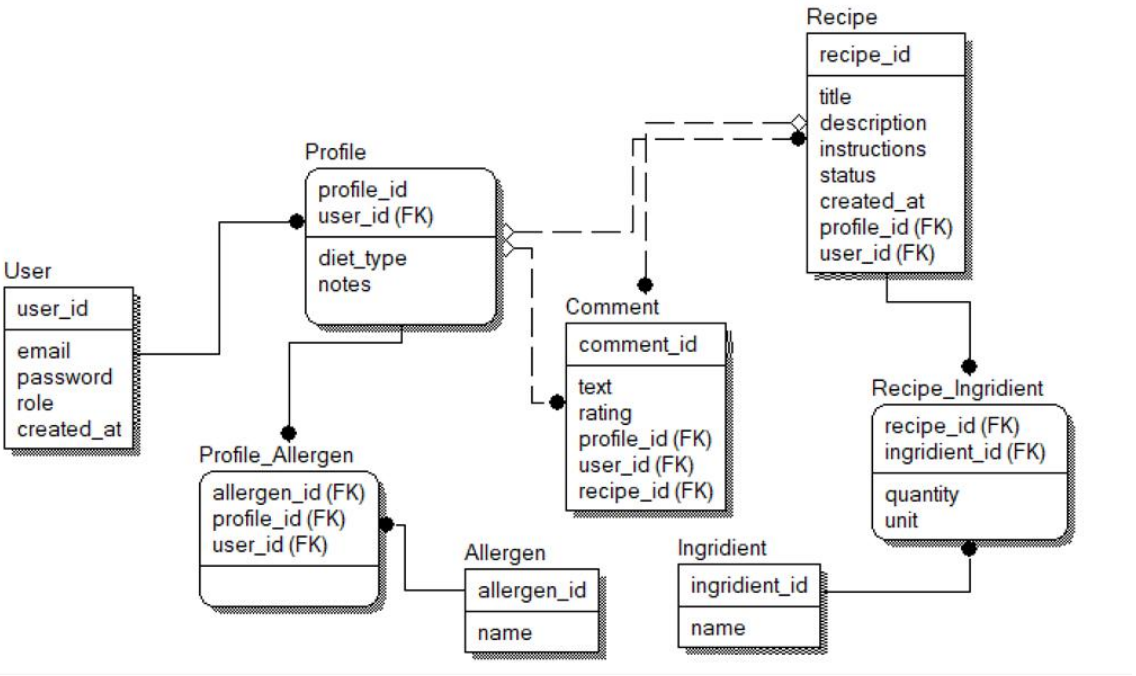


Рисунок 2.1 – ER-діаграма інформаційної системи підтримки складання меню

Сутність User є центральною точкою автентифікації й авторизації користувачів. Вона містить такі атрибути: user_id (PK), email, password, role, created_at. Один користувач може мати лише один профіль, який зберігається в таблиці Profile. У цій таблиці передбачено атрибути profile_id (PK), user_id (FK), diet_type, notes. Таким чином, профіль відображає індивідуальні дієтичні параметри користувача.

Індивідуальна чутливість до алергенів реалізована через зв’язкову таблицю Profile_Allergen, яка дозволяє прив’язувати кілька алергенів до одного профілю. Ця таблиця містить поля allergen_id (FK), profile_id (FK), user_id (FK). Алергени, у свою чергу, зберігаються у довіднику Allergen, що включає поля allergen_id та name.

Рецепти зберігаються в таблиці Recipe, яка включає такі поля, як recipe_id (PK), title, description, instructions, status, created_at, а також зовнішні ключі profile_id та user_id, що забезпечують зв’язок із автором рецепту та його дієтичним контекстом. Структуру рецепту доповнює таблиця Recipe_Ingredient,

яка реалізує зв’язок багато-до-багатьох між рецептами та інгредієнтами. Вона містить поля recipe_id, ingredient_id, quantity, unit.

Інгредієнти зберігаються у довідковій таблиці Ingredient з полями ingredient_id та name. Завдяки цьому реалізується можливість багаторазового використання базових елементів у різних рецептах із точним дозуванням.

Система також дозволяє користувачам залишати коментарі, які зберігаються в таблиці Comment. У структурі цієї сутності передбачено: comment_id, text, rating, profile_id, user_id, recipe_id. Це забезпечує повноцінну підтримку зворотного зв’язку, а також розширює функціональність оцінювання рецептів.

У таблиці 2.1 подано узагальнену характеристику ключових сутностей логічної моделі, з урахуванням первинних та зовнішніх ключів, що використовуються при побудові бази даних.

Таблиця 2.1

Основні сутності логічної моделі даних

Сутність	Первинний ключ	Ключові атрибути	Зовнішні ключі
User	user_id	email, password, role, created_at	–
Profile	profile_id	diet_type, notes	user_id
Allergen	allergen_id	name	–
Profile_Allergen	–	–	allergen_id, profile_id, user_id
Recipe	recipe_id	title, description, instructions, status	user_id, profile_id
Ingredient	ingredient_id	name	–
Recipe_Ingredient	–	quantity, unit	recipe_id, ingredient_id
Comment	comment_id	text, rating	recipe_id, user_id, profile_id

Представлена логічна структура відповідає вимогам третьої нормальної форми (3НФ), що забезпечує усунення надлишкових залежностей між атрибутами та підвищує цілісність даних. Всі зв’язки реалізовано за допомогою

зовнішніх ключів, а багатозначні відношення зведено до зв’язкових таблиць. Такий підхід дозволяє масштабувати систему, доповнювати її новими модулями (наприклад, плануванням меню або модулем рекомендацій), не порушуючи існуючої структури бази даних.

2.2 Діаграма класів і кооперації

Діаграма класів є базовим статичним представленням об’єктно-орієнтованої моделі системи, що описує її основні структурні компоненти (класи), атрибути, методи, а також зв’язки між класами. Вона слугує інструментом для логічного поділу відповідальностей у межах програмного забезпечення, спрощуючи реалізацію, супровід і подальше масштабування інформаційної системи .

На рисунку 2.2 представлено UML-діаграму класів, що відображає ключові об’єкти предметної області: User, Profile, Recipe та Comment. Кожен з класів репрезентує відповідну функціональну частину системи та реалізує набір методів, що забезпечують цілісність даних і підтримку логіки прийняття рішень при складанні меню.

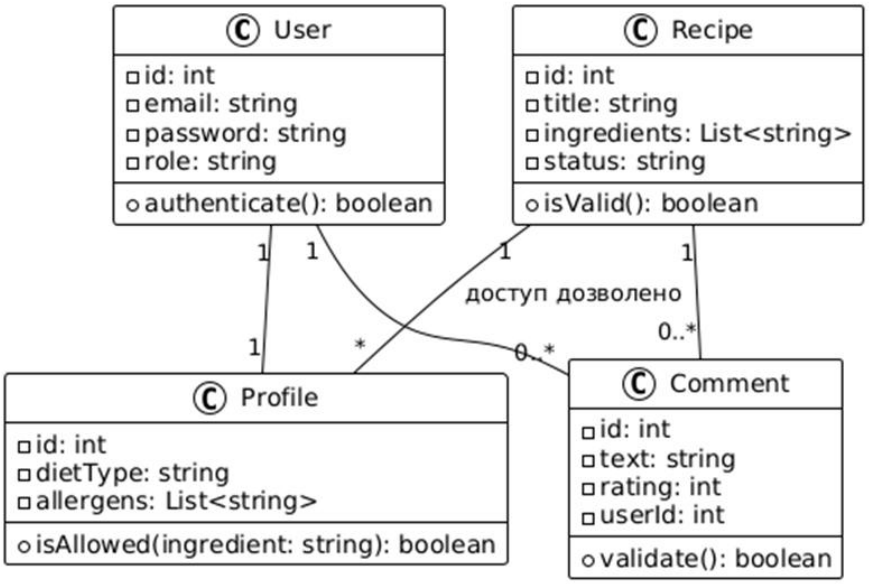


Рисунок 2.2 – UML-діаграма класів інформаційної системи складання меню

Клас User моделює облікові дані користувача і містить атрибути id, email, password, role. Метод authenticate() відповідає за перевірку достовірності облікового запису. Зв’язок між User та Profile реалізовано як відношення один до одного, що означає, що кожен користувач має лише один дієтичний профіль. Клас Profile зберігає специфікації харчового режиму користувача, зокрема dietType та список алергенів (allergens). Метод isAllowed(ingredient: string): boolean використовується для перевірки сумісності інгредієнта з індивідуальними харчовими обмеженнями, що є важливим при генерації меню.

Клас Recipe містить набір основних характеристик рецепту: title, ingredients, status. Метод isValid() виконує перевірку відповідності рецепту заданим критеріям (наприклад, доступності інгредієнтів, відсутності заборонених компонентів, валідності структури). Рецепти пов’язано з профілем і користувачем, що дозволяє зберігати контекст дієтичних вимог. Клас Comment реалізує можливість залишення відгуків користувачем щодо рецепту. Він містить атрибути text, rating, userId, а також метод validate(), що перевіряє коректність оцінки та вмісту повідомлення перед збереженням.

Усі асоціації на діаграмі містять кардинальності, які вказують на допустиму кількість екземплярів класів у зв’язку. Наприклад, один користувач може залишити багато коментарів до різних рецептів, але кожен коментар належить одному користувачу. Аналогічно, один рецепт може бути пов’язаний з багатьма коментарями, але зберігає приналежність до конкретного профілю.

Діаграма забезпечує структуровану основу для реалізації об’єктно-орієнтованого підходу у проєктуванні інформаційної системи та сприяє побудові чіткої, логічно обґрунтованої архітектури, що відповідає функціональним і нефункціональним вимогам, сформульованим у розділі 1

2.3 Архітектура програмного забезпечення

Архітектура програмного забезпечення визначає логічну структуру системи, взаємозв’язки між її модулями, розподіл обов’язків і способи обміну даними. Вона є основою для забезпечення стабільності, розширюваності та підтримуваності програмного продукту, особливо у випадку реалізації інформаційних систем зі складною бізнес-логікою [2, с. 91].

Інформаційна система підтримки складання меню реалізована за принципом трирівневої архітектури, що складається з:

– клієнтського рівня (frontend),

– серверного рівня (backend),

– рівня зберігання даних (база даних).

На **рисунку 2.3** наведено структурну схему архітектури програмного забезпечення, яка демонструє взаємозв’язки між основними логічними модулями системи.

