Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка вебзастосунку для раціонального пошуку кулінарних рецептів

Виконав: студент IV курсу, групи СН-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Юсюк О. М.
(підпис) (прізвище та ініціали)

Керівник _____ Готович В. А.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Липак Г.І.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Мудрик І.Я.
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _червня_ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Юсюку Олегу Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебзастосунку для раціонального пошуку кулінарних рецептів

Керівник роботи Готович Володимир Анатолійович, к.т.н., доц., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*14*» *травня* 2026 року № *4/9-239*

2. Термін подання студентом завершеної роботи 23 червня 2026 р.

3. Вихідні дані до роботи Літературні та інтернет джерела по темі роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та постановка завдання. 1.1 Огляд існуючих рішень для пошуку рецептів. 1.2 Обґрунтування вибору технологій та засобів розробки вебзастосунку для раціонального пошуку кулінарих рецептів. 1.3 Постановка завдання для розробки вебзастосунку для раціонального пошуку кулінарних рецептів. 1.4 Висновки до першого розділу. 2. Проєктування та розробка вебзастосунку для раціонального пошуку кулінарних рецептів. 2.1 Розробка архітектури вебзастосунку для раціонального пошуку кулінарних рецептів. 2.2 Пошук актантів та варіантів використання. 2.3 Розробка користувацької частини вебзастосунку для раціонального пошуку кулінарних рецептів. 2.4 Розробка серверної частини вебзастосунку для раціонального пошуку кулінарних рецептів. 2.5 Програмна реалізація логіки вебзастосунку для раціонального пошуку кулінарних рецептів. 2.6 Висновки до другого розділу. 3. Демонстрація та тестування вебзастосунку для раціонального пошуку кулінарних рецептів. 3.1 Демонстрація функціоналу вебзастосунку для раціонального пошуку кулінарних рецептів. 3.2 Типові сценарії роботи користувача з вебзастосунком для раціонального пошуку кулінарних рецептів. 3.3 Тестування вебзастосунку для раціонального пошуку кулінарних рецептів. 3.4 Висновки до третього розділу. 4. Безпека життєдіяльності, основи охорони праці. 4.1 Психологічні чинники. небезпеки. 4.2 Психофізіологічне розвантаження для працівників. Висновки. Перелік джерел. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О.Я., к.т.н., доцент		

7. Дата видачі завдання 07 травня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	07.05.2026	*Виконано*
2.	Підбір та опрацювання літературних джерел по	08.05.2026-14.05.2026	*Виконано*
	розробці вебзастосунків для раціонального пошуку		
	кулінарних рецептів.		*Виконано*
3.	Виконання дослідження щодо вибору програмних	15.05.2026-17.05.2026	
	засобів.		
4.	Розроблення вебзастосунку для раціонального	18.05.2026-25.05.2026	
	пошуку кулінарних рецептів.		
5.	Оформлення розділу «Аналіз предметної області та	26.05.2026-29.05.2026	*Виконано*
	Постановка завдання»		
6.	Оформлення розділу «Проєктування та розробка веб-	30.05.2026-04.06.2026	*Виконано*
	застосунку для раціонального пошуку кулінарних		
	рецептів»		
7.	Оформлення розділу «Демонстрація та тестування	05.06.2026-10.06.2026	*Виконано*
	вебзастосунку для раціонального пошуку кулінарних		
	рецептів»		
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»	11.06.2026-12.06.2026	*Виконано*
9.	Виконання завдання до підрозділу «Основи охорони	13.06.2026-14.06.2026	*Виконано*
	праці»		
	Оформлення кваліфікаційної роботи	15.06.2026-19.06.2026	*Виконано*
	Нормоконтроль	20.06.2026-21.06.2026	*Виконано*
	Перевірка на плагіат	22.06.2026	*Виконано*
	Попередній захист кваліфікаційної роботи	25.06.2026	*Виконано*
	Захист кваліфікаційної роботи	26.06.2026	

Студент _____ Юсюк О.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Готович В.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка веб-застосунку для раціонального пошуку кулінарних рецептів // Кваліфікаційна робота освітнього рівня «Бакалавр» // Юсюк Олег Миколайович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-42 // Тернопіль, 2026 // С. 59, рис. – 19, табл. – 2, кресл. – , додат. – 2, бібліогр. – 50.

Ключові слова: вебзастосунок, кулінарні рецепти, раціональний пошук, облік інгредієнтів, алгоритм пошуку, Laravel, SPA.

Кваліфікаційна робота присвячена розробці вебзастосунку для раціонального пошуку кулінарних рецептів.

В першому розділі кваліфікаційної роботи проведено аналіз предметної області, розглянуто існуючі аналоги, визначено їх переваги та недоліки, сформульовано вимоги до розроблюваного веб-застосунку, а також проаналізовано вибір інструментарію для розробки.

У другому розділі кваліфікаційної роботи виконано проєктування архітектури веб-застосунку, яка базується на принципах SPA та REST API. Розроблено інформаційну модель системи та реляційну базу даних для зберігання облікових записів та списків продуктів. Створено діаграми варіантів використання та описано програмну реалізацію основних модулів веб-застосунку.

У третьому розділі кваліфікаційної роботи продемонстровано реалізацію розробленого вебзастосунку для раціонального пошуку кулінарних рецептів та проведено його тестування. Проаналізовано зручність графічного інтерфейсу користувача та коректність роботи веб-застосунку.

Об'єкт дослідження: процес управління запасами харчових продуктів та автоматизованого підбору кулінарних рецептів.

Предмет дослідження: методи, алгоритми та програмні засоби розробки клієнт-серверного веб-застосунку для раціонального використання продуктів.

ANNOTATION

Development of a Web Application for Rational Culinary Recipe Search // Qualification work of the educational level «Bachelor» // Yusiuk Oleg // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-42 // Ternopil, 2026 // P. 59, fig. – 19, tabl. – 2, chair. – , annexes. – 2, references – 50.

Keywords: web application, cooking recipes, rational search, ingredient accounting, search algorithm, Laravel, SPA.

The qualification work is dedicated to the development of a web application for rational search of culinary recipes.

The first chapter of the qualification work provides an analysis of the subject area, considers existing analogues, identifies their advantages and disadvantages, formulates the requirements for the developed web application, and analyzes the choice of development tools.

In the second section of the qualification work, the web application architecture was designed, which is based on the principles of SPA and REST API. The information model of the system and a relational database for storing accounts and product lists were developed. Use case diagrams were created and the software implementation of the main modules of the web application was described.

In the third section of the qualification work, the implementation of the developed web application for rational recipe search is demonstrated and its testing is carried out. The convenience of the graphical user interface and the correctness of the web application are analyzed.

Object of research: the process of managing food stocks and automated selection of culinary recipes.

Subject of research: methods, algorithms and software tools for developing a client-server web application for the rational use of products.

ПЕРЕЛІК СКОРОЧЕНЬ

API (англ. Application Programming Interface) – Прикладний програмний інтерфейс.

CSS (англ. Cascading Style Sheets) – Каскадні таблиці стилів.

DOM (англ. Document Object Model) – Об’єктна модель документа.

HTTP (англ. HyperText Transfer Protocol) – Протокол передачі гіпертексту.

JSON (англ. JavaScript Object Notation) – Текстовий формат обміну даними.

MVC (англ. Model-View-Controller) – Модель-Вигляд-Контролер.

ORM (англ. Object-Relational Mapping) – Об’єктно-реляційне відображення.

OWASP (англ. Open Worldwide Application Security Project) – Відкритий проєкт безпеки веб-застосунків.

PDO (англ. PHP Data Objects) – Об’єкти даних PHP.

REST (англ. Single Page Application) – Передача стану представлення.

SPA (англ. HyperText Transfer Protocol) – Односторінковий веб-застосунок.

SQL (англ. Structured Query Language) – Мова структурованих запитів.

БД – База даних.

СУБД – Система управління базами даних.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі питання раціонального споживання та економії ресурсів стає все більш гострим. В умовах економічної нестабільності та зростання цін на продовольчі товари, домогосподарства стикаються з необхідністю оптимізації сімейного бюджету. Водночас, парадоксальною залишається проблема нераціонального використання куплених продуктів.

Згідно зі звітом ООН про індекс харчових відходів (UNEP Food Waste Index Report 2024), саме домогосподарства генерують значну частину харчових відходів, викидаючи понад 1 мільярд страв щодня по всьому світу [1]. Часто це стається через невміння планувати меню або відсутність ідей, що приготувати з наявних залишків.

Одним із рішень цієї проблеми є використання сучасних інформаційних технологій. Перевагою веб-застосунків є доступ з будь-якого пристрою через браузер. Це дозволить користувачам швидко знаходити кулінарні рішення на основі наявних інгредієнтів. Створення такого інструменту дозволить користувачам не лише спробувати нові рецепти, а й суттєво зекономити час на планування та кошти на закупівлі.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи є розробка вебзастосунку для раціонального пошуку кулінарних рецептів, який дозволяє формувати меню на основі наявних у користувача харчових продуктів.

Для досягнення поставленої мети потрібно виконати ряд **задач**, зокрема:

- Сформулювати постановку задачі кваліфікаційної роботи.
- Дослідити існуючі аналоги до розроблюваного веб-застосунку для підбору рецептів та проаналізувати їхні переваги і недоліки.
- Обґрунтувати вибір інструментальних засобів та методів для веб-розробки.
- Визначити варіанти використання та вимоги до системи.
- Розробити архітектуру веб-застосунку.

- Реалізувати основні функції інформаційної системи веб-застосунку: базу даних продуктів та рецептів, алгоритм пошуку страв, особистий кабінет користувача.
- Провести тестування розробленого веб-застосунку, перевірити коректність роботи та виправити помилки у разі виявлення.

Об’єкт дослідження: процес управління запасами харчових продуктів та автоматизованого підбору кулінарних рецептів.

Предмет дослідження: методи, алгоритми та програмні засоби розробки клієнт-серверного веб-застосунку для раціонального використання продуктів.

Практичне значення одержаних результатів. Розроблений веб-застосунок для раціонального пошуку кулінарних рецептів має важливе практичне значення, оскільки надає користувачам інструмент для вирішення щоденної побутової задачі коли вони не знають що приготувати. Впровадження такого веб-застосунку дозволяє економити час на пошук інформації, зменшити кількість імпульсивних покупок та мінімізувати псування продуктів, що в результаті сприяє більш раціональному веденню домашнього господарства.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Огляд відомих програмних рішень для пошуку кулінарних рецептів

На сьогоднішній день ринок технологій у сфері харчування пропонує широкий спектр веб-ресурсів та мобільних застосунків, які спрямовані на допомогу користувачам у приготуванні їжі. Для обгрунтування доцільності розробки власного вебзастосунку для раціонального пошуку кулінарних рецептів, необхідно спочатку проаналізувати найпопулярніші рішення, виявити їхні функціональні можливості та недоліки.

Серед світових лідерів у даній предметній області можна виділити такі платформи, як SuperCook, Allrecipes та KitchenAid.