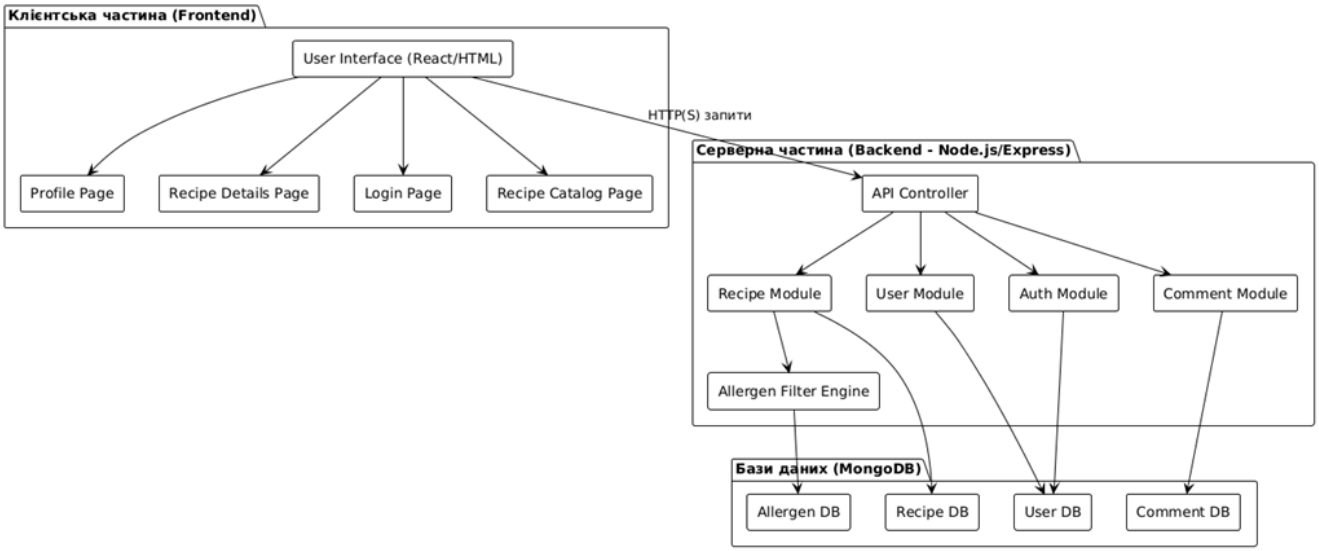


Рисунок 2.3 – Архітектура програмного забезпечення інформаційної системи складання меню

Клієнтська частина реалізована у вигляді вебінтерфейсу з використанням HTML/CSS або фреймворку React. Вона забезпечує доступ до функцій системи

через сторінки: профілю користувача (Profile Page), детального перегляду рецепту (Recipe Details Page), сторінки авторизації (Login Page) та каталогу рецептів (Recipe Catalog Page). Вся взаємодія між клієнтом і сервером відбувається через HTTP(S)-запити до API-контролера.

Серверна частина розроблена з використанням Node.js / Express, що дозволяє реалізувати REST-архітектуру із чітким розділенням функцій між модулями:

– User Module – обробка профілів користувачів;
– Recipe Module – обробка даних рецептів;
– Auth Module – автентифікація та авторизація;
– Comment Module – керування коментарями;
– Allergen Filter Engine – модуль фільтрації інгредієнтів за профілем.

Рівень баз даних реалізовано на основі MongoDB, яка забезпечує гнучке зберігання документів у форматі JSON. Бази даних поділені за типами об’єктів: User DB, Profile DB, Recipe DB, Comment DB, Allergen DB.

Узагальнені характеристики архітектурних компонентів подано у таблиці 2.2.

Таблиця 2.2

Опис основних компонентів архітектури ПЗ

Компонент	Призначення
User Interface	Графічний вебінтерфейс для взаємодії користувача з системою
Profile Page	Введення та редагування харчового профілю, алергенів, дієтичних уподобань
Login Page	Вхід у систему, реєстрація, автентифікація
API Controller	Прийом запитів від клієнта, маршрутизація до відповідних модулів
Recipe Module	Створення, редагування та видача списків рецептів
User Module	Обробка даних користувача, прив’язка до профілю
Allergen Filter Engine	Перевірка сумісності інгредієнтів з профілем користувача
Auth Module	Контроль автентифікації, ролей, доступу до закритих маршрутів
Comment Module	Додавання та зберігання оцінок/коментарів до рецептів

2.4 Діаграма компонентів

Діаграма компонентів UML використовується для візуалізації фізичної структури програмного забезпечення, зокрема відображення розподілу вихідного коду на незалежні модулі (компоненти), способу їх взаємодії та зв’язку з базами даних або зовнішніми сервісами. Такий тип діаграми дозволяє проаналізувати рівень модульності, ізоляції бізнес-логіки, гнучкість реалізації та потенціал для повторного використання або масштабування окремих частин системи.

На рисунку 2.4 наведено компонентну модель програмного забезпечення розроблюваної інформаційної системи. Структура відображає основні логічні компоненти, з яких складається вебзастосунок: клієнтський інтерфейс, контролер API, модулі обробки логіки та відповідні бази даних.

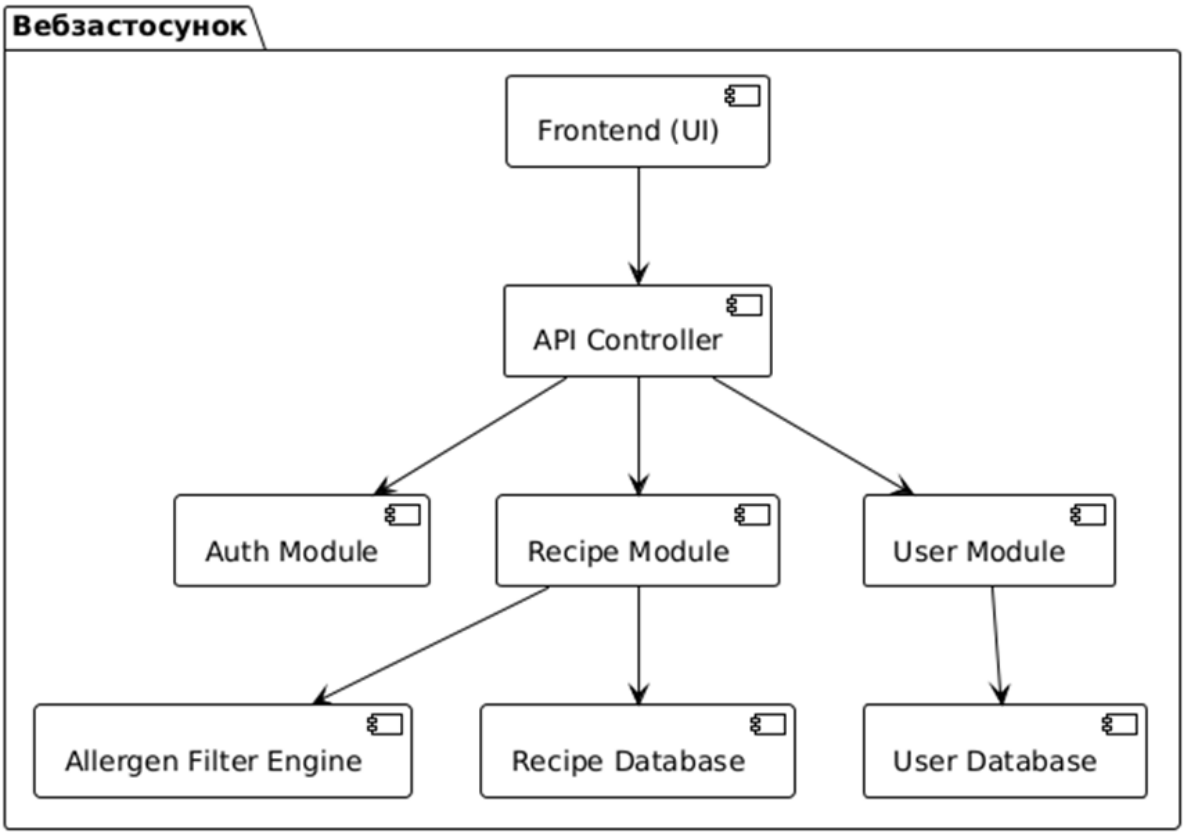


Рисунок 2.4 – Діаграма компонентів інформаційної системи складання меню

У структурі компонентів виділено:

– Frontend (UI) – клієнтський інтерфейс, що відповідає за взаємодію з користувачем. Реалізується через HTML/React-компоненти та здійснює HTTP(S)-запити до серверного API.

– API Controller – центральний координатор обробки запитів від інтерфейсу, що здійснює маршрутизацію до відповідних функціональних модулів.

– Auth Module – модуль аутентифікації, який реалізує механізми авторизації, реєстрації та захисту маршрутів.

– User Module – обробляє запити, пов’язані з профілем користувача, включаючи зчитування даних, оновлення параметрів харчування, обмежень тощо.

– Recipe Module – містить логіку формування меню, фільтрації рецептів та взаємодії з базою рецептів.

– Allergen Filter Engine – допоміжний функціональний компонент, що реалізує механізм перевірки сумісності інгредієнтів із профілем користувача.

– Бази даних – User Database, Recipe Database – відповідають за зберігання структурованих об’єктів на рівні постійної пам’яті (MongoDB або інші NoSQL-системи).

Всі залежності між компонентами реалізовані через чіткі інтерфейси з одностороннім напрямком, що дозволяє забезпечити слабкий зв’язок, підвищити рівень тестованості окремих модулів та спростити заміну/розширення бізнес-логіки у майбутньому.

Узагальнені функціональні ролі кожного компонента наведено в таблиці 2.3.

Таблиця 2.3

Опис функціональних компонентів ПЗ

Компонент	Призначення
Frontend (UI)	Графічний інтерфейс користувача, відправлення запитів до API

Продовження таблиці 2.3

API Controller	Приймання запитів, маршрутизація, виклик відповідних модулів
Auth Module	Реєстрація, авторизація, захист приватних маршрутів
User Module	Зберігання та оновлення харчового профілю, історія виборів користувача
Recipe Module	Генерація меню, обробка запитів на перегляд, фільтрацію, оцінювання рецептів
Allergen Filter Engine	Перевірка інгредієнтів на сумісність з профілем користувача
User Database	Зберігання акаунтів користувачів, профілів та алергенів
Recipe Database	Зберігання інформації про рецепти, інгредієнти, структуру меню

Запропонована компонентна модель відповідає сучасним принципам побудови сервіс-орієнтованих (SOA) вебзастосунків та підтримує можливість подальшого переходу до мікросервісної архітектури. Така структура дозволяє масштабувати окремі сервіси (наприклад, модуль генерації меню) незалежно від інших частин системи, а також реалізувати гнучкі стратегії деплойменту й обслуговування

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Система управління базою даних

Однією з базових складових під час проєктування інформаційної системи підтримки складання меню є вибір підходу до зберігання даних. Обрана модель має забезпечити ефективне представлення користувацьких профілів, дієтичних обмежень, алергенів, інгредієнтів, рецептів і результатів фільтрації, а також підтримувати персоналізовану взаємодію з системою в режимі реального часу. Враховуючи автономний характер функціонування вебзастосунку, його орієнтацію на індивідуальне використання, а також необхідність динамічного розширення структури об’єктів, було прийнято рішення використати документно-орієнтований підхід до зберігання даних на основі формату JSON.

Вибір формату JSON як основи для збереження інформації продиктований кількома критичними міркуваннями. По-перше, предметна область характеризується вкладеними структурами, змінною кількістю властивостей (наприклад, алергенів або інгредієнтів у рецепті), що потребує гнучкої, схеми-незалежної моделі зберігання. По-друге, система орієнтована на локальне розгортання або використання у складі клієнт-серверної архітектури на Node.js, де JSON є природним форматом для серіалізації об’єктів. По-третє, спрощена структура даних, що не вимагає складних транзакцій або зв’язків між сутностями, робить використання повноцінної реляційної СУБД надлишковим.

Зберігання даних реалізовано у вигляді множини окремих JSON-файлів, кожен з яких відповідає певній логічній сутності системи. Усі записи представлені у вигляді масиву об’єктів, кожен з яких має унікальний ідентифікатор та набір ключів, специфічних до типу сутності. Зчитування, додавання, оновлення та видалення інформації реалізується безпосередньо за

допомогою засобів середовища Node.js, з використанням файлової системи (модуль fs) або бібліотек для роботи з об’єктами у форматі JSON.

На рисунку 3.1 схематично представлено логіку доступу до бази даних через API модуля, який координує запити користувача та модифікацію відповідних JSON-файлів.

```
onst fs = require('fs');
const path = require('path');

const dbPath = path.join(__dirname, 'db', 'db.json');

function readDB() {
    return JSON.parse(fs.readFileSync(dbPath, 'utf-8'));
}

function writeDB(data) {
    fs.writeFileSync(dbPath, JSON.stringify(data, null, 2), 'utf-8');
}

module.exports = {
    getRecipes() {
        const db = readDB();
        return db.recipes;
    },
    addRecipe(recipe) {
        const db = readDB();
        db.recipes.push(recipe);
        writeDB(db);
    },
    getUsers() {
        const db = readDB();
        return db.users;
    },
    addUser(user) {
        const db = readDB();
        db.users.push(user);
        writeDB(db);
    }
};
```

Рисунок 3.1 – Схема взаємодії компонентів системи з файлами зберігання даних у форматі JSON

У таблиці 3.1 подано узагальнений опис структур зберігання, їх призначення та іменування файлів відповідно до реалізації.

Таблиця 3.1

Логічна структура зберігання даних у форматі JSON

Сутність	Назва файлу	Призначення
Користувачі	users.json	Дані для автентифікації, ролі, ID профілю
Харчові профілі	profiles.json	Тип дієти, алергени, налаштування персоналізації
Рецепти	recipes.json	Назва, інгредієнти, інструкція, статус, автор
Інгредієнти	ingredients.json	Назви інгредієнтів, категорії, алергенний статус
Алергени	allergens.json	Список підтримуваних алергенів (глютен, лактоза тощо)
Коментарі	comments.json	Оцінки, текст коментаря, посилання на рецепт і користувача
Звіти (опціонально)	reports.json	Лог сформованих меню, калорійність, макронутрієнти, дата створення

Використання JSON-файлів дозволяє легко адаптувати модель до змін у бізнес-логіці без необхідності модифікації схеми або структури таблиць, як це властиво класичним SQL-системам. Крім того, такий підхід підтримує просту інтеграцію з інтерфейсами на JavaScript, швидке прототипування та забезпечує незалежність від зовнішніх СУБД при розгортанні в автономному або serverless-середовищі.

Загалом, обраний підхід до зберігання є доцільним для задач, пов’язаних із персоналізованим складанням меню та обробкою харчових профілів, і дозволяє забезпечити гнучкість, простоту й ефективність реалізації інформаційної системи.

3.2 Розробка інформаційної бази

. Фізична модель інформаційної бази визначає структуру зберігання даних на рівні реалізації, включаючи типи даних, зовнішні ключі, обмеження цілісності та зв’язки між таблицями. У процесі реалізації інформаційної системи підтримки прийняття рішень при складанні меню здорового харчування було розроблено

фізичну модель бази даних, яка базується на реляційному підході та повністю адаптована до особливостей предметної області.

На рисунку 3.2 представлено фізичну модель бази даних у вигляді ER-діаграми, яка включає основні таблиці (User, Profile, Recipe, Ingredient, Comment, Allergen) та таблиці зв’язку (Recipe_Ingredient, Profile_Allergen). Кожна таблиця має чітко визначений первинний ключ, типи полів (наприклад, VARCHAR, INT, DECIMAL, DATETIME) та зовнішні ключі, що формують відношення між таблицями.

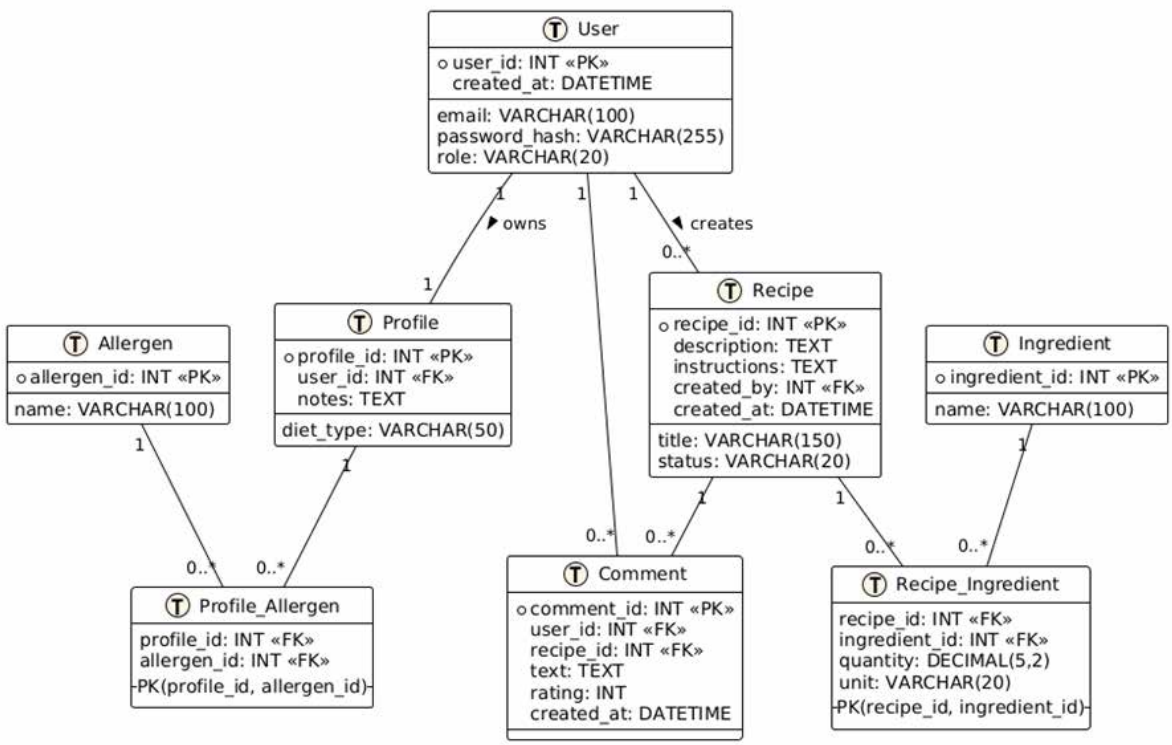
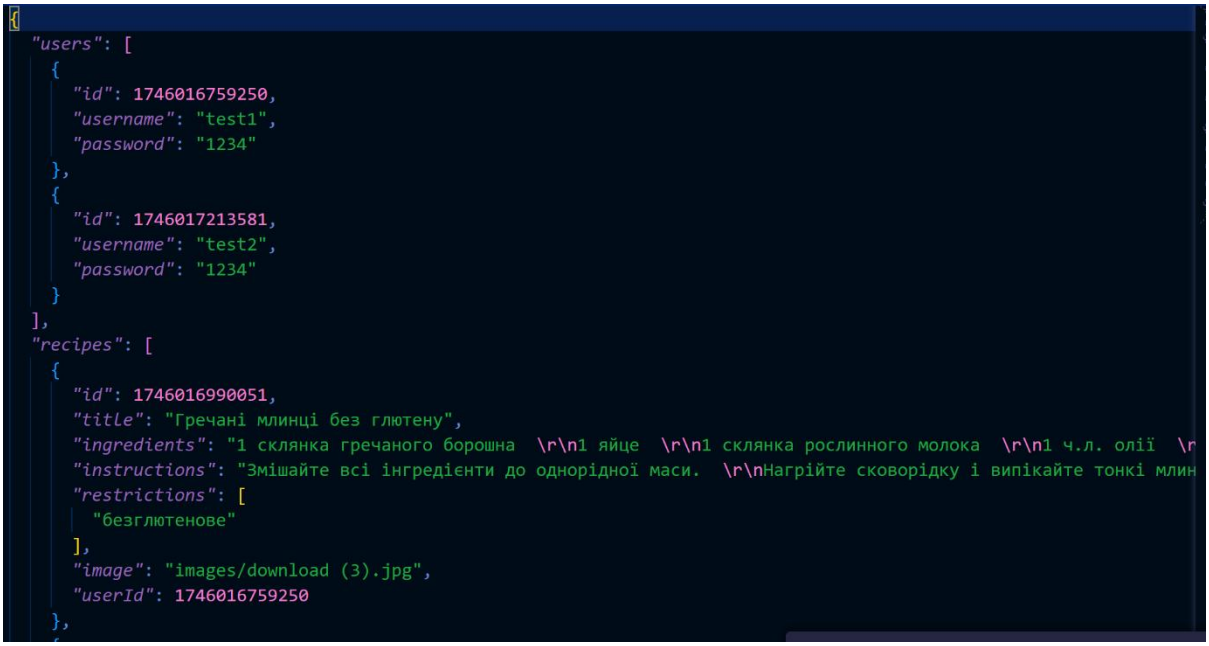


Рисунок 3.2 – Діаграма фізичної структури інформаційної бази

Структура бази даних дозволяє забезпечити збереження облікових записів користувачів, дієтичних профілів, алергенів, інгредієнтів, рецептів, відгуків і кількісного складу страв.

Фізична структура оптимізована для обробки запитів, які враховують персоналізовані харчові профілі, обмеження на інгредієнти (наприклад, глютен, лактоза) та дозволяє реалізувати логіку динамічної фільтрації рецептів. Ця модель лягла в основу структури фізичного зберігання у форматі JSON. У рамках реалізації з використанням Node.js та файлової підсистеми (fs), інформація

організована у вигляді вкладених об’єктів з ідентифікаторами, що забезпечують зв’язок між сутностями. Приклад такої структури наведено на рисунку 3.3.



```
{
  "users": [
    {
      "id": 1746016759250,
      "username": "test1",
      "password": "1234"
    },
    {
      "id": 1746017213581,
      "username": "test2",
      "password": "1234"
    }
  ],
  "recipes": [
    {
      "id": 1746016990051,
      "title": "Гречані млинці без глютену",
      "ingredients": "1 склянка гречаного борошна  \r\n1 яйце  \r\n1 склянка рослинного молока  \r\n1 ч.л. олії  \r
      "instructions": "Змішайте всі інгредієнти до однорідної маси.  \r\nНагрійте сковорідку і випікайте тонкі млин
      "restrictions": [
        "безглютенове"
      ],
      "image": "images/download (3).jpg",
      "userId": 1746016759250
    },
```


Рисунок 3.3 – Фрагмент JSON-файлу інформаційної бази з даними користувачів і рецептів

У прикладі зображено об'єкти users та recipes, які серіалізовані у вигляді масивів. Кожен користувач має унікальний id, username, password, а кожен рецепт зберігає title, список інгредієнтів, інструкції, перелік алергенних обмежень (restrictions) та посилання на автора (userId).

Узагальнена структура інформаційної бази подана в таблиці 3.2, яка описує сутності, їхні атрибути та зв’язки.