SuperCook є одним з найбільш відомих спеціалізованих сервісів, який фокусується на пошуку рецептів за наявними інгредієнтами. Основна ідея даної платформи полягає в тому, що користувач обирає зі списку продукти які в нього є, а вже за обраними користувачем продуктами сервіс шукає рецепт, який він може приготувати, фільтруючи базу даних і залишаючи тільки ті рецепти які підходять, такий спосіб дозволяє приготувати щось смачне без додаткового походу в магазин [2].

Перевагами цього сервісу є: чітка спеціалізація на пошуку за інгредієнтами, велика база даних рецептів та можливість вказати дієтичні обмеження на випадок якщо користувач веган або вегетаріанець наприклад. Недоліками SuperCook є те, що в ньому відсутня функція збереження стану запасів, тому платформа не буде підлаштовуватись якщо наприклад, в користувача якийсь продукт зіпсується, крім того інтерфейс платформи доволі перенавантажений, що може відлякувати деяких користувачів. Також сервіс не має української локалізації, що може бути проблемою для українців які не знають англійської. Інтерфейс SuperCook можна побачити на рисунку 1.1.

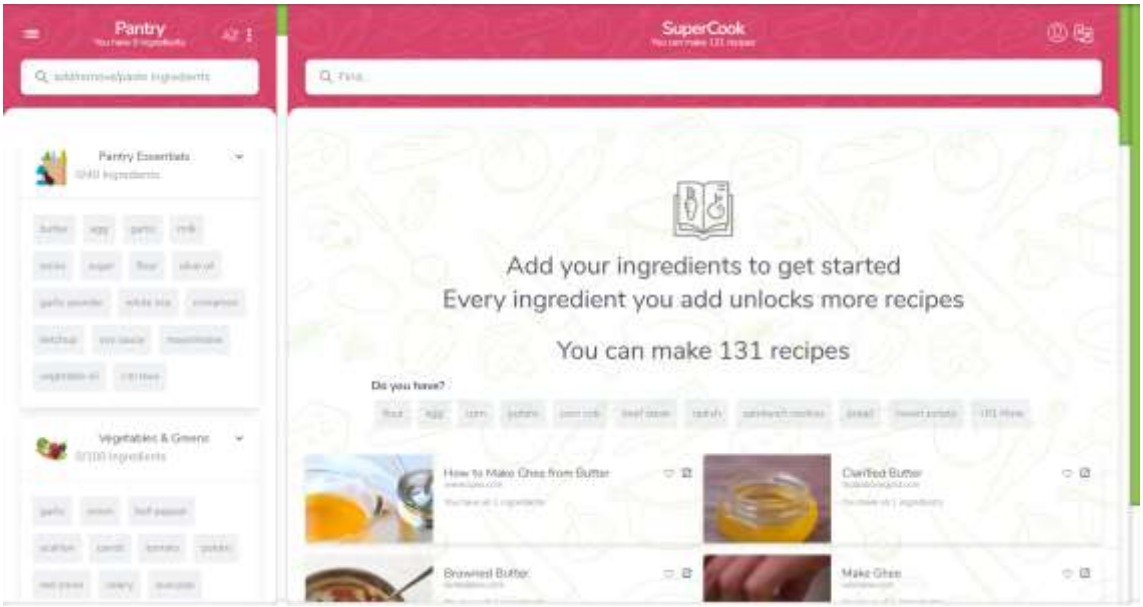

Рисунок 1.1 – Інтерфейс SuperCook

Allrecipes – це одна з найстаріших платформ, яка функціонує як соціальна мережа для обміну рецептами. Основний акцент зроблено на спільноті: платформу підтримують користувачі, які самі додають рецепти, залишають відгуки та фотографії готових страв [3].

Перевагами платформи є величезна база перевірених користувачами рецептів та наявність рейтингів і коментарів, це дозволяє оцінити якість страви до початку приготування.

Проте платформа має значні недоліки, одним з яких є традиційний пошук: користувачі шукають страву за її назвою, а не за інгредієнтами які в них є. Пошук за інгредієнтами тут можливий, але реалізований як додатковий фільтр і працює менш ефективно. Іншим недоліком є суб’єктивність даних: оскільки контент створюють самі користувачі, опис рецептів та фотографії страв можуть бути некоректними, або їхня якість може суттєво різнитись. Також сайт локалізований тільки англійською мовою, та не має інших перекладів. Інтерфейс платформи зображено на рисунку 1.2.

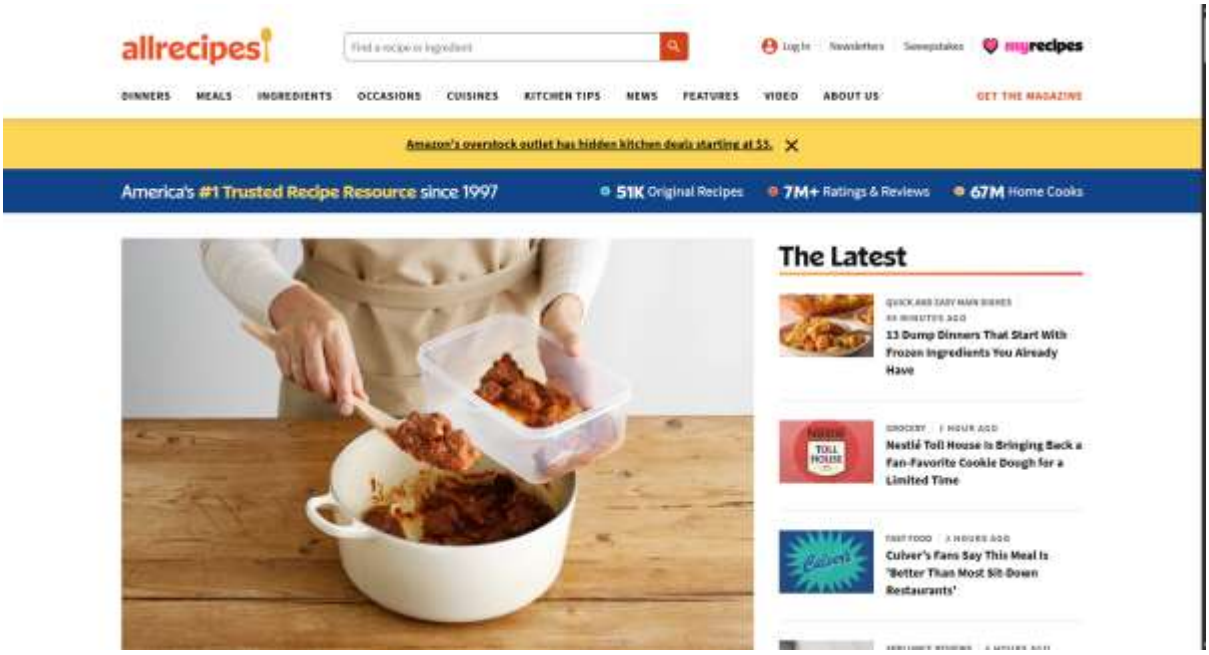

Рисунок 1.2 – Інтерфейс Allrecipes

Ще однією альтернативою є інтелектуальна платформа KitchenAid. Вона позиціонує себе як “розумний” помічник, що використовує алгоритми машинного навчання для персоналізації видачі. Система аналізує вподобання користувача в залежності від того, які рецепти він вподобав, та пропонує схожі варіанти [4].

Перевагами платформи є високий рівень персоналізації та якісний візуальний контент. Також для користувачів із США платформа має інтеграцію зі службами доставки продуктів.

Недоліками платформи є комерційна спрямованість: основною метою платформи є продаж супутніх товарів або платних підписок, а не допомога в економії продуків. Також слід відзначити високу складність користування: для простого пошуку (наприклад, “що приготувати з яєць”) користувач повинен пройти довгу процедуру налаштування профілю. Крім того сервіс як і попередні аналоги, не має української локалізації. Інтерфейс платформи KitchenAid є на рисунку 1.3

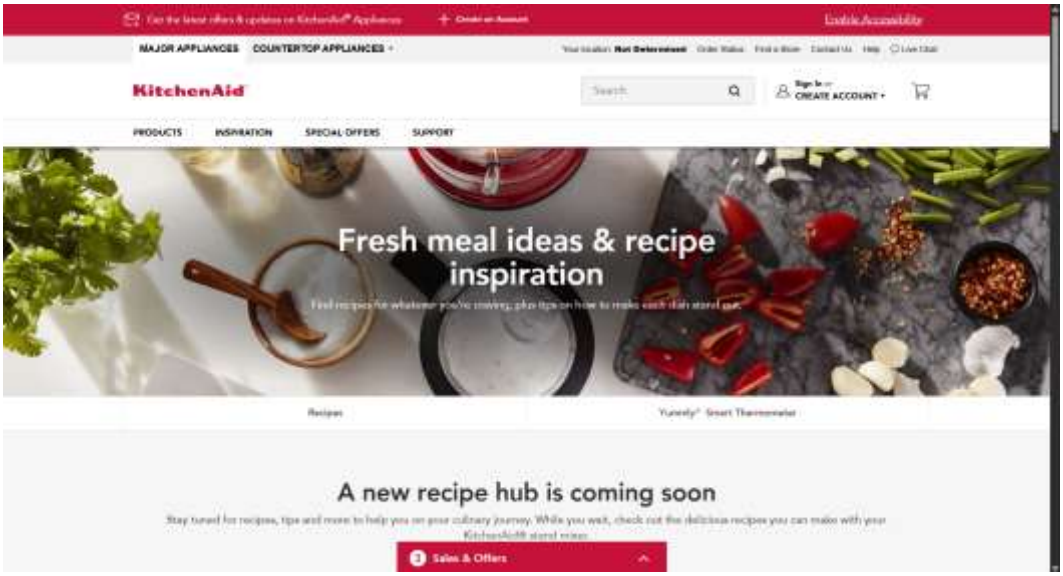

Рисунок 1.3 – Інтерфейс KitchenAid

Для наочності порівняємо переваги та недоліки розглянутих аналогів. Результати порівняння винесено в таблицю 1.1.

Таблиця 1.1. – Порівняння переваг та недоліків відомих програмних рішень

Критерій	**SuperCook**	**Allrecipes**	**KitchenAid**
Пошук за інгредієнтами	Є (Основна)	Є (Додаткова)	Є (Додаткова)
Облік запасів	Ні	Ні	Ні
Моніторинг термінів придатності	Ні	Ні	Ні
Персоналізація видачі	Є	Є	Висока
Українська Локалізація	Часткова	Ні	Ні
Складність інтерфейсу	Висока	Середня	Висока

Проведений аналіз показав, що існуючі рішення здебільшого орієнтовані на прямий пошук рецептів або продаж продуктів. На ринку відсутній (або слабо

представлений) інструмент, який би поєднував пошук рецептів із функцією інвентаризації домашніх запасів. Більшість сервісів не вирішують проблему моніторингу термінів придатності, що є ключовим фактором для раціонального споживання. Таким чином, розробка веб-застосунку, який закриває цю нішу, є актуальною та доцільною.