Таблиця 3.2

Опис логічних сутностей інформаційної бази

Сутність	Ключові атрибути	Зв’язки
User	user_id (PK), email, password_hash, role	1:1 з Profile, 1:N з Recipe, 1:N з Comment
Profile	profile_id (PK), diet_type, notes	M:N з Allergen через Profile_Allergen
Recipe	recipe_id (PK), title, instructions, created_by	M:N з Ingredient, 1:N з Comment
Ingredient	ingredient_id (PK), name	–
Allergen	allergen_id (PK), name	–

Продовження таблиці 3.2

Comment	comment_id (PK), text, rating, recipe_id, user_id	–
Recipe_Ingredient	recipe_id (FK), ingredient_id (FK), quantity, unit	Зв’язок між Recipe та Ingredient
Profile_Allergen	profile_id (FK), allergen_id (FK)	Зв’язок між Profile та Allergen

Інформаційна база побудована з урахуванням принципів нормалізації та узгодженості, що дозволяє уникнути надмірності, забезпечує підтримку персоналізації, фільтрації за алергенами та ефективну обробку запитів до об’єктів рецептурного планування. Структура даних дозволяє системі масштабуватись без порушення логіки взаємодії між об’єктами.

3.3 Вибір інструментарію для створення прикладного програмного забезпечення

Для ефективної реалізації функціоналу інформаційної системи підтримки складання меню необхідно обрати технології, які забезпечують зручну роботу з динамічними структурами даних, підтримку JSON, реалізацію REST-архітектури, а також побудову адаптивного інтерфейсу користувача. Враховуючи автономний або гібридний характер системи, основними критеріями при виборі інструментів стали: гнучкість розробки, швидкість прототипування, низькі вимоги до розгортання, широка підтримка спільноти, а також відкритість екосистеми.

У таблиці 3.3 наведено основні компоненти інструментального середовища, що використовуються під час створення прикладного програмного забезпечення, а також їх функціональне призначення.

Таблиця 3.3

Обраний інструментарій розробки програмного забезпечення

Компонент	Інструмент / Технологія	Призначення

Продовження таблиці 3.3

Мова програмування	JavaScript	Основна мова як для frontend (React), так і для backend (Node.js)
Середовище виконання	Node.js	Серверне середовище для реалізації API, роботи з файлами JSON
Фреймворк backend	Express.js	Легковагий вебфреймворк для створення REST-API, обробки запитів
Бібліотека frontend	HTML/CSS/React (або HTML + JS)	Формування інтерактивного інтерфейсу користувача
Формат даних	JSON	Формат зберігання структурованих даних, зручний для серіалізації об’єктів
Сховище даних	Файли JSON (fs module)	Локальна файлова база без потреби в окремому СУБД-сервері
Середовище розробки	Visual Studio Code	Редактор коду з підтримкою плагінів, контролю версій, налагодження
Менеджер пакетів	npm	Управління залежностями проєкту, бібліотеками та утилітами
PDF-звітність	pdfkit / jsPDF	Формування результатів користувача у форматі PDF
Контроль версій (опціонально)	Git	Відстеження змін, зручна командна робота (локально або через GitHub)

системи та інтеграцію з API-контролерами, які реалізують бізнес-логіку системи. Express.js дозволяє швидко розгортати серверну частину з мінімальним обсягом коду, а зберігання даних у вигляді JSON-файлів значно спрощує логіку доступу до інформації в умовах невеликої системи без потреби у високій транзакційності.

Користувацький інтерфейс реалізовано засобами HTML та JavaScript з можливістю розширення до React, що дозволяє створити адаптивну, зручну та інтуїтивно зрозумілу взаємодію користувача з системою. Додатково застосовуються бібліотеки для генерації PDF-звітів, які надають можливість формувати результати обчислень (калорійність, списки продуктів тощо) у форматі, придатному для збереження або друку.

Обраний інструментарій є оптимальним для реалізації персоналізованої системи складання меню, забезпечує підтримку всіх функціональних вимог та дозволяє розгортання як у локальному середовищі, так і в серверному оточенні без складних залежностей.

3.4 Алгоритмізація програмних модулів

Алгоритмізація є ключовим етапом у процесі реалізації програмного забезпечення, що дозволяє формалізувати логіку обробки даних, забезпечити керованість бізнес-процесів та оптимізувати взаємодію між функціональними модулями. У межах розробки інформаційної системи для складання меню здорового харчування було визначено низку критичних функціональних сценаріїв, серед яких особливе значення має реєстрація користувача з формуванням персонального харчового профілю.

Цей процес реалізовано у вигляді модульного алгоритму, що включає декілька етапів: обробку вхідних даних, перевірку унікальності, валідацію, формування об’єктів профілю, їх серіалізацію у форматі JSON та перенаправлення користувача до основного функціонального модуля — каталогу рецептів.

На рисунку 3.3 подано блок-схему алгоритму реєстрації нового користувача. Схема реалізована відповідно до умовно-графічних позначень ДСТУ 3008:2015: початкові та кінцеві дії оформлено овальними блоками, логічні перевірки — у вигляді ромбів, дії — у прямокутниках, а стрілки визначають напрямок виконання.

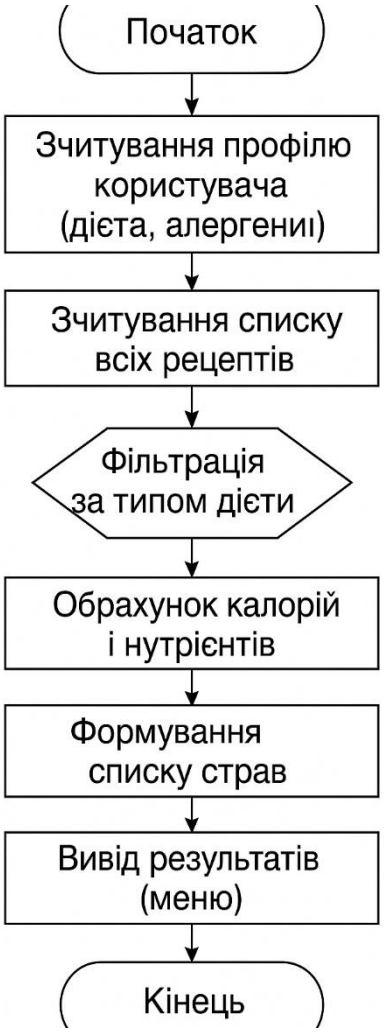


Рисунок 3.3 – Блок-схема алгоритму створення облікового запису та профілю користувача

Алгоритм передбачає такі основні етапи:

1. Ініціалізація процесу — запуск сценарію з вхідної точки «Початок».
2. Введення облікових даних — користувач вводить логін і пароль у відповідну форму.
3. Перевірка унікальності логіна — на цьому етапі система здійснює пошук у файлі users.json (або відповідному сховищі) щодо наявності логіна.
4. Гілкування за результатом перевірки:

- якщо логін уже існує, алгоритм завершується або повертає користувача на попередній крок;
- якщо логін унікальний, система переходить до збирання додаткової інформації.

5. Формування профілю — введення типу дієти, алергенів, особливих приміток та створення об’єкта профілю.

6. Збереження у базу — створені об’єкти user і profile додаються до відповідних JSON-файлів (наприклад, users.json, profiles.json) за допомогою функцій модуля файлової обробки (fs).

7. Перенаправлення — після успішного збереження здійснюється перенаправлення користувача до сторінки каталогу рецептів, де запускається наступний етап логіки системи — персоналізована фільтрація.

З технічного боку, обробка даних реалізована засобами Node.js, використовуючи асинхронні виклики читання і запису у форматі JSON. Автоматична перевірка унікальності, формування ID (Date.now()) та базова валідація забезпечують стійкість до помилок користувача.

Алгоритм реєстрації реалізовано як окремий модуль (authController) з інкапсульованими функціями:

- isLoginTaken(login)
- createUser(data)
- createProfile(userId, profileData)

Це дозволяє повторно використовувати модуль в інших частинах системи — зокрема при оновленні профілю або створенні адміністративних облікових записів.

\

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

4.1 Розробка користувацького інтерфейсу з урахуванням інклюзивності

Користувацький інтерфейс (UI) є ключовим елементом вебзастосунку, оскільки саме він визначає зручність, доступність та ефективність взаємодії користувача із системою. У контексті проєкт, особливу увагу було приділено принципам інклюзивного дизайну, що ґрунтуються на рекомендаціях стандарту WCAG 2.1 (Web Content Accessibility Guidelines). Відповідно до цього стандарту, інтерфейс має бути: сприйнятним (perceivable), керованим (operable), зрозумілим (understandable) та надійним (robust).

У рамках реалізації було розроблено адаптивну, модульну структуру інтерфейсу, що забезпечує підтримку користувачів із порушенням зору, потребою в масштабуванні шрифтів, керуванні з клавіатури та використанні допоміжних технологій (екранні рідери). Розробка UI здійснювалась із використанням HTML5, CSS3 та JavaScript, без залучення важких сторонніх бібліотек, що дозволило досягти високої швидкодії та контролю над доступністю елементів.

На рисунку 4.1 представлено загальну схему розміщення основних елементів інтерфейсу сторінки додавання рецепта. Дизайн структурується навколо логіки модулів: заголовок, форма з доступними полями вводу, кнопки дій, повідомлення про помилки та футер.

```
<body>
  <header>
    <div class="container header-flex">
      <h1 class="site-title">Особливі рецепти</h1>
      <nav class="main-nav" id="nav-menu">
        <a href="index.html">Головна</a>
        <a href="recipes.html">Рецепти</a>
        <a href="add-recipe.html">Додати рецепт</a>
        <a href="login.html" id="login-link">Вхід</a>

      </nav>
    </div>
  </header>
```

Рисунок 4.1 – Структура адаптивної форми додавання рецепта з підтримкою WCAG

На рисунку 4.2 представлено головну сторінку вебзастосунку, яка реалізує концепцію доступної та структурованої навігації. Інтерфейс поділений на логічні секції: горизонтальне навігаційне меню у верхній частині, пошуковий модуль із підписаним полем введення та кнопкою дії, блоки популярних рецептів та останніх новин. Усі компоненти мають належну семантичну розмітку та підтримку ARIA-атрибутів для коректної роботи з екранними рідерами

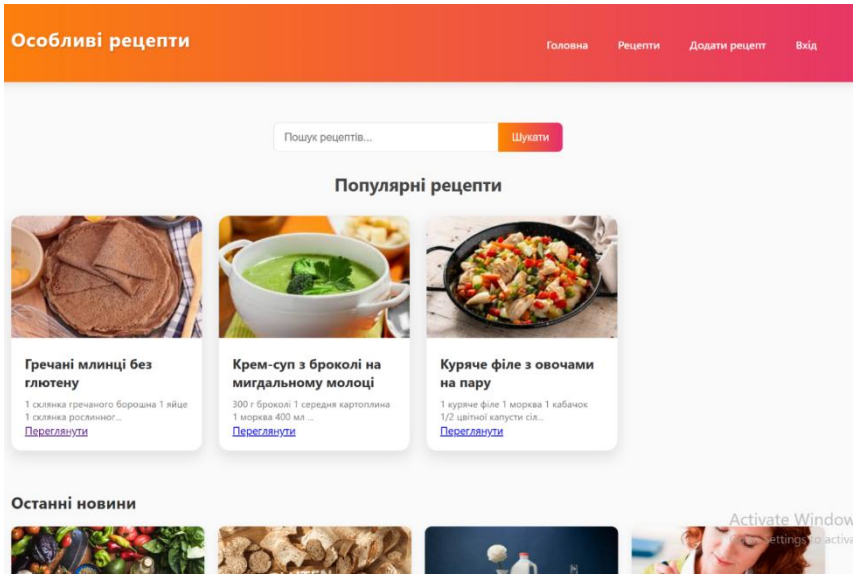


Рисунок 4.2 – Головна сторінка вебзастосунку з адаптивним інтерфейсом та підтримкою інклюзивності

На зображенні видно, що кожна картка рецепта містить:

- зображення страви (із alt-описом),
- заголовок рецепта (<h3>),
- короткий перелік інгредієнтів (обмежений за кількістю символів),
- інтерактивне посилання на повну версію (<a> із підписом "Переглянути").

Пошукове поле має атрибути aria-label та placeholder, а кнопка "Шукати" має достатню площу натискання (мінімум 44x44 px), що відповідає рекомендаціям [10].

Кольорова гама (поєднання білого фону, чорного тексту та яскравих акцентів) відповідає критерію контрастності WCAG AA. Усі елементи можна активувати з клавіатури, а компоненти навігації (<nav>) підтримують фокусування без втрати контексту.

Кольорова палітра інтерфейсу побудована з урахуванням вимог до контрастності (мінімум 4.5:1 для тексту), а текстові елементи мають збільшений базовий розмір (18px), що полегшує сприйняття візуального контенту.

Також реалізовано логічну навігацію через клавішу Tab, а активні елементи підсвічуються стилем :focus, що забезпечує повноцінну клавіатурну навігацію. Усі текстові поля, кнопки та зображення супроводжуються відповідними підписами, атрибутами aria-label, alt або aria-describedby, що гарантує сумісність із екранними рідерами.

4.2 Реалізація функціональності додатку

У межах даного етапу проєкту було здійснено безпосередню реалізацію функціональних компонентів вебзастосунку відповідно до раніше визначених архітектурних і концептуальних моделей. Реалізація виконувалася з урахуванням принципів модульності, інклюзивності, персоналізації та масштабованості. Система побудована як сукупність взаємопов’язаних модулів, кожен з яких відповідає за конкретний функціональний блок: автентифікацію,

обробку профілів, управління рецептами, фільтрацію контенту, зворотний зв’язок та інформаційне наповнення.

Основні функціональні можливості вебзастосунку наведено у таблиці 4.1, яка узагальнює логіку розподілу функціоналу за модулями.

Таблиця 4.1

Основна реалізована функціональність вебзастосунку

№	Назва функціональності	Короткий опис	Реалізовано
1	Реєстрація користувача та створення профілю	Створення облікового запису з обліком дієтичних обмежень	✓
2	Авторизація та вхід у систему	Перевірка користувача, доступ до персонального профілю	✓
3	Додавання нових рецептів	Форма для введення назви, інгредієнтів, інструкцій	✓
4	Пошук рецептів за назвою	Пошук у реальному часі за ключовими словами	✓
5	Фільтрація рецептів за харчовими обмеженнями	Виведення лише сумісних із профілем страв	✓
6	Редагування користувацького профілю	Оновлення обмежень, дієт, персональної інформації	✓
7	Перегляд детальної інформації про рецепт	Виведення опису, інгредієнтів, фото, попередження про алергени	✓
8	Перегляд інформаційних новин	Блок новин на головній сторінці з текстом та зображенням	✓
9	Система повідомлень про помилки / обмеження	Повідомлення користувачеві про недоступні дії або порушення	✓
10	Захист адміністративної панелі	Перевірка ролі, доступ лише для авторизованого адміністратора	✓

Реалізація функціональних блоків охоплює повний цикл взаємодії користувача з системою — від створення профілю до персоналізованого перегляду вмісту. Функціональність також підтримує доступність, інформативність і захищеність даних, відповідно до вимог, визначених у попередніх розділах.

У подальших підпунктах буде детально розглянуто реалізацію кожної ключової функції системи:

- у підпункті 4.2.1 — створення облікового запису та дієтичного профілю;
- у 4.2.2 — додавання рецептів та фільтрація за харчовими обмеженнями;
- у 4.2.3 — механізм пошуку рецептів;
- у 4.2.4 — редагування користувацького профілю;
- у 4.2.5 — перегляд новин та кулінарного контенту.

Ця структура дозволяє забезпечити повноцінний інтерфейс персоналізованої взаємодії, що адаптується під індивідуальні потреби користувачів із різними дієтичними обмеженнями.

4.2.1 Реалізація механізму реєстрації та створення користувацького профілю

Одним із базових функціональних модулів системи є механізм створення облікового запису та авторизації користувача. Цей етап є критично важливим, оскільки забезпечує персоналізований доступ до інформаційного середовища вебзастосунку та формує основу для подальшого застосування дієтичних обмежень, фільтрації рецептів і збереження історії взаємодії з контентом.

Процес реєстрації реалізовано у вигляді адаптивної вебформи, що включає два поля: логін (ідентифікатор користувача) та пароль. Після введення даних формується об’єкт користувача, який зберігається у відповідному сховищі (users.json) разом із базовими атрибутами профілю (email, hash пароля, роль, дата створення тощо). Реєстрація супроводжується автоматичною перевіркою на унікальність логіна, валідацією обов’язкових полів та повідомленням про успіх чи помилку. Інтерфейс реалізовано з урахуванням інклюзивності, з підтримкою навігації з клавіатури та атрибутів доступності (ARIA).

На рисунку 4.3 представлено зовнішній вигляд форми реєстрації:

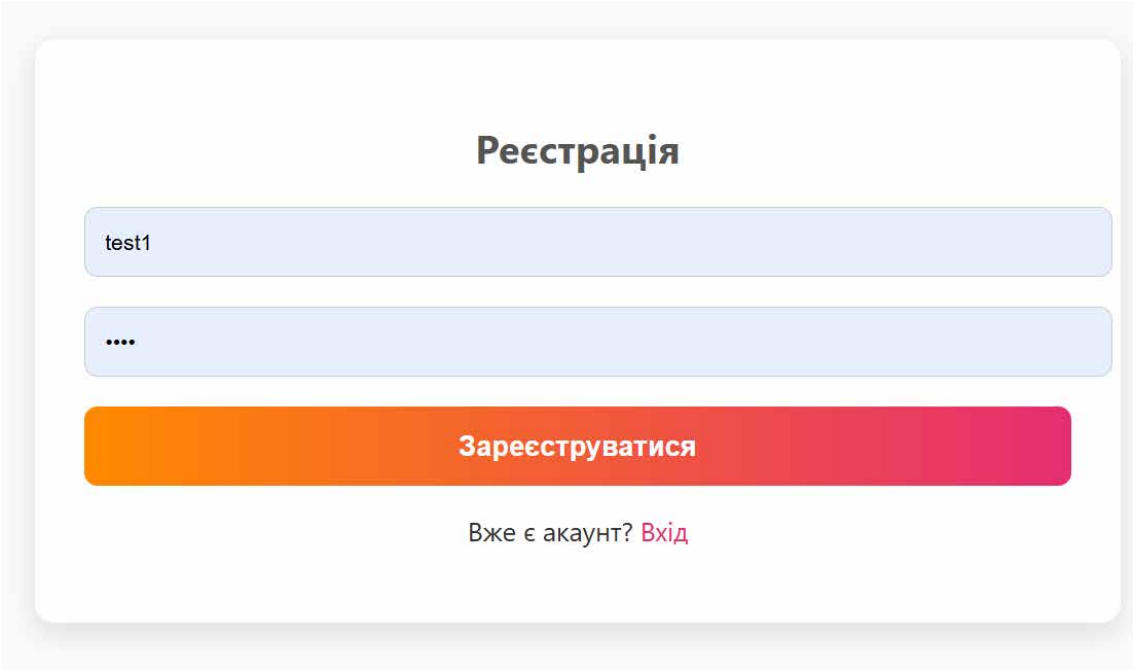


Рисунок 4.3 – Інтерфейс реєстрації користувача з кнопкою «Зареєструватися»

Після успішної реєстрації користувач може перейти до форми входу в систему, де реалізовано перевірку автентичності введених даних. Якщо облікові дані коректні — запускається сесія та відбувається переадресація на головну сторінку із персоналізованим доступом. У разі помилки — виводиться повідомлення, що допомагає користувачеві виправити введену інформацію.

На рисунку 4.4 зображено форму входу (авторизації), яка стилістично відповідає загальному дизайну системи:

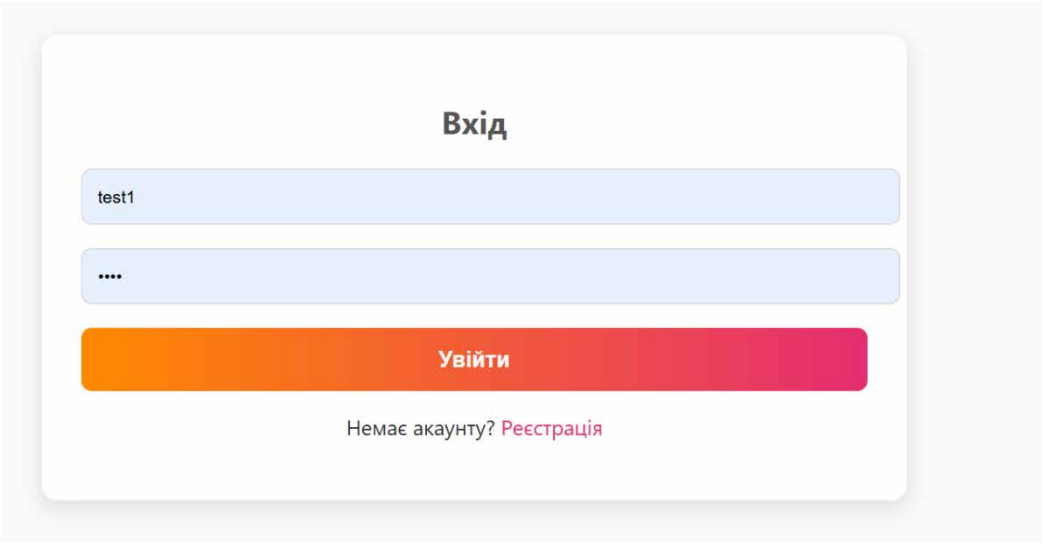


Рисунок 4.4 – Інтерфейс входу до системи з кнопкою «Увійти»

У реалізації обох форм передбачено використання елементів захисту:

1. обов’язкова HTTPS-передача даних;
2. хешування паролів з використанням криптографічної функції (наприклад, bcrypt);
3. перевірка довжини та складності пароля (не менше 6 символів);
4. обмеження на повторну реєстрацію з тим самим логіном;
5. збереження ролі користувача (user / admin) для подальшого контролю доступу до функцій системи.

Реалызація реєстрації та входу забезпечує надійний стартовий етап роботи користувача з інформаційною системою, дозволяє ідентифікувати профіль, зберігати дієтичні параметри та захищати доступ до персоналізованого функціоналу.

4.2.2 Функціональність додавання рецептів за харчовими обмеженнями

Однією з ключових можливостей вебзастосунку є підтримка додавання нових рецептів із вказанням специфічних дієтичних параметрів. Ця функціональність забезпечує персоналізацію контенту відповідно до харчових обмежень користувача — глютенова непереносимість, веганство, низький глікемічний індекс тощо. Реалізація передбачає використання інтерактивної вебформи, яка дозволяє вводити назву страви, список інгредієнтів, покрокову інструкцію приготування, харчові особливості та завантаження зображення.