1.2 Обґрунтування вибору технологій та засобів розробки веб-застосунку

Вибір технологічного стеку є критичним етапом проєктування програмного забезпечення, оскільки він визначає швидкодію, масштабованість, безпеку та зручність подальшої підтримки веб-застосунку. Виходячи з вимог до системи (клієнт-серверна архітектура, інтерактивний інтерфейс, робота з реляційними даними), було проведено аналіз сучасних засобів веб-розробки.

Для реалізації серверної частини розглядалися мови програмування PHP, Python та Node.js. Враховуючи специфіку веб-орієнтованих систем, було обрано мову PHP через її широку підтримку хостинг-провайдерами та спеціалізацію саме на веб-задачах.

Однак, використання «чистого» PHP у сучасній розробці є нераціональним через низьку швидкість написання коду та проблеми з безпекою. Тому було прийнято рішення використовувати фреймворк. Серед найпопулярніших рішень було обрано Laravel.

Основні причини вибору Laravel:

1. Архітектура MVC: Забезпечує чітке розділення логіки обробки даних, інтерфейсу та управління запитами [5].

2. ORM Eloquent: Вбудована система об’єктно-реляційного відображення дозволяє працювати з базою даних, використовуючи об’єкти PHP, а не громіздкі SQL-запити, що підвищує читабельність коду та захищає від SQL-ін’єкцій.

3. Вбудована система аутентифікації: Фреймворк надає готові інструменти для реєстрації та захисту маршрутів (API routes), що є критичним для створення особистого кабінету користувача.

Для створення сучасного інтерфейсу було відкинуто застарілий підхід із перезавантаженням сторінок на користь архітектури SPA. Для цього необхідне використання JavaScript-бібліотеки або фреймворку. Для розробки обрано бібліотеку React.js від компанії Meta.

Переваги React.js для розроблюваного веб-застосунку:

1. Віртуальний DOM: React створює легку копію DOM-дерева в пам’яті. При зміні даних React оновлює лише змінену частину сторінки, а не перезавантажує її повністю. Це забезпечує високу швидкодію інтерфейсу.

2. Компонентний підхід: Інтерфейс розбивається на незалежні компоненти, які можна повторно використовувати в різних частинах застосунку [6].

3. Екосистема: Величезна кількість готових бібліотек спрощує інтеграцію з серверною частиною.

Проєктована система оперує чітко структурованими даними, які мають зв’язки один до багатьох та багато до багатьох. Через наявність жорстких зв’язків використання NoSQL баз даних є недоцільним. Тому було обрано реляційну модель. Серед реляційних СУБД обрано MySQL, через її простоту адміністрування та підтримку зовнішніх ключів.

У результаті аналізу було сформовано наступний стек технологій для реалізації веб-застосунку:

- Backend: PHP (фреймворк Laravel) – для обробки бізнес-логіки та надання REST API.
- Frontend: JavaScript (бібліотека React.js) – для створення динамічного інтерфейсу користувача.
- СКБД: MySQL – для надійного зберігання структурованих даних.

Такий вибір забезпечує баланс між швидкістю розробки, продуктивністю системи та сучасними стандартами веб-розробки.

1.3 Постановка завдання для розробки вебзастосунку для раціонального пошуку кулінарних рецептів

На основі проведеного аналізу предметної області та існуючих аналогів, було сформульовано основну мету роботи – створення вебзастосунку для раціонального пошуку кулінарних рецептів, який дозволяє користувачам ефективно керувати власними продуктовими запасами.

Для досягнення цієї мети система повинна відповідати ряду функціональних та нефункціональних вимог.

Функціональні вимоги системи були згруповані за модулями: модуль авторизації та керування профілем, модуль управління запасами, модуль пошуку та підбору рецептів, модуль статистики.

Модуль авторизації та керування профілем відповідає за реєстрацію нових користувачів із валідацією введених даних, аутентифікацію користувачів за допомогою логіну та паролю, захист персональних даних шляхом хешування паролів перед збереженням у базі даних, можливість виходу із системи.

Модуль управління запасами потрібен для додавання продуктів до особистого списку користувача із зазначенням назви, кількості, одиниць виміру та терміну придатності, редагування параметрів продуктів, видалення продуктів зі списку, відображення списку наявних продуктів із індикацією свіжості.

Модуль пошуку та підбору рецептів відповідає за формування пошукового запиту на основі обраних користувачем продуктів, взаємодію із зовнішнім API кулінарних рецептів, фільтрацію отриманих результатів та відображення детальної інформації про обраний рецепт.

Модуль статистики є додатковим, він потрібен для ведення історії використаних та викинутих продуктів, що дозволить користувачам знати скільки продуктів вони зекономили завдяки веб-застосунку.

Нефункціональні вимоги визначають якісні атрибути системи, такі як продуктивність, безпека та зручність використання. До них відносяться такі вимоги як:

- Архітектура – система повинна бути побудована за архітектурою клієнт-сервер, де клієнтська частина взаємодіє із серверною через REST API.
- Продуктивність – час відповіді сервера на запит користувача не повинен перевищувати 2-3 секунди при стабільному інтернет-з'єднанні.
- Інтерфейс – графічний інтерфейс користувача повинен бути інтуїтивно зрозумілим, мати сучасний дизайн та адаптивну верстку для коректного відображення на різних типах пристроїв.
- Обробка помилок – система повинна коректно обробляти помилки та повідомляти про це користувача, не припиняючи роботу застосунку.
- Масштабованість – структура бази даних повинна дозволяти легке додавання нових сутностей без необхідності повної переробки системи.

Таким чином, визначені функціональні та нефункціональні вимоги формують основу технічного завдання на розробку вебзастосунку для раціонального пошуку кулінарних рецептів. Реалізація сформульованих вимог забезпечить створення конкурентоспроможного програмного продукту, який вирішує актуальну проблему раціонального використання продовольчих запасів.

1.4 Висновки до першого розділу

В першому розділі кваліфікаційної роботи проведено аналіз предметної області та обґрунтовано доцільність розробки вебзастосунку для раціонального пошуку кулінарних рецептів. Дослідження існуючих аналогів показало відсутність на ринку зручних систем, які б поєднували інтелектуальний пошук страв із повноцінним управлінням домашніми продуктовими запасами. Для розв'язання цієї проблеми та забезпечення високої швидкодії системи обґрунтовано вибір сучасного стеку технологій. На основі проведеного аналізу сформульовано постановку задачі та визначено ключові функціональні та нефункціональні вимоги до розроблюваного вебзастосунку для раціонального пошуку кулінарних рецептів.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ РАЦІОНАЛЬНОГО ПОШУКУ КУЛІНАРНИХ РЕЦЕПТІВ

2.1 Розробка архітектури вебзастосунку для раціонального пошуку кулінарних рецептів

Основою для побудови надійної, масштабованої та зручної в підтримці інформаційної системи є правильний вибір програмної архітектури. Для розробки вебзастосунку раціонального пошуку кулінарних рецептів було обрано клієнт-серверну архітектуру з використанням підходу SPA та взаємодією компонентів через RESTful API [7].

Такий архітектурний патерн передбачає чітке розділення системи на дві незалежні частини: клієнтську та серверну, які обмінюються даними у форматі JSON [8]. Загальна архітектура спроєктованої системи складається з наступних ключових рівнів:

- Клієнтська частина, яка реалізована за допомогою бібліотеки React.js. Ця частина відповідає за візуалізацію даних та безпосередню взаємодію з користувачем. Оскільки система побудована як SPA, при переході між розділами сторінка не перезавантажується повністю. Замість цього клієнтський застосунок асинхронно відправляє HTTP-запити на сервер, отримує необхідні дані та динамічно оновлює лише відповідні компоненти інтерфейсу за допомогою технології Virtual DOM [9]. Це забезпечує високу швидкість відгуку системи, наближену до десктопних програм.
- Серверна частина, яка відповідає за бізнес-логіку. Реалізована за допомогою PHP-фреймворку Laravel, який використовує архітектурний шаблон MVC [10]. Backend виконує роль прошарку між клієнтом, базою даних та зовнішніми сервісами. Його основними завданнями є: маршрутизація, аутентифікація користувача, обробка бізнес-правил та інтеграція із зовнішнім API. Серверна частина формує запити до стороннього кулінарного сервісу,

отримує відповідь, фільтрує її та передає стандартизований результат на клієнтську частину [11].

- Зберігання даних, яке реалізоване за допомогою реляційної СУБД MySQL. Взаємодія серверної частини з базою даних відбувається не через прямі SQL-запити, а за допомогою вбудованої в Laravel системи об’єктно-реляційного відображення Eloquent [12]. Це дозволяє працювати з таблицями бази даних як з об’єктами у PHP-коді, що значно підвищує рівень безпеки та спрощує підтримку коду.

Така архітектура вебзастосунку для раціонального пошуку кулінарних рецептів є гнучкою, масштабованою та дозволяє в майбутньому розширювати функціонал системи шляхом оновлень, що збільшить її життєвий цикл. Візуалізацію архітектури вебзастосунку для раціонального пошуку кулінарних рецептів, яку було створено через сервіс Mermaid [30], можна побачити на рисунку 2.1

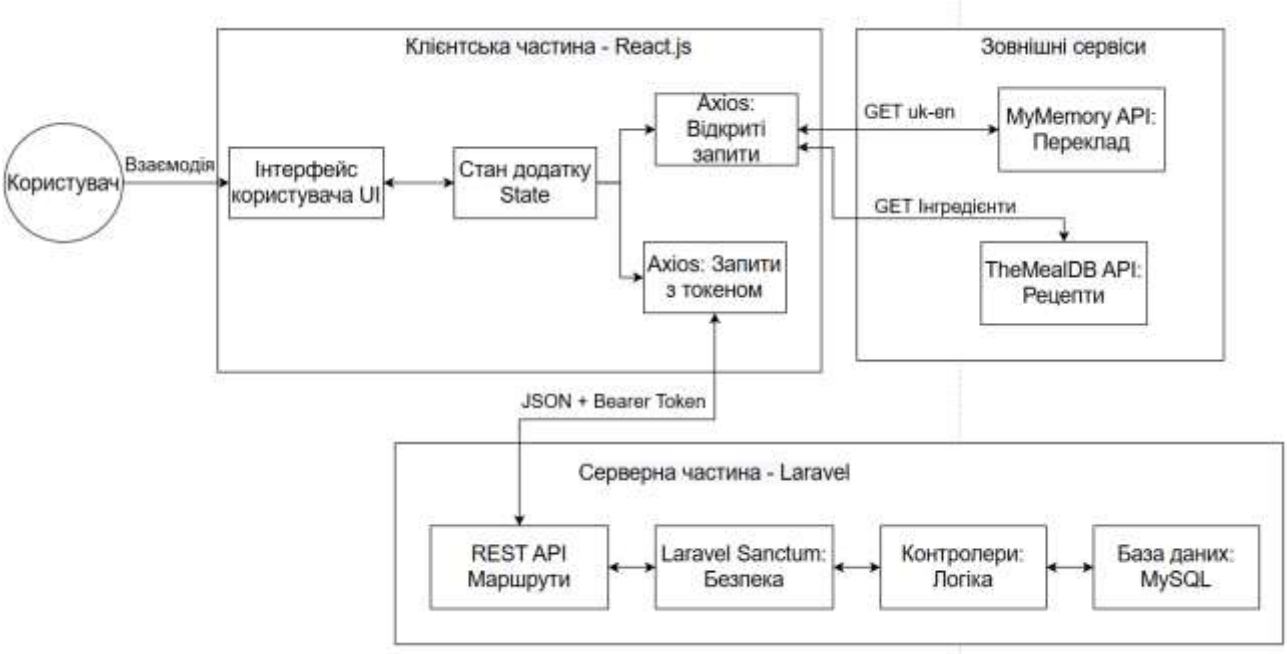

Рисунок 2.1 – Архітектура веб-застосунку

Для зручності розробки та подальшого супроводу, файлову структуру проєкту було організовано згідно з принципами модульності та стандартів

обраних фреймворків. Файлова система серверної частини суворо наслідує патерн MVC: виділено окремі директорії для моделей бази даних, контролерів бізнес-логіки, конфігураційних файлів та міграцій. Клієнтська частина має класичну структуру SPA-додатку, де основний вихідний код (компоненти інтерфейсу, логіка стану та стилі) консолідовано в єдиній робочій директорії, а управління залежностями здійснюється через пакетний менеджер. Такий підхід дозволяє уникнути надмірної зв’язності коду.