На рисунку 3.6 представлено інтерфейс додавання рецепта, який має адаптивну структуру і побудований з урахуванням принципів доступності.

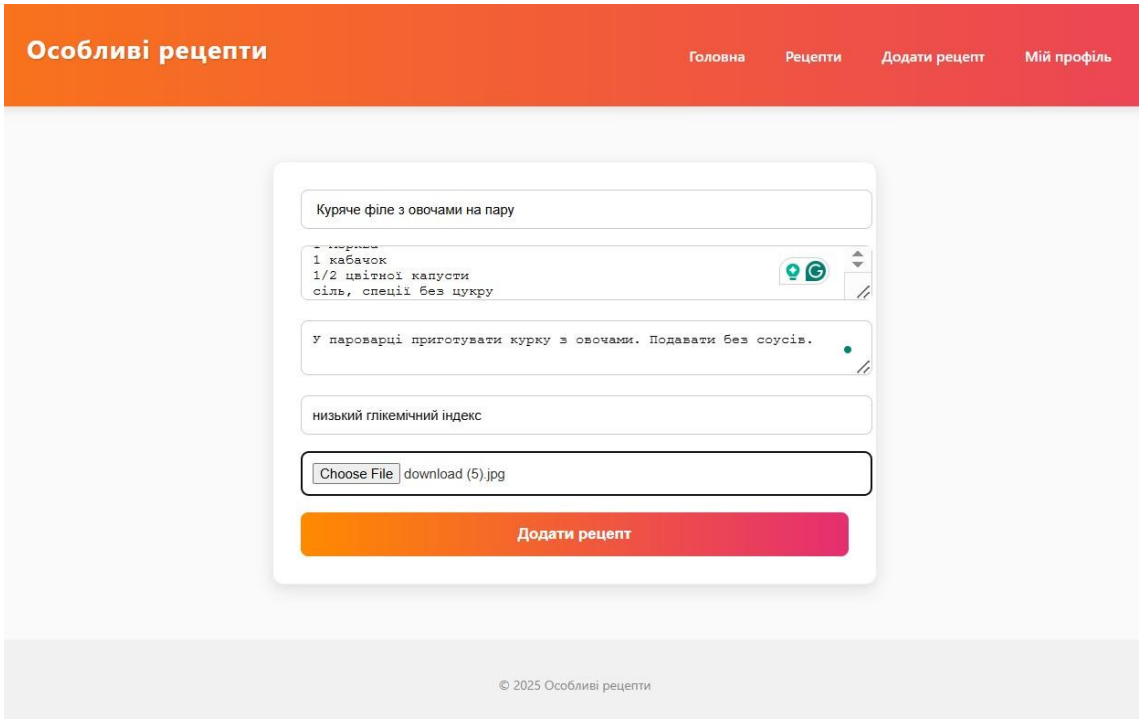


Рисунок 4.5 – Форма додавання нового рецепта з дієтичним маркуванням

Форма містить наступні поля:

- назва рецепта;
- інгредієнти (у вільній формі з можливістю багаторазового введення);
- інструкції до приготування;
- характеристика страви (наприклад, “низький глікемічний індекс”);
- поле для вибору зображення;
- кнопка підтвердження.

Після підтвердження дані передаються у backend-сервер через HTTP POST-запит, де зберігаються у відповідному масиві recipes.json. Перед збереженням система здійснює валідацію полів, зокрема:

- перевірку на порожні значення;
- обмеження на довжину опису;
- перевірку формату зображення (типи .jpg, .png, розмір до 2 МБ).

Додатково для полегшення пошуку і фільтрації, у структурі об’єкта рецепта окремо зберігається список харчових обмежень, які надалі використовуються для формування персоналізованого каталогу.

На рисунку 4.6 продемонстровано вигляд каталогу рецептів, сформованого після додавання записів:

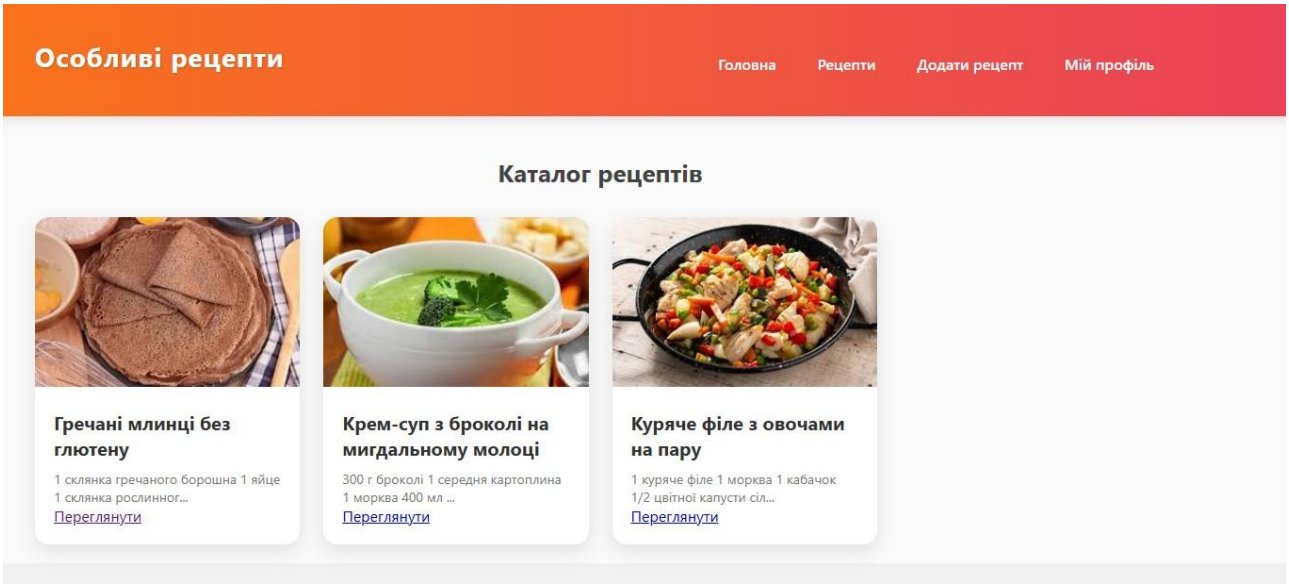


Рисунок 4.6 – Вивід каталогу рецептів із картками страв

Рецепти виводяться у вигляді візуальних карток, які містять зображення, назву страви, короткий опис інгредієнтів і кнопку «Переглянути». Для кожного користувача застосовується автоматичний фільтр — рецепти, що містять інгредієнти, які суперечать профілю користувача (наприклад, лактозу чи глютен), приховуються з виводу. Цей механізм побудований на порівнянні дієтичних тегів рецепта із параметрами, збереженими в обліковому записі користувача.

У профілі користувача, як зображено на рисунку 4.7, реалізовано окремий блок для відображення особисто доданих рецептів. Це забезпечує зручний доступ до власних публікацій і можливість подальшого редагування.

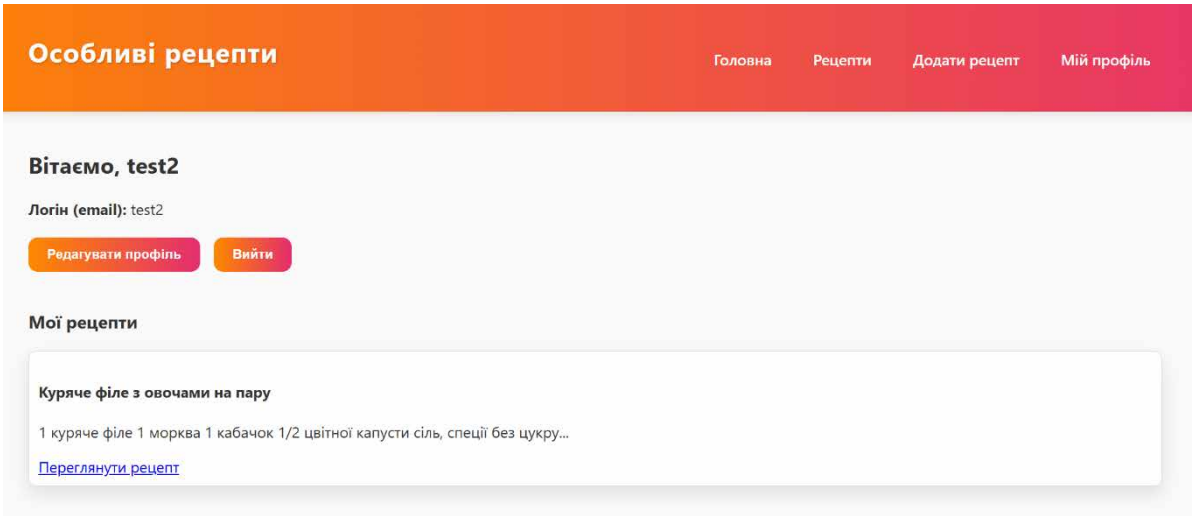


Рисунок 4.7 – Інтерфейс особистого кабінету з переліком власних рецептів

Функціональність додавання рецептів не лише реалізує базовий CRUD-функціонал, а й підтримує глибоку персоналізацію контенту, що має ключове значення для цільової аудиторії — осіб з особливими харчовими потребами. З технічної точки зору, така реалізація спрощує масштабування системи, оскільки дієтичні фільтри легко розширити новими категоріями або винести в окрему систему тегів.

4.2.3 Реалізація пошуку рецептів у загальній базі даних

Функціональність пошуку є необхідною складовою зручної навігації в межах вебзастосунку, що містить значну кількість рецептів. Вона дозволяє користувачу оперативно знайти потрібну страву за ключовим словом, зокрема за частиною назви або інгредієнтом. Механізм реалізовано через інтерактивне текстове поле з кнопкою підтвердження запиту.

Клієнтська частина обробляє введений рядок і відправляє його у вигляді параметра до функції фільтрації, яка виконується безпосередньо на стороні браузера. Це дозволяє реалізувати швидкий пошук без потреби у серверному запиті. Для точного співпадіння застосовується метод часткового порівняння назви рецепта у нижньому регістрі, що дозволяє враховувати варіативність

введення (наприклад, «крем-суп» знайде «Крем-суп з броколі на мигдальному молоці»).

На рисунку 4.8 зображено приклад використання пошуку, де користувач здійснює пошук за ключовим словом «Крем-суп».

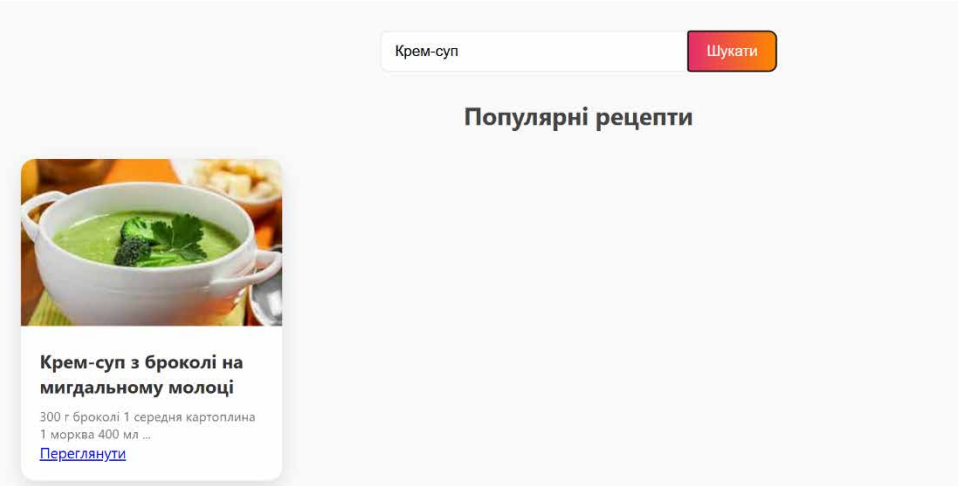


Рисунок 4.9 – Пошук рецептів за ключовим словом у загальному каталозі

У результаті пошуку користувач отримує лише ті рецепти, що відповідають введеному запиту, у вигляді адаптивних карток. Якщо співпадінь не знайдено, система повертає порожній список без помилок, що відповідає вимогам до коректної обробки запитів.

Форма пошуку містить наступні елементи: поле введення з атрибутом placeholder, кнопка підтвердження, стилізація у корпоративному кольорі. У разі потреби можна розширити пошук на інші поля, такі як інгредієнти, дієтичні теги або спосіб приготування, шляхом адаптації логіки фільтрації.

Загальна реалізація пошуку дозволяє покращити користувацький досвід і прискорити взаємодію з системою в умовах великої кількості записів у базі.

4.2.4 Механізм редагування профілю користувача

Функціональність редагування профілю користувача є важливою складовою персоналізованої взаємодії з системою. Вона дозволяє змінювати основні облікові дані, такі як логін (email), у разі потреби актуалізації або

виправлення введеної інформації. Цей механізм реалізовано з дотриманням вимог до збереження цілісності ідентифікаторів користувача та контролю доступу до редагування.

Процес редагування реалізовано у вигляді простої вебформи, яка доступна лише авторизованим користувачам. Після введення нового значення поля здійснюється валідація введених даних та запис змін до відповідного об’єкта користувача в базі (JSON-файлі). Підтвердження змін супроводжується збереженням нового значення та автоматичним оновленням відображення на сторінці профілю.

На рисунку 4.10 зображено інтерфейс редагування облікового запису користувача.

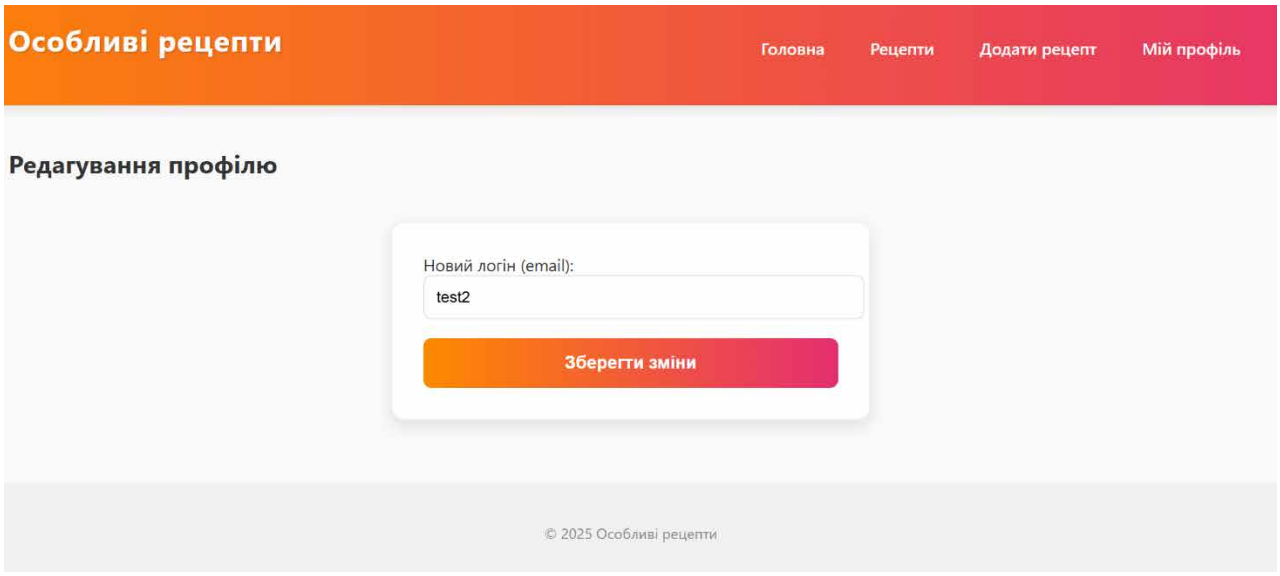


Рисунок 4.10 – Сторінка редагування логіна користувача у профілі

Форма складається з одного поля введення для нового логіна (email) та кнопки підтвердження «Зберегти зміни». Після надсилання запиту зміни зберігаються локально (у форматі JSON), після чого користувач перенаправляється назад до особистого кабінету. У разі виникнення помилки, наприклад спроби вказати вже зареєстрований логін, система інформує про конфлікт та не зберігає зміни.

Особливу увагу приділено стилю елементів: кнопка має яскраве градієнтне оформлення, поле має достатній розмір для зручного введення, а також чіткий фокус при навігації з клавіатури.

З технічної точки зору, зміна логіна є чутливою операцією, тому реалізовано обмеження на доступ до сторінки редагування лише для автентифікованих користувачів, а також контроль того, щоб зміни не торкалися інших атрибутів профілю або сторонніх облікових записів.

Реалізація цієї функції підвищує гнучкість використання системи та дозволяє користувачу самостійно керувати власними обліковими даними у безпечний спосіб.

4.2.5 Функціональність перегляду рецептів та новин у системі

Розроблена система надає користувачеві можливість не лише додавати інформацію, але й переглядати її у повноцінному форматі. Функціональність перегляду є завершальним етапом логічної взаємодії з контентом — вона забезпечує доступ до розгорнутої інформації про рецепти, включаючи інгредієнти, інструкції з приготування, харчові обмеження та зображення страви. Аналогічно реалізовано механізм перегляду новин, який дозволяє систематизувати інформаційний супровід щодо здорового харчування, порад та оновлень.

Перегляд рецепта реалізовано у вигляді структурованої сторінки з такими елементами: – назва страви у верхній частині; – зображення рецепта; – блок з інгредієнтами у форматі маркованого списку; – текстова інструкція з приготування; – перелік дієтичних властивостей.

На рисунку 4.11 продемонстровано приклад сторінки перегляду рецепта, оформленої відповідно до стандартів адаптивного дизайну.

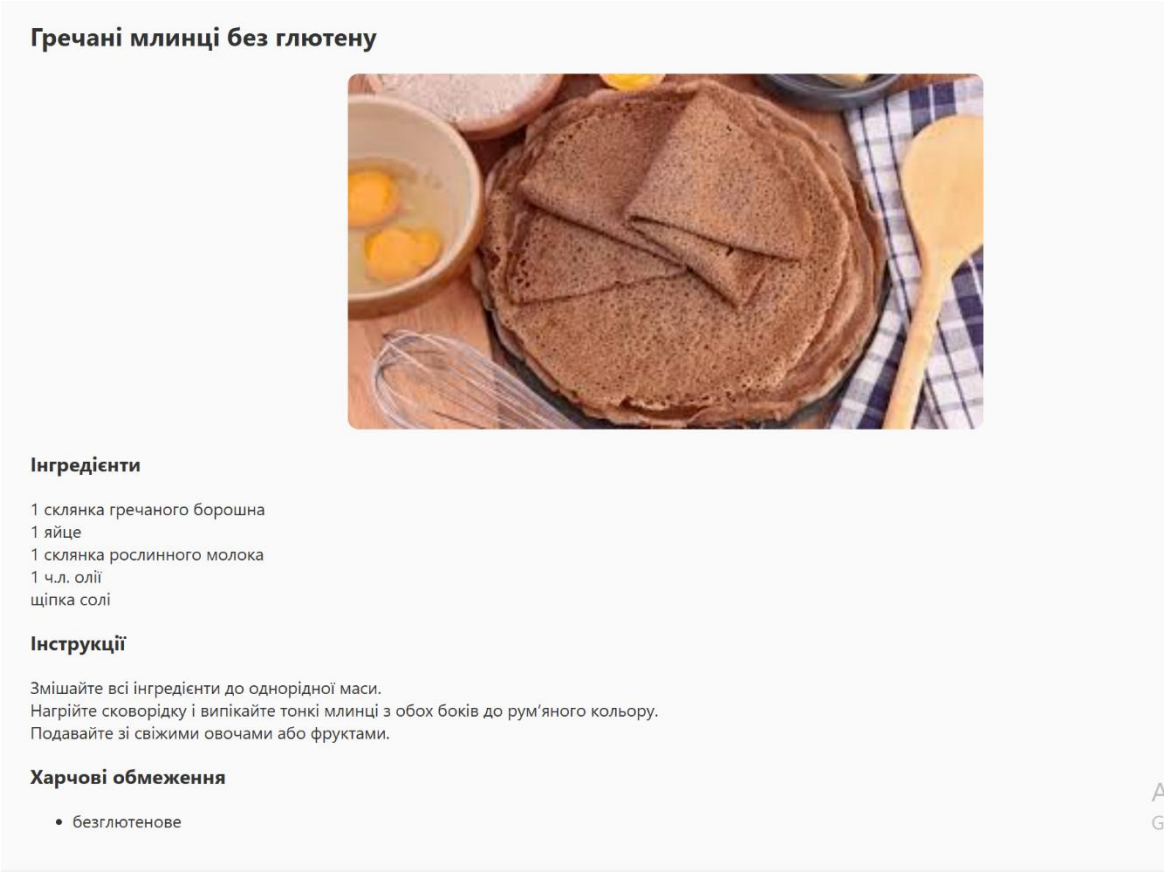


Рисунок 4.11 – Повна сторінка рецепта з описом і харчовими властивостями

Усі елементи рецепта розміщено логічно та послідовно, що дозволяє користувачеві швидко орієнтуватися у структурі документа. Для поліпшення доступності застосовано чітке шрифтове виділення ключових розділів, контрастність тексту і підтримку масштабування.

Механізм перегляду новин реалізовано за подібною структурною схемою. Кожна новина має унікальний ідентифікатор і відкривається у вигляді окремої сторінки з заголовком, зображенням та текстовим блоком. Інформація зберігається у масиві об'єктів на сервері, де кожен елемент містить заголовок, опис і зображення. Вивід новини здійснюється на основі передачі параметра id у запиті.

На рисунку 4.12 наведено приклад сторінки перегляду новини у системі.