Для забезпечення взаємодії між сутностями системи було спроєктовано логічну структуру бази даних та відповідних класів моделей. Оскільки в основі серверної частини лежить ORM Eloquent, кожна таблиця в базі даних має відповідний клас-модель, який інкапсулює логіку роботи з даними.

UML-діаграму класів спроєктованої системи наведено на рисунку 2.2.

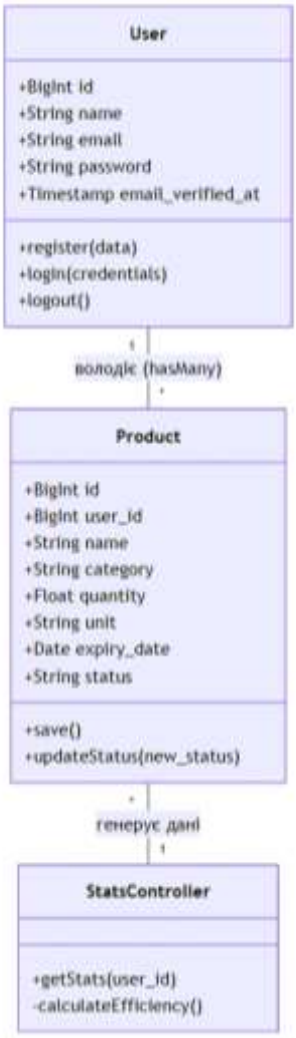

Рисунок 2.2 – UML-діаграма класів системи

Головними класами розробленої системи є User (Користувач) та Product (Продукт). Зв’язок між ними реалізовано за типом реляційного відношення «один-до-багатьох» (один користувач може мати багато продуктів у своєму віртуальному сховищі, але кожен конкретний запис про продукт належить лише одному користувачу). За розрахунок ефективності та обробку статусів продуктів відповідає додатковий програмний контролер StatsController.

Використання об’єктно-орієнтованого підходу та інкапсуляція бази даних у класи моделей дозволяє захистити систему від прямих SQL-ін'єкцій та робить програмний код більш структурованим і придатним для подальшого масштабування.

2.2 Пошук актантів та варіантів використання

Дуже важливим етапом проєктування програмного забезпечення є створення діаграми прецедентів, яка візуально зображає різноманітні сценарії взаємодії між акторами (користувачами) і прецедентами (випадками використання); описує функціональні аспекти системи (бізнес логіку) [13].

В межах дипломної роботи було проведено аналіз майбутніх взаємодій користувачів з вебзастосунком для раціонального пошуку кулінарних рецептів та сформовано перелік актантів і типових варіантів використання. Це дало змогу сформувати технічне бачення основних процесів, які розроблювана система має підтримувати.

У розроблюваному веб-застосунку виділено два типи актантів, які мають різні рівні доступу та сценарії використання:

1. Гість – це відвідувач системи, який ще не створив обліковий запис. Його взаємодія з вебзастосунком для раціонального пошуку кулінарних рецептів обмежується загальнодоступними функціями та сторінками.

2. Авторизований користувач – це користувач, який увійшов у свій обліковий запис і має повний доступ до персонального «Віртуального холодильника» та інтелектуального пошуку рецептів.

3. Зовнішня кулінарна система – це вторинний актант, який представляє собою сторонній сервіс баз даних рецептів. Цей актант не ініціює дії самостійно, але бере безпосередню участь у бізнес-логіці застосунку, відповідаючи на запити сервера та повертаючи необхідний контент у форматі JSON [14].

Після визначення актантів було визначено варіанти використання:

- Відкриття головної сторінки: вхідна точка взаємодії, що ініціює завантаження лендінгу з інформацією про раціональне споживання продуктів.
- Реєстрація у системі: гість заповнює форму реєстрації для створення нового облікового запису.
- Автентифікація: гість вводить логін і пароль для входу в свій обліковий запис.
- Керування запасами продуктів: користувачу доступне створення нових записів про продукти, редагування існуючих та видалення.
- Додати продукт: додавання нового інгредієнта із зазначенням назви, кількості та терміну придатності.
- Видалити продукт: видалення продукту з бази у разі його повного споживання або псування.
- Редагувати продукт: зміна даних про продукт.
- Пошук рецептів: авторизований користувач ініціює процес підбору страв на основі наявних продуктів. Ця дія автоматично запускає ланцюг системних прецедентів.
- Формування API-запиту: внутрішній механізм веб-застосунку компілює список продуктів користувача та відправляє запит до Зовнішньої кулінарної системи.
- Обробка запиту зовнішнім сервісом: актант «Зовнішня кулінарна система» приймає параметри, проводить фільтрацію на своєму боці та генерує відповідь.

- Передача кулінарних даних: актант «Зовнішня кулінарна система» повертає веб-застосунку масив даних для подальшого рендерингу.
- Обробка помилок API: прецедент, при якому система інформує користувача про відсутність з’єднання із зовнішнім сервісом або відсутність рецептів за заданими критеріями [15].
- Перегляд деталей рецепта: користувач відкриває обрану страву для перегляду покрокової інструкції.
- Вихід із системи: авторизований користувач завершує сесію та перетворюється в актант «Гість».

На рисунку 2.3 зображено діаграму варіантів використання веб-застосунку для раціонального пошуку кулінарних рецептів.

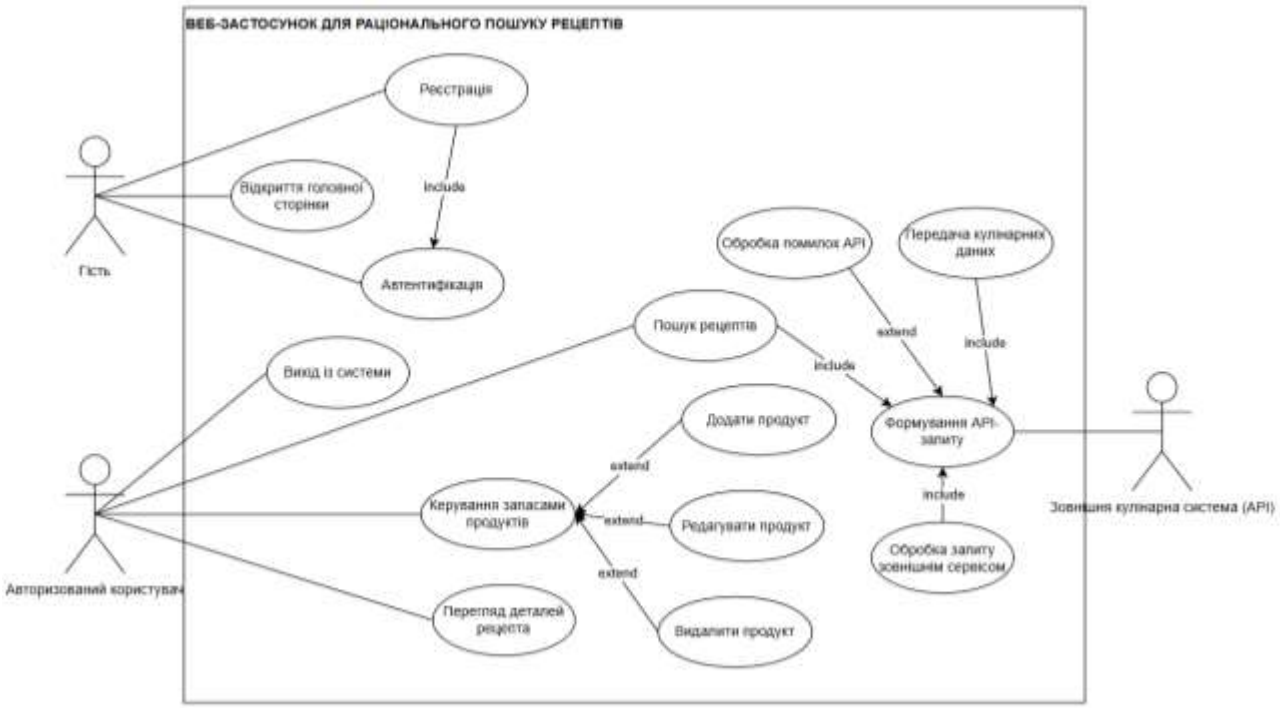

Рисунок 2.3 – Діаграма варіантів використання

Побудова даної діаграми варіантів використання дозволила чітко структурувати функціональні вимоги до системи, забезпечити повноту сценаріїв взаємодії з користувачем та слугує важливим етапом у розробці логіки вебзастосунку для раціонального пошуку кулінарних рецептів. Для

побудови діаграми варіантів використання було використано сервіс diagrams.net [16].

2.3 Розробка користувацької частини вебзастосунку для раціонального пошуку кулінарних рецептів

Важливим етапом створення сучасного програмного забезпечення є проєктування та розробка зручного користувацького інтерфейсу (UI) і забезпечення позитивного користувацького досвіду (UX). Оскільки головною метою вебзастосунку для раціонального пошуку кулінарних рецептів є допомога користувачам у швидкому підборі страв та управлінні запасами, інтерфейс має бути інтуїтивно зрозумілим, швидким та адаптивним [17]. Розробка інтерфейсу базувалася на стандартах консорціуму W3C щодо семантичної верстки та доступності веб-контенту [18].

Для реалізації клієнтської частини веб-застосунку було використано бібліотеку React.js. Цей вибір зумовлений можливістю створення односторінкових застосунків, де оновлення даних відбувається асинхронно без перезавантаження всієї сторінки, що значно покращує швидкодію системи.

В основі розробки на React.js лежить компонентний підхід. Весь інтерфейс вебзастосунку для раціонального пошуку кулінарних рецептів було розділено на незалежні, ізольовані модулі (компоненти), які можна повторно використовувати в різних частинах програми [19]. Це значно спрощує процес розробки та подальшого тестування.

Основне дерево компонентів розроблюваної системи включає наступні елементи:

1. App – кореневий компонент, який обгортає весь застосунок, ініціалізує маршрутизацію та зберігає глобальний стан автентифікації користувача.
2. Navigation – забезпечує перехід між основними розділами застосунку.

3. RefrigeratorList – один із ключових компонентів вебзастосунку для раціонального пошуку кулінарних рецептів. Відповідає за виведення списку наявних інгредієнтів. Він містить логіку візуального виділення продуктів, термін придатності яких добігає кінця, що стимулює користувача до їх раціонального використання.

4. ProductForm – компонент, що містить поля для введення назви, кількості та дати придатності нового інгредієнта з вбудованою валідацією на стороні клієнта.

5. RecipeSearch – компонент, який акумулює дані з "Віртуального холодильника" та формує параметри для відправки запиту на підбір рецептів.

6. RecipeCard – компонент для відображення короткої інформації про знайдену страву, як-от фотографія, назва, час приготування та відсоток збігу за інгредієнтами.

Оскільки застосунок функціонує як SPA, класична маршрутизація через запити до сервера не використовується. Замість цього застосовано бібліотеку react-router-dom [20]. Вона дозволяє перехоплювати зміни URL-адреси в браузері та динамічно підвантажувати відповідні React-компоненти, імітуючи багатосторінковий сайт.

Управління станом всередині компонентів реалізовано за допомогою сучасного підходу React Hooks. Зокрема, хук useState використовується для локального збереження даних форми, списку продуктів та отриманих з API рецептів, а хук useEffect, який застосовується для виконання побічних ефектів, таких як відправка асинхронних HTTP-запитів до Laravel Backend при завантаженні компонента.

Для зв’язку користувацької частини із серверною та зовнішньою кулінарною системою використовується бібліотека Axios [21]. Усі запити до сервера: GET, POST, PUT та DELETE, виконуються асинхронно у форматі JSON.

Приклад життєвого циклу взаємодії у вебзастосунку для раціонального пошуку кулінарних рецептів: коли користувач натискає кнопку "Знайти

рецепти", компонент RecipeSearch через Axios відправляє POST-запит на сервер, передаючи масив наявних продуктів. На час очікування відповіді користувачу відображається індикатор завантаження. Після отримання масиву даних від сервера стан компонента оновлюється, і React автоматично перемальовує інтерфейс, виводячи список карток RecipeCard.

Такий підхід до розробки користувацької частини гарантує високу продуктивність, модульність та повну відповідність сучасним стандартам веб-розробки.

2.4 Розробка серверної частини вебзастосунку для раціонального пошуку кулінарних рецептів

Серверна частина є фундаментальним компонентом вебзастосунку для раціонального пошуку кулінарних рецептів, оскільки вона відповідає за обробку бізнес-логіки, безпечне зберігання даних користувачів та інтеграцію із зовнішніми кулінарними сервісами. Для реалізації Backend-складової було обрано фреймворк Laravel, який базується на мові програмування PHP та використовує архітектурний патерн MVC [22]. Цей вибір зумовлений наявністю розвиненої екосистеми та вбудованих інструментів, що дозволяють значно прискорити розробку RESTful API, необхідного для взаємодії з клієнтською React-частиною.

Архітектурна структура серверної частини організована таким чином, щоб забезпечити максимальну розділеність обов’язків між компонентами системи. Усі вхідні HTTP-запити від клієнта проходять через спеціалізований маршрутизатор, де використання префікса дозволяє чітко відокремити запити до даних від видачі статичного контенту. Ці маршрути надійно захищені посередниками, які перевіряють автентичність токена користувача перед наданням доступу до функціоналу «Віртуального холодильника» [23]. Після проходження перевірки запити спрямовуються до відповідних контролерів, які викликають необхідну бізнес-логіку та повертають результат у форматі JSON.

Для взаємодії з базою даних MySQL замість написання сирих SQL-запитів використовується система об’єктно-реляційного відображення Eloquent [24]. Такий підхід дозволяє маніпулювати продуктами та рецептами як об’єктами, що автоматично захищає систему від атак типу SQL-ін’єкція та значно підвищує читабельність програмного коду.

Ключовою функцією серверної частини вебзастосунку для раціонального пошуку кулінарних рецептів є взаємодія із зовнішніми джерелами даних. Оскільки зберігання мільйонів рецептів у власній базі даних є недоцільним, реалізовано інтеграцію із професійним кулінарним API. Процес обробки передбачає, що сервер отримує запит від авторизованого користувача на підбір страв, після чого спеціальний сервісний шар витягує з бази даних список інгредієнтів, термін придатності яких добігає кінця. Використовуючи вбудований у Laravel HTTP-клієнт, сервер формує та відправляє запит до зовнішньої системи [25]. Отримана від стороннього сервісу відповідь ретельно фільтрується на сервері: видаляються дублікати та перевіряється наявність необхідної інформації, після чого оброблений масив даних відправляється на клієнтську частину.

При розробці серверної частини особлива увага приділялася безпеці даних та надійності системи. У вебзастосунку для раціонального пошуку кулінарних рецептів кожен вхідний запит проходить сувору валідацію на відповідність типам даних, формату дат та обов’язковості полів [26]. Для захисту облікових записів застосовується надійне хешування паролів за допомогою сучасних криптографічних алгоритмів. Крім того, налаштовано політики спільного використання ресурсів з різними джерелами, що дозволяє приймати запити виключно з довіреного домену клієнтської частини. Загалом, використання можливостей фреймворку Laravel дозволило створити надійний, безпечний та масштабований бекенд, який забезпечує стабільну роботу всіх модулів системи та високу швидкість обробки інформації.

2.5 Програмна реалізація логіки вебзастосунку для раціонального пошуку кулінарних рецептів

Основним завданням розроблюваного програмного продукту є не лише зберігання даних, а й виконання складних обчислювальних операцій для забезпечення інтелектуального підбору страв. Логічне ядро вебзастосунку для раціонального пошуку кулінарних рецептів побудоване навколо алгоритму аналізу продовольчих запасів користувача. Цей алгоритм ініціюється в момент переходу до розділу генерації меню і складається з послідовності взаємопов'язаних етапів обробки інформації [27]. Для списку рецептів було використано зовнішні сервіси TheMealDB, який надає системі список страв та список інгредієнтів, які необхідні для приготування, а також MyMemory API, що використовується для перекладу продуктів англійською мовою. Програмна реалізація цього алгоритму передбачає виконання наступних логічних кроків:

1. Збір та сортування локальних даних. Система звертається до бази даних та отримує масив усіх активних продуктів. Далі відбувається сортування цього масиву за датою закінчення терміну придатності у порядку зростання. Для подальшого аналізу алгоритм відбирає три найбільш критичні продукти, які потрібно спожити першими.

2. Машинний переклад інгредієнтів. Оскільки міжнародні кулінарні бази даних оперують переважно англійською мовою, система здійснює асинхронні запити до зовнішнього сервісу MyMemory API. На цьому етапі україномовні назви відібраних критичних продуктів перекладаються англійською мовою для формування коректного пошукового запиту.

3. Взаємодія з кулінарним API. Отримані англомовні терміни передаються до зовнішнього кулінарного REST API – TheMealDB. Спочатку виконується запит на фільтрацію, який повертає перелік страв, що містять заданий інгредієнт. Після цього алгоритм робить додаткові запити для отримання повної деталізації по кожному знайденому рецепту, включаючи інструкції з приготування, пропорції та фотографії страв.

4. Аналіз та зіставлення інгредієнтів. На цьому етапі відбувається обробка отриманих від API даних. Алгоритм розбирає кожен рецепт і порівнює список необхідних для нього інгредієнтів із загальним масивом продуктів, які наразі є у віртуальному холодильнику користувача. Програмна логіка динамічно формує два списки для кожного рецепта, а саме ті продукти які є в наявності та ті які потрібно додати

5. Формування та видача результату. Оброблений та структурований масив даних у форматі JSON повертається на клієнтську частину. Користувацький інтерфейс відображає картки рецептів, візуально підсвічуючи наявні інгредієнти зеленим кольором, а відсутні – червоним, що дозволяє користувачу миттєво оцінити доцільність вибору тієї чи іншої страви.

Блок-схему (рис. 2.4) побудовано за допомогою сервісу Lucidchart [28].

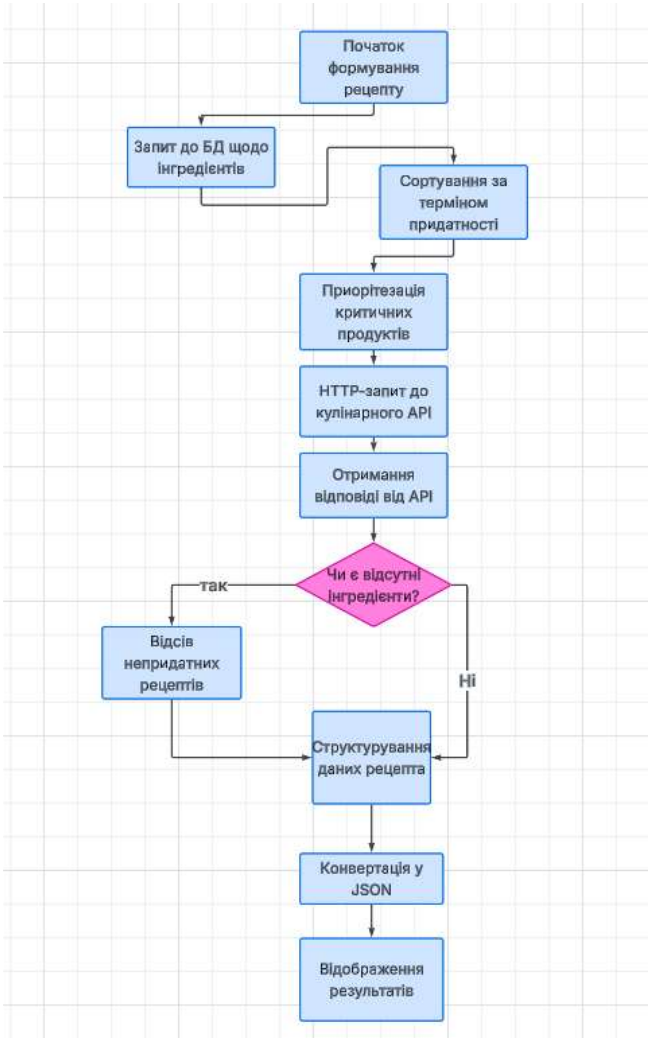

Рисунок 2.4 – Блок-схема алгоритму пошуку кулінарних рецептів

Логіка роботи системи тісно пов’язана з графічним інтерфейсом користувача, який на даному етапі розробки архітектури реалізується у вигляді візуальних прототипів та макетів. Процес побудови макета сайту застосовується для планування структури екранів віртуального холодильника та карток рецептів у спеціалізованому середовищі Figma [29]. На макеті головного екрана особистого кабінету проєктується візуальна логіка динамічного відображення запасів, де продукти з критичним терміном придатності виділяються відповідним кольоровим акцентом. Процес додавання нових інгредієнтів на прототипах супроводжується елементами інтерфейсу для миттєвої валідації введених даних, що в майбутньому підвищить стабільність застосунку. Візуальне представлення результатів підбору страв у вигляді wireframe-макетів підтверджує коректність закладеної логіки навігації та дозволяє узгодити користувацький досвід. Макет графічного інтерфейсу зображений на рисунку 2.5.