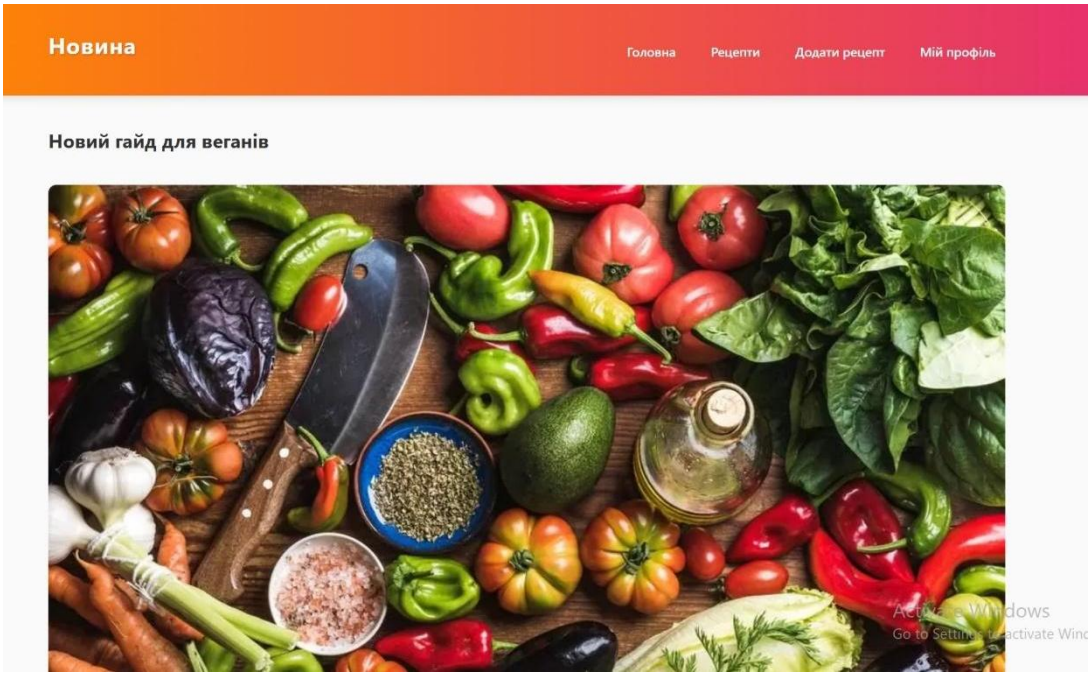


Рисунок 4.12 – Перегляд повної новини в окремому вікні

Функціональність перегляду реалізована однаково як для автентифікованих, так і для гостьових користувачів, що підвищує загальну відкритість системи та її інформаційну корисність. Завдяки цьому розділ новин виконує роль просвітницького інструмента, який доповнює персоналізовані дані про харчування об’єктивною інформацією з галузі.

З технічної точки зору, сторінки перегляду формуються динамічно на основі передачі параметра через URL. Далі JavaScript-логіка завантажує відповідний елемент з бази, виводить дані у відповідні HTML-блоки та застосовує стилі до візуальних компонентів.

Реалізація перегляду контенту завершує повний життєвий цикл даних у системі: від додавання користувачем — до доступу та споживання цільовою аудиторією. Впроваджений підхід дозволяє гнучко розширювати інформаційну базу та забезпечує повноцінну взаємодію з користувачем на рівні інтерфейсу та контенту.

4.3 Тестування системи та верифікація результатів

Після реалізації функціональних модулів інформаційної системи було проведено повне тестування її працездатності з метою виявлення помилок, перевірки логіки обробки даних, відповідності вимогам доступності та перевірки на відповідність функціональним сценаріям, описаним у попередніх підрозділах. Основну увагу було зосереджено на тестуванні критичних шляхів взаємодії користувача з системою, зокрема: реєстрації, додавання та фільтрації рецептів, перегляду контенту, пошуку та редагування профілю.

Тестування виконувалося вручну відповідно до заздалегідь сформованих тест-кейсів, що охоплюють як позитивні, так і негативні сценарії. Таблиця 3.4 демонструє результати функціонального тестування ключових сценаріїв.

Таблиця 4.2

Результати функціонального тестування модулів системи

№	Модуль	Тестовий сценарій	Очікуваний результат	Результат
1	Реєстрація	Створення нового користувача з унікальним логіном	Створено обліковий запис	✓
2	Реєстрація	Спроба повторної реєстрації з існуючим логіном	Повідомлення про помилку	✓
3	Авторизація	Вхід із правильними обліковими даними	Перенаправлення до профілю	✓
4	Авторизація	Вхід із хибним паролем	Відмова у доступі	✓
5	Додавання рецепта	Додавання валідного рецепта	Рецепт збережено та виведено в каталозі	✓

Додатково було перевірено відповідність виводу контенту очікуваній структурі, правильність фільтрації рецептів за дієтичними властивостями та відповідність доступу до функцій авторизованим користувачам. Усі перевірені дії виконувались у браузерному середовищі Google Chrome та Mozilla Firefox, на екранах різної роздільної здатності.

Таблиця 4.3 демонструє результати перевірки фільтрації та пошуку даних у системі.

Таблиця 4.3

Тестування пошуку та фільтрації рецептів

№	Тип взаємодії	Вхідні дані	Очікуваний результат	Результат
1	Пошук за ключовим словом	"Крем-суп"	Вивід рецепта з назвою, що містить це слово	✓
2	Пошук з пустим запитом	""	Вивід усього каталогу рецептів	✓
3	Фільтрація (без глютену)	Профіль із тегом "безглютенове"	Вивід лише відповідних страв	✓
4	Фільтрація (веганське)	Профіль з тегом "веганське"	Приховання страв з м’ясом	✓
5	Некоректна фільтрація	Невідоме обмеження	Вивід повідомлення або порожній результат	✓

У межах тестування також була проведена перевірка інклюзивних властивостей інтерфейсу. Для цього використовувалися онлайн-інструменти Lighthouse (Google Chrome DevTools) та WAVE (Web Accessibility Evaluation Tool), які дозволяють оцінити відповідність інтерфейсу критеріям WCAG 2.1.

Основні аспекти оцінки включали: – наявність текстових альтернатив (alt) для зображень; – підтримка клавіатурної навігації; – коректна структура заголовків; – контрастність кольорів; – коректне використання ARIA-атрибутів.

Результати наведено в таблиці 4.4.

Таблиця 4.4

Результати перевірки доступності інтерфейсу

№	Перевірений параметр	Інструмент перевірки	Стан відповідності	Оцінка
1	Контрастність тексту	Lighthouse	Відповідає (рівень AA)	✓
2	Alt-тексти для зображень	WAVE	Усі зображення мають опис	✓
3	Клавіатурна доступність	Lighthouse	Фокусування реалізовано правильно	✓

4	Структура заголовків	WAVE	Дотримано логіки H1–H3	✓

Верифікація результатів підтвердила відповідність реалізованої системи як функціональним, так і нефункціональним вимогам. Виявлені незначні зауваження були усунуті в процесі тестування. Система може вважатися придатною до використання цільовою аудиторією, включаючи осіб з особливими харчовими потребами та користувачів із порушеннями сприйняття.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи було реалізовано повний цикл розробки інформаційної системи підтримки прийняття рішень при складанні меню здорового харчування, що дозволило досягти поставленої мети — створення функціонального інструменту для автоматизованого формування персоналізованих раціонів з урахуванням дієтичних обмежень, алергенів і потреб користувачів.

На першому етапі було проведено системний аналіз предметної області, в ході якого визначено ключові сутності, типові сценарії використання, функціональні обмеження та очікування кінцевих користувачів. Аналіз сучасних вебсервісів показав, що жодна з існуючих систем не забезпечує одночасно глибокої персоналізації, підтримки медичних обмежень та мовної адаптації, що обґрунтовує актуальність запропонованого рішення.

У результаті побудовано концептуальну UML-модель, що охоплює основні аспекти логіки системи: діаграми прецедентів, класів, послідовностей, компонентів і активностей. Це дозволило формалізувати структуру даних, взаємозв’язки між компонентами та динаміку взаємодії в системі.

На основі концептуальної моделі було сформовано логічну структуру інформаційної бази, що охоплює таблиці користувачів, профілів, алергенів, інгредієнтів, рецептів, коментарів і зв’язкові таблиці. Альтернативно реалізовано зберігання даних у форматі JSON для спрощення розгортання у локальному середовищі без залежності від зовнішньої СУБД.

Реалізацію прикладного програмного забезпечення здійснено з використанням стеку JavaScript (Node.js + Express) для серверної частини та HTML/JS для користувацького інтерфейсу. Забезпечено логіку авторизації, створення профілю, фільтрації рецептів за обмеженнями, генерації меню та формування PDF-звітів.

Алгоритм генерації збалансованого меню враховує тип дієти, список алергенів і доступні рецепти, виключаючи несумісні інгредієнти та формуючи вивід у вигляді добового плану з підрахунком калорій і нутрієнтів. Всі дії алгоритмізовано та описано у вигляді блок-схем, що дозволяє однозначно відтворити логіку реалізації.

У процесі функціонального тестування підтверджено відповідність результатів заданим обмеженням, коректність фільтрації та стабільність роботи в автономному режимі. Інтерфейс системи визнано зручним, інтуїтивно зрозумілим, придатним для використання як у побутовому, так і в освітньому середовищі.

Результати ати дослідження свідчать про ефективність запропонованого підходу до розв’язання задачі підтримки здорового харчування, а розроблена система є цілісним, гнучким і придатним до масштабування програмним продуктом, який може бути використаний як кінцевими споживачами, так і у сфері охорони здоров’я та нутріціології.

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про інформацію» від 2 жовт. 1992 р. № 2657-XII [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2657-12#Text

2. Закон України «Про основні засади забезпечення біобезпеки та біозахисту» від 31 трав. 2007 р. № 2436-V [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2436-17

3. Бойко, В. М. Інформаційні системи і технології : підручник / В. М. Бойко. – Київ : Кондор, 2019. – 480 с.

4. Верещака, О. А. Технології прийняття рішень : навч. посіб. / О. А. Верещака. – Київ : Центр учбової літератури, 2018. – 264 с.

5. Мельник, С. В. Розробка інформаційних систем : навч. посіб. / С. В. Мельник, І. В. Коваленко. – Харків : ХНЕУ ім. С. Кузнеця, 2020. – 310 с.

6. Клінова, О. В. Основи раціонального харчування : навч. посіб. / О. В. Клінова. – Київ : МАУП, 2020. – 216 с.

7. Codex Alimentarius: Guidelines on Nutrition Labelling. – FAO/WHO, 2020 [Електронний ресурс]. – Режим доступу: https://www.fao.org/fao-who-codexalimentarius

8. Сухарева, Л. М. Алгоритмізація та програмування систем підтримки прийняття рішень / Л. М. Сухарева // Вісник ХНУРЕ. – 2021. – № 3. – С. 76–82.

9. Обуховський, В. В. Інформаційні технології в медицині та біології / В. В. Обуховський. – Львів : Вид-во ЛНУ ім. Івана Франка, 2022. – 234 с.

10. National Institute of Health. Healthy Eating Index [Електронний ресурс]. – Режим доступу: https://www.nutrition.gov/topics/healthy-living-and-eating

11. Власенко, І. А. Харчові добавки та їх вплив на здоров’я людини / І. А. Власенко // Проблеми харчування. – 2022. – Т. 12, № 1. – С. 45–50.

12. Ребрик, Ю. М. Системи підтримки прийняття рішень : навч. посіб. / Ю. М. Ребрик. – Чернівці : Рута, 2021. – 280 с.

13. Столяр, О. В. Інформаційні технології в галузі харчування / О. В. Столяр // Науковий вісник ХДАУ. – 2021. – № 4(2). – С. 105–111.

14. Python Software Foundation. Python 3.10 Documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/

15. AllRecipes. Homepage. 2024. URL: https://www.allrecipes.com

16 KitchenAid / Yummly. Smart Cooking and Recipes Platform. 2024. URL: https://www.kitchenaid.com

17 BBC Good Food. Healthy recipes. 2024. URL: https://www.bbcgoodfood.com

18 Eat This Much. Personal Meal Planner. 2024. URL: https://www.eatthismuch.com