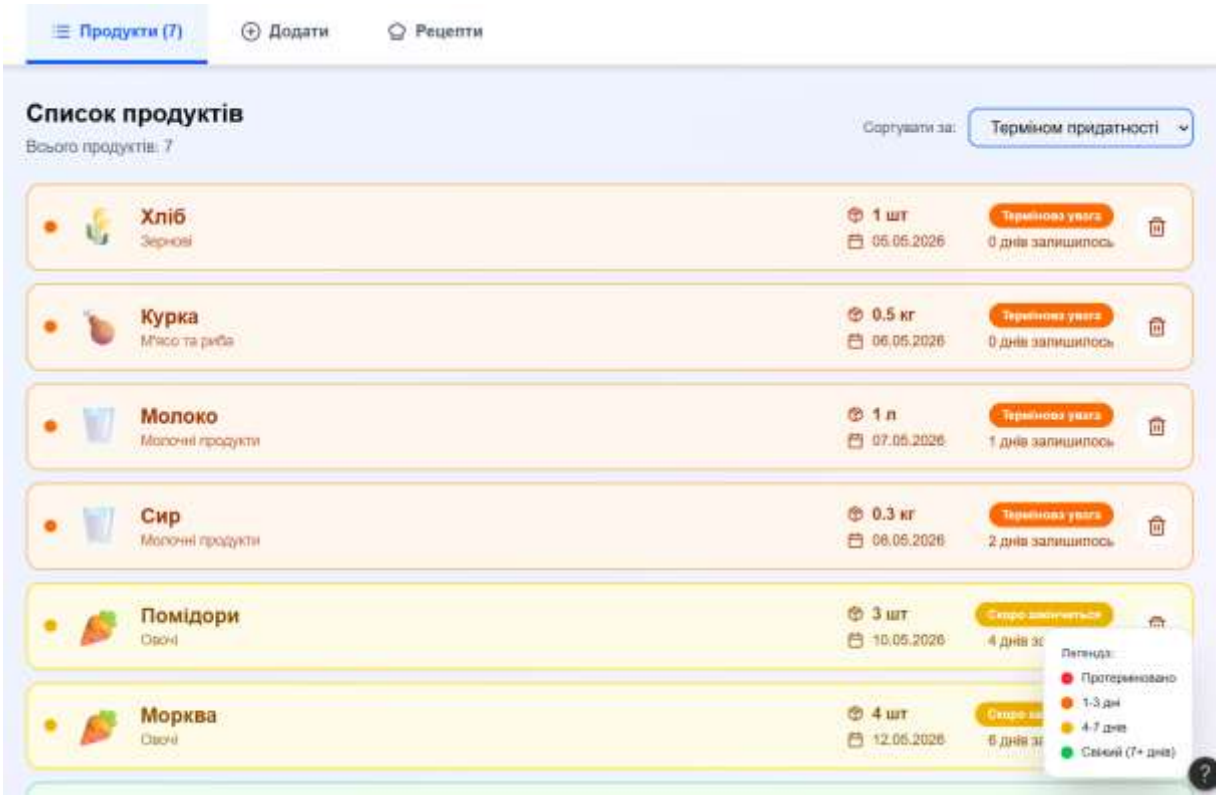

Рисунок 2.5 – Макет сайту

Сформулюємо висновки до розділу.

2.6 Висновки до другого розділу

У другому розділі було виконано детальне проєктування системи управління запасами продуктів та підбору рецептів.

Обґрунтовано архітектуру програмного рішення, було обрано клієнт-серверний підхід із використанням архітектурного стилю REST API. Такий розподіл забезпечує високу масштабованість системи, безпеку даних та чітке розмежування логіки серверної частини і користувацького інтерфейсу.

Спроєктовано структуру бази даних системи, а саме розроблено інформаційну модель та визначено ключові сутності: користувачі та продукти. Додано механізм відстеження життєвого циклу продукту за допомогою системи статусів (активний, спожитий, утилізований), що створює необхідне підґрунтя для подальшого збору статистики та розрахунку ефективності споживання. Розроблено алгоритмічну модель розумного підбору рецептів.

Побудовано логіку взаємодії системи із зовнішніми інтеграційними сервісами. Алгоритм враховує необхідність попереднього машинного перекладу інгредієнтів з подальшим зверненням до міжнародної кулінарної бази, а також включає механізми обробки винятків та зіставлення наявних продуктів із необхідними для страви.

Затверджено концепцію односторінкового застосунку з підтримкою токенової аутентифікації. Визначено основні екранні форми: панель авторизації, головний дашборд управління ресурсами, модуль аналітики (статистики) та інтерактивне модальне вікно рецептів.

Підсумовуючи, можна стверджувати, що обраний стек технологій, створена архітектура, спроєктована реляційна база даних та розроблені алгоритми формують повну та достатню основу для переходу до етапу безпосередньої програмної реалізації системи.

РОЗДІЛ 3. ДЕМОНСТРАЦІЯ ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ РАЦІОНАЛЬНОГО ПОШУКУ КУЛІНАРНИХ РЕЦЕПТІВ

3.1 Демонстрація функціоналу вебзастосунку для раціонального пошуку кулінарних рецептів

Готовий застосунок працює за принципом SPA. Оскільки клієнтська частина написана на базі бібліотеки React.js, всі переходи між сторінками і оновлення даних відбуваються миттєво, без необхідності повного перезавантаження сторінки браузера. Це робить інтерфейс максимально швидким, чуйним і зручним для кінцевого користувача, наближаючи досвід використання веб-застосунку до настільних програм.

Робота з програмою починається з вікна авторизації. Оскільки кожен користувач має свій власний ізольований віртуальний холодильник та індивідуальну статистику ефективності, доступ до основного функціоналу суворо закритий для неавторизованих відвідувачів. Щоб почати роботу, необхідно зареєструватися в системі або увійти в існуючий обліковий запис. Механізм авторизації забезпечує видачу унікального токена доступу, який згодом додається до кожного мережевого запиту.

Після успішної авторизації та перевірки токена сервером, користувач потрапляє на головний робочий простір системи – вкладку «Продукти». Інтерфейс робочого простору умовно поділено на три основні функціональні зони:

1. Навігаційна панель (Header): містить графічний логотип системи, електронну пошту або ім’я поточного користувача та кнопку безпечного виходу з облікового запису.
2. Панель управління та пошуку: містить інтерактивне текстове поле для швидкого пошуку продуктів за назвою та головну кнопку ініціалізації розумного алгоритму підбору рецептів.

3. Візуальна сітка продуктів: динамічно відображає наявні запаси користувача у вигляді компактних інформативних карток.

На кожній картці виводиться вся базова інформація, отримана з бази даних: назва продукту, його категорія та точний залишок із зазначенням одиниць виміру. Головна візуальна функція картки – це кольоровий індикатор свіжості. Програма має вбудовану логіку, яка в режимі реального часу вираховує різницю між поточною датою і датою закінчення терміну придатності. На основі цих обчислень система автоматично призначає статус, підсвічуючи картку відповідним кольором: зеленим (безпечно), помаранчевим (увага) або червоним (критично). На рисунку 3.1 можна побачити загальний вигляд головної сторінки вебзастосунку для раціонального пошуку кулінарних рецептів.

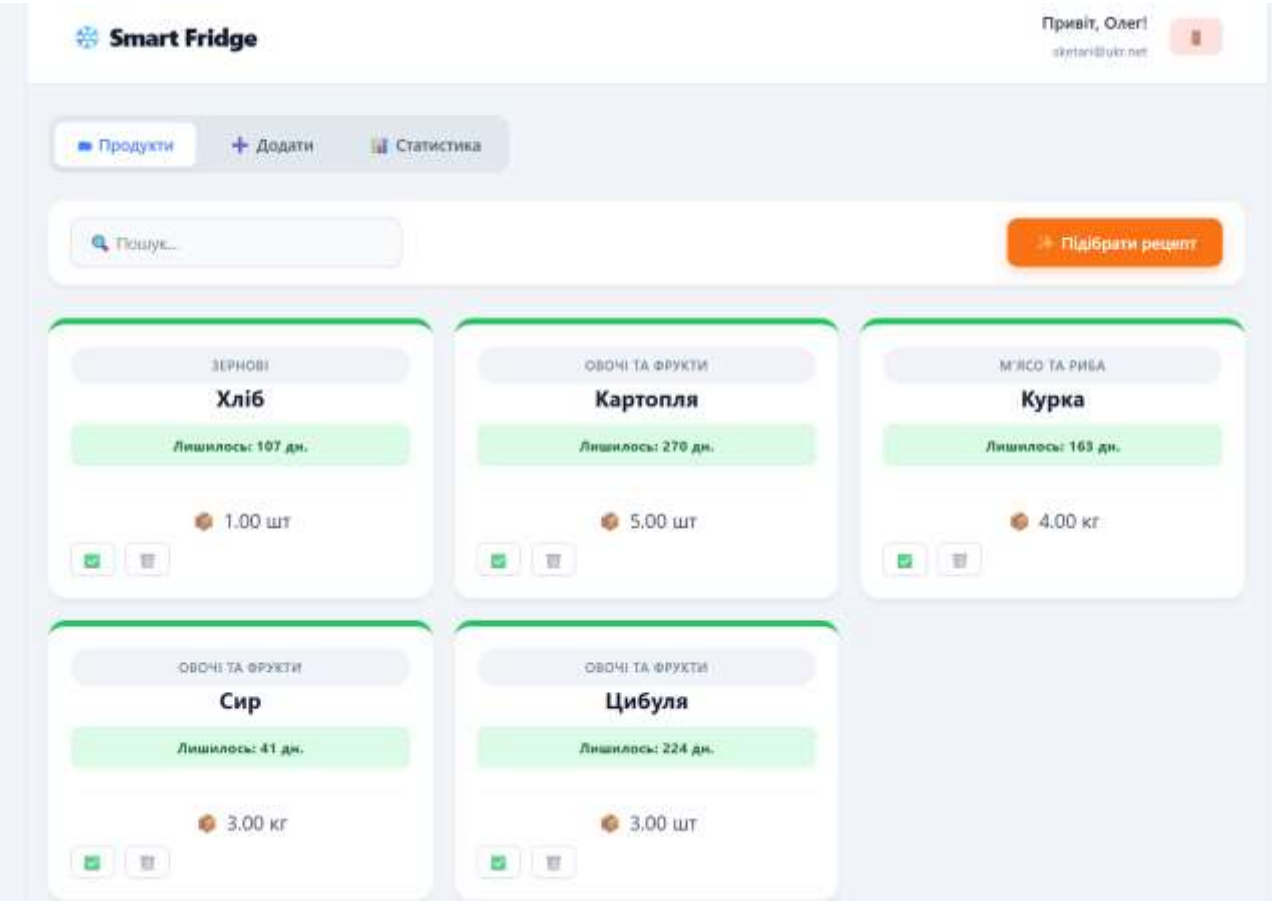

Рисунок 3.1 – Вигляд головної сторінки веб-застосунку

Крім відображення інформації, на картках передбачено елементи управління життєвим циклом та статистикою: користувач може відмітити, що продукт успішно спожито (для цього потрібно натиснути на кнопку, яка позначена зеленою галочкою), або вказати, що він зіпсувався і його довелося утилізувати (кнопка із зображенням кошика). Якщо продукт досягає критичної дати псування, система автоматично надає йому статус – «Зіпсовано».

Щоб користувачу не доводилося довго гортати весь список наявних запасів, було реалізовано механізм миттєвого текстового пошуку. Він працює в режимі реального часу на стороні клієнта: як тільки вводиться частина слова, клієнтська частина миттєво фільтрує масив даних (стан компонента) і залишає на екрані тільки відповідні результати, наприклад, картку з картоплею. Це значно пришвидшує навігацію, особливо коли віртуальний холодильник містить десятки позицій. Демонстрацію роботи пошуку наведено на рисунку 3.2.

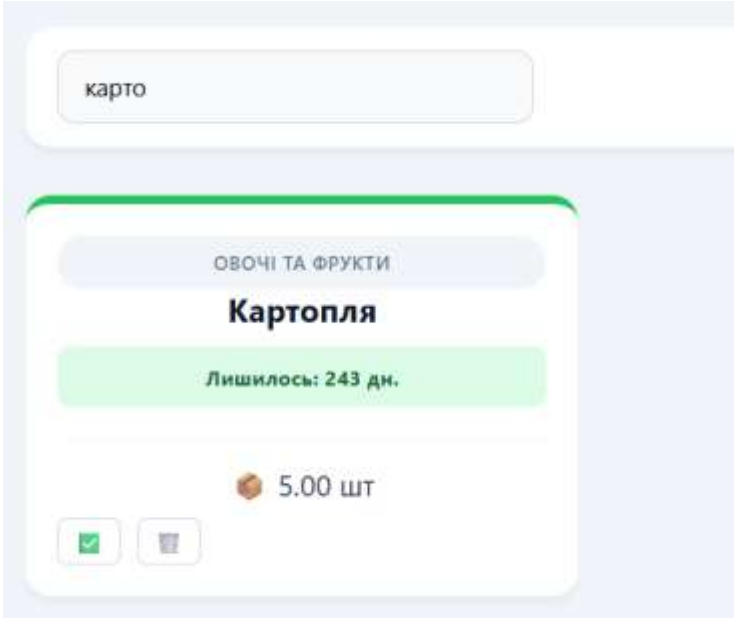

Рисунок 3.2 – Пошук продукту за назвою

Для додавання нових запасів до віртуального холодильника передбачено окрему вкладку «Додати». Форма введення спроєктована з урахуванням ергономіки: необхідно ввести назву, обрати категорію з випадаючого списку,

вказати числову кількість і за допомогою інтегрованого календаря встановити термін придатності. Перед відправкою HTTP-запиту на бекенд, форма проходить клієнтську валідацію, перевіряючи коректність та повноту заповнених полів. Вікно додавання продуктів зображено на рисунку 3.3.

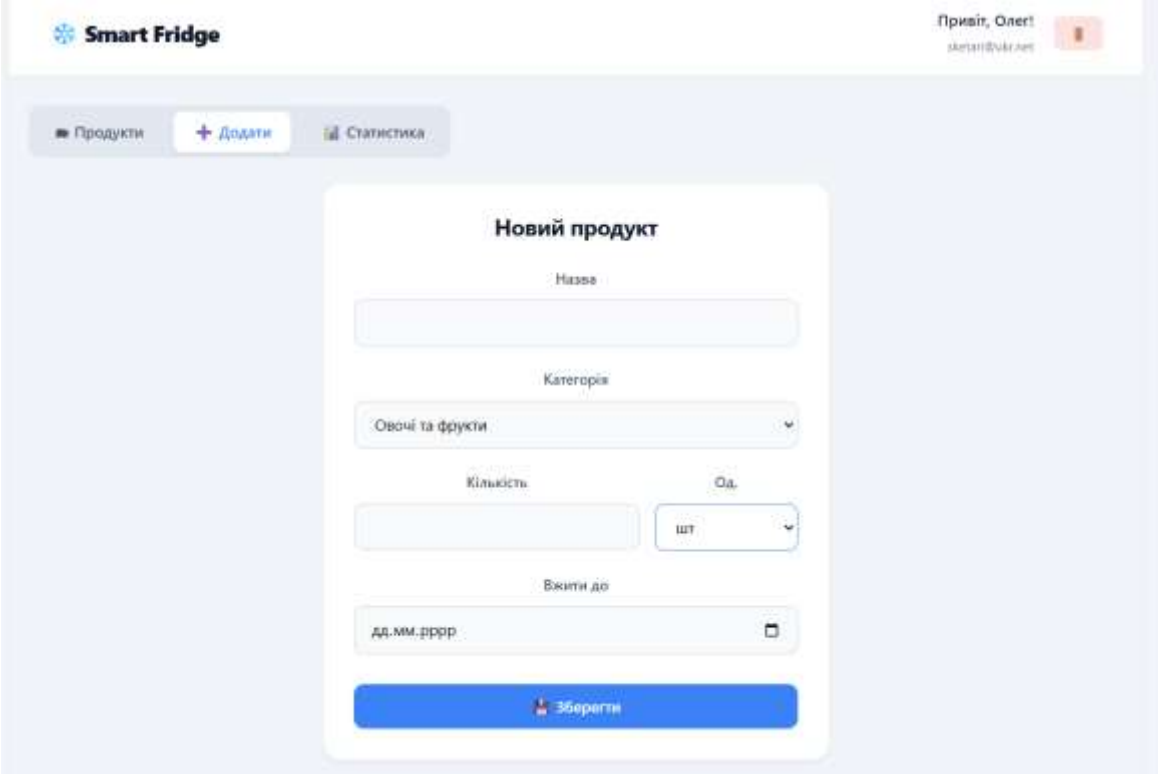

Рисунок 3.3 – Інтерфейс форми додавання нового продукту

Як вже зазначалося, важливою функцією інтерфейсу є динамічна зміна кольорів на картках залежно від терміну придатності. Якщо часу до псування залишається достатньо, картка відображається у нейтральному зеленому кольорі. Проте, якщо продукт знаходиться «на межі» – наприклад, як у випадку з молоком, якому залишився всього один день – система автоматично змінює стилізацію компонента, фарбуючи індикатор у червоний колір. Це дозволяє ефективно привертати увагу користувача до продуктів першочергового споживання. Приклад візуалізації продукту з критичним терміном придатності зображено на рисунку 3.4.

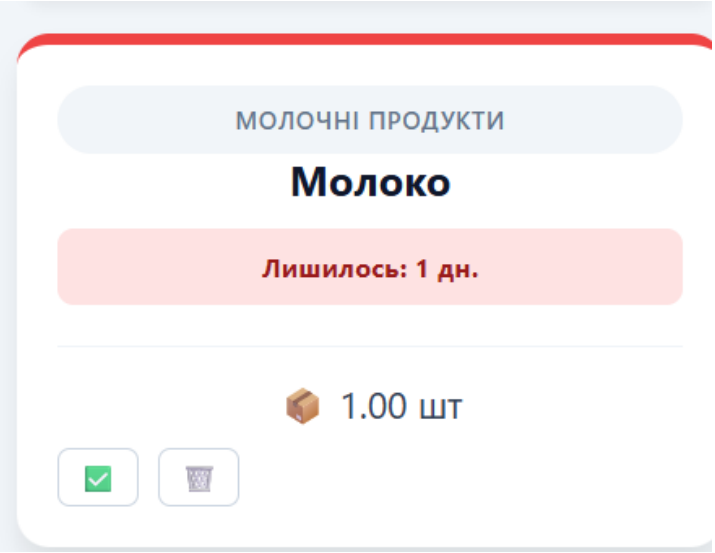

Рисунок 3.4 – Червона картка молока яке зіпсується через 1 день

Найскладніша частина застосунку з технічної точки зору – це генерація рецептів. Коли користувач натискає кнопку пошуку, система збирає список його продуктів, перекладає їхні назви на англійську мову і відправляє запити до зовнішньої кулінарної бази TheMealDB.

Знайдені страви з’являються у спливаючому модальному вікні. Програма не просто видає картинку і назву рецепту, а ще й розбирає його на інгредієнти та порівнює з тим, що вже є у холодильнику користувача. Зеленими тегами підсвічується те, що є в наявності, а червоними – те, що доведеться докупити в магазині. Приклад вікна підбору рецептів можна побачити на рисунку 3.5.

Оскільки міжнародна кулінарна база даних TheMealDB приймає запити виключно англійською мовою, було розроблено асинхронний алгоритм інтеграції декількох API. Логіка роботи цього алгоритму складається з наступних етапів:

1. Вибірка всіх продуктів користувача зі статусом «активний» та їх автоматичне сортування за критерієм найближчої дати псування.
2. Проходження назв продуктів через локальний словник перекладу (для базових продуктів) з метою мінімізації мережевих запитів.
3. У разі відсутності слова в словнику – звернення до MyMemory API для динамічного машинного перекладу інгредієнта англійською мовою.

4. Формування GET-запиту до TheMealDB API для отримання переліку страв, що містять перекладений інгредієнт.

5. Глибокий парсинг отриманого JSON-об'єкта рецепту: алгоритм циклічно перевіряє до 20 можливих інгредієнтів кожної страви (strIngredient1 ... strIngredient20) і зіставляє їх з наявними у базі даними користувача.

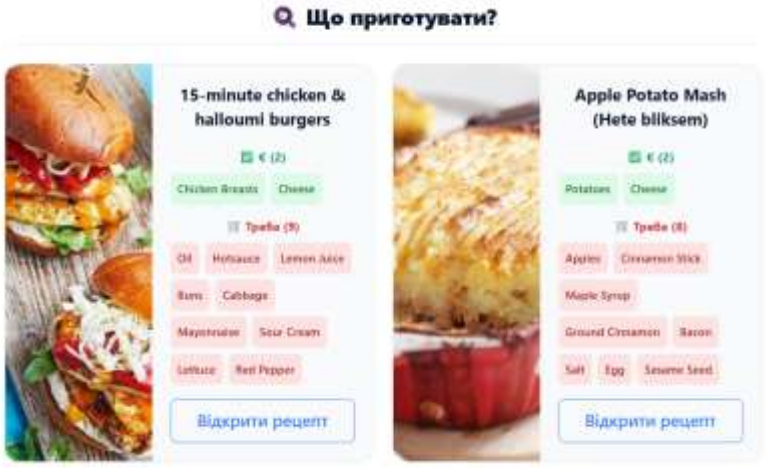

Рисунок 3.5 – Вікно підбору рецептів

Для реалізації цієї складної логіки було написано відповідний програмний код мовою JavaScript з використанням синтаксису async/await для обробки промісів. Основний фрагмент коду, що відповідає за переклад та зіставлення даних, наведено у лістингу 3.1.

Лістинг 3.1 – Фрагмент коду алгоритму перекладу та підбору рецептів

```
const findSmartRecipes = async () => {
    // Етап 1: Асинхронний переклад наявних запасів
     const translatedFridge = await Promise.all(products.map(async
(p) => ({
      original: p.name || 'Продукт',
      eng: await translateWord(p.name),
      expiry_date: p.expiry_date
    })));

    // Етап 2: Сортування за критичністю термінів
     const sortedFridge = [...translatedFridge].sort((a, b) => new
Date(a.expiry_date) - new Date(b.expiry_date));
    let meals = [];
```

```
    // Етап 3: Пошук страв за пріоритетними інгредієнтами
    for (let p of sortedFridge) {
      if (!p.eng) continue;
      try {
                              const      searchRes      =      await
recipeApi.get(`https://www.themealdb.com/api/json/v1/1/filter.php?
i=${p.eng}`);
         if (searchRes.data && Array.isArray(searchRes.data.meals))
{
           meals = searchRes.data.meals.slice(0, 4);
           break;
         }
      } catch(e) { console.error('Помилка пошуку страв'); }
    }

    // Етап 4: Парсинг інгредієнтів знайдених рецептів (скорочено)
      // Розподіл на масиви 'have' (в наявності) та 'missing'
(відсутні)
};
```

Щоб зробити застосунок цікавішим і додати елемент гейміфікації, було розроблено модуль статистики. Управління цією статистикою відбувається прямо з карток продуктів де є кнопки у вигляді зеленої галочки (означає, що продукт успішно з’їдено) та кошика (довелося викинути через псування).

На початковому етапі, коли користувач тільки почав вести облік і ще нічого не встиг спожити чи викинути, панель статистики абсолютно чиста. Як видно на рисунку 3.6, лічильники стоять на нулях, а загальна ефективність дорівнює 0%.

Підсумовуючи, можна зазначити, що розроблений вебзастосунок для раціонального пошуку кулінарних рецептів вийшов зручним та інтуїтивно зрозумілим.

Рисунок 3.6 – Початкова нульова статистика

Але як тільки ми почнемо натискати кнопки управління на картках, фронтенд відправить запит на бекенд, дані в базі оновляться, і система автоматично перерахує наш відсоток врятованих продуктів.

Коли користувач спробує з’їсти, або викинути якісь продукти його статистика автоматично буде змінюватись як на рисунку 3.7.

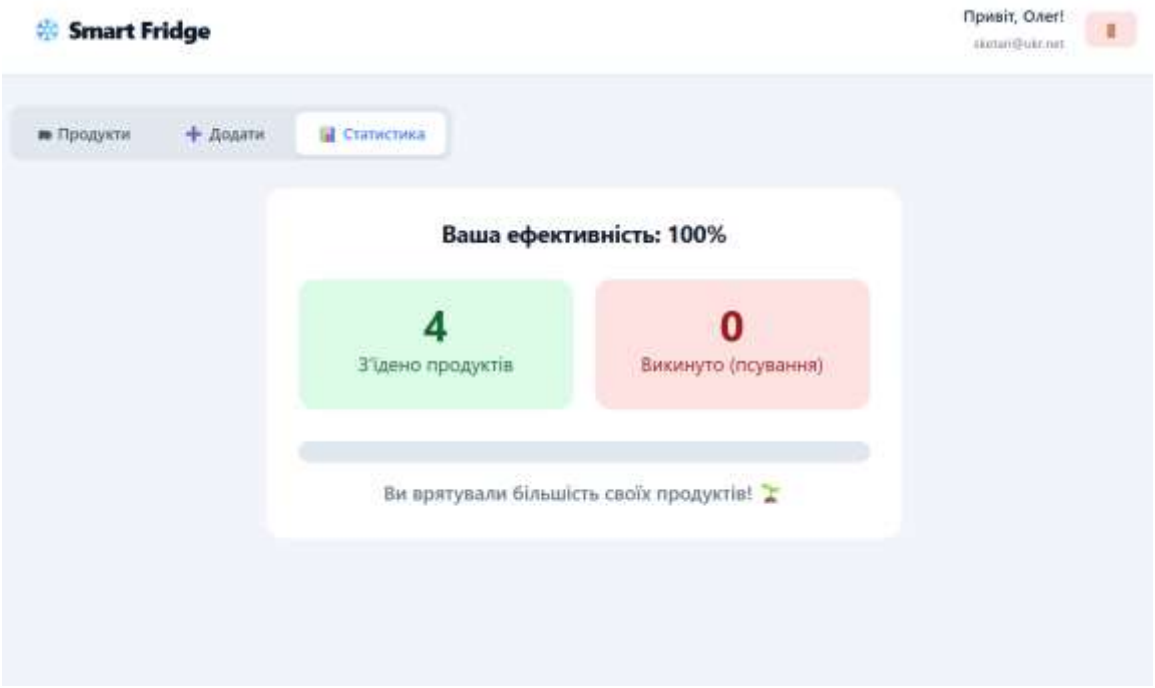

Рисунок 3.7 – Зміна статистики при використані вебзастосунку

Було успішно реалізовано весь запланований функціонал: облік продуктів, візуальний контроль термінів придатності та швидкий пошук рецептів. Завдяки використанню бібліотеки React застосунок працює швидко, а всі дані, списки та статистика оновлюються миттєво без перезавантаження сторінки. Усі основні модулі, зокрема авторизація користувачів, облік продуктів, динамічний контроль термінів придатності, розрахунок статистики ефективності та інтеграція із зовнішніми API для пошуку рецептів, функціонують згідно із закладеною логікою.

3.2 Типові сценарії роботи користувача з вебзастосунком для раціонального пошуку кулінарних рецептів

Для розуміння практичної цінності та підтвердження функціональної повноти розробленого програмного продукту наведено основні типові сценарії взаємодії (Use Cases) кінцевого користувача з веб-застосунком.

Сценарій 1: Поповнення бази запасів та ініціалізація відліку термінів придатності

- Умова виконання: Користувач здійснив закупівлю продовольчих товарів і має на меті внести їх у систему для подальшого обліку.
- Послідовність дій: Користувач переходить на вкладку «Додати», заповнює поля форми (назва, категорія, числова кількість, дата закінчення терміну придатності) та підтверджує збереження.
- Очікуваний результат: Сервер успішно записує дані в БД MySQL. На клієнтській стороні новий продукт миттєво з’являється у загальному списку. Система починає автоматично вираховувати кількість днів до псування, своєчасно змінюючи кольоровий статус картки при наближенні критичної дати.

Сценарій 2: Утилізація залишків та формування кулінарного запиту

- Умова виконання: Користувач стикається з проблемою вибору страви для приготування з урахуванням продуктів, термін придатності яких добігає кінця (наприклад, залишки курячого м’яса та овочів).
- Послідовність дій: Користувач ініціює алгоритм пошуку натисканням кнопки «Підібрати рецепт».
- Очікуваний результат: Алгоритм пріоритезує найшвидкопсувніші продукти, перекладає їх та формує запит до зовнішньої бази. Користувач отримує модальне вікно з варіантами страв, де чітко розмежовано інгредієнти, які вже є в наявності (зелені теги), та інгредієнти, що потребують додаткової закупівлі (червоні теги).

Сценарій 3: Фіксація дій та перерахунок аналітичної статистики

- Умова виконання: Користувач фізично використав певний продукт для приготування страви або був змушений його утилізувати через псування.
- Послідовність дій: На відповідній картці продукту в інтерфейсі користувач здійснює натискання на іконку підтвердження («З'їдено» або «Викинуто»).
- Очікуваний результат: Формується PUT-запит на бекенд, який змінює статус продукту, вилучаючи його з активного списку. Надалі при переході на вкладку «Статистика» система автоматично агрегує оновлені дані та перераховує відсоток загальної ефективності споживання їжі.

3.3 Тестування вебзастосунку для раціонального пошуку кулінарних рецептів

Для забезпечення надійності, безперебійної роботи та безпеки розробленої системи управління запасами було проведено комплексне тестування веб-застосунку. Процес тестування включав перевірку функціональних вимог, тестування безпеки та перевірку інтеграції із зовнішніми програмними інтерфейсами.

Функціональне тестування проводилося методом «чорного ящика» [31]. Основною метою було переконатися, що всі елементи користувацького інтерфейсу правильно взаємодіють із серверною частиною та базою даних. У таблиці 3.1 наведено основні тестові сценарії та результати їх виконання.

Усі базові CRUD-операції(Create, Read, Update, Delete) системи: створення, читання, оновлення та видалення працюють коректно. Валідація вхідних даних відбувається на двох рівнях: первинна – на стороні клієнта за допомогою засобів HTML5 та бібліотеки React для розвантаження сервера, та вторинна – на стороні сервера за допомогою механізмів Laravel Form Request для забезпечення повної цілісності бази даних та захисту від некоректних запитів [32].

Таблиця 3.1. – Порівняння переваг та недоліків

Опис тестового сценарію	**Очікуваний результат**	**Фактичний результат**	**Статус**
Реєстрація з існуючим email	Система видає помилку валідації та відмовляє у створенні дублікату запису.	Відображено повідомлення про помилку.	Успішно
Авторизація з невірним паролем	Відмова у доступі, повідомлення про невірні облікові дані (HTTP статус 401).	Доступ відхилено, виведено помилку.	Успішно
Додавання продукту без вказання назви	Блокування відправки форми на стороні клієнта (HTML5 валідація required).	Форма не відправляється, поле підсвічується.	Успішно
Зміна статусу продукту на «З'їдено»	Продукт зникає зі списку активних, лічильник спожитих товарів збільшується на 1.	Дані оновлено миттєво, статистика перерахована.	Успішно
Виклик пошуку рецептів з порожнім списком	Блокування запиту до API, виведення попередження «Холодильник порожній!».	Запит не виконано, показано модальне вікно «alert».	Успішно

Демонстрацію роботи клієнтської валідації зображено на рисунку 3.8.

Оскільки розроблена система обробляє облікові та персональні дані користувачів, окрему увагу було приділено тестуванню безпеки на основі методології та загальноприйнятих стандартів OWASP (Open Worldwide Application Security Project) [33]. Перевірка включала такі ключові етапи:

перевірка механізму аутентифікації та авторизації, захист від SQL-ін’єкцій, та захист від міжсайтового скриптингу.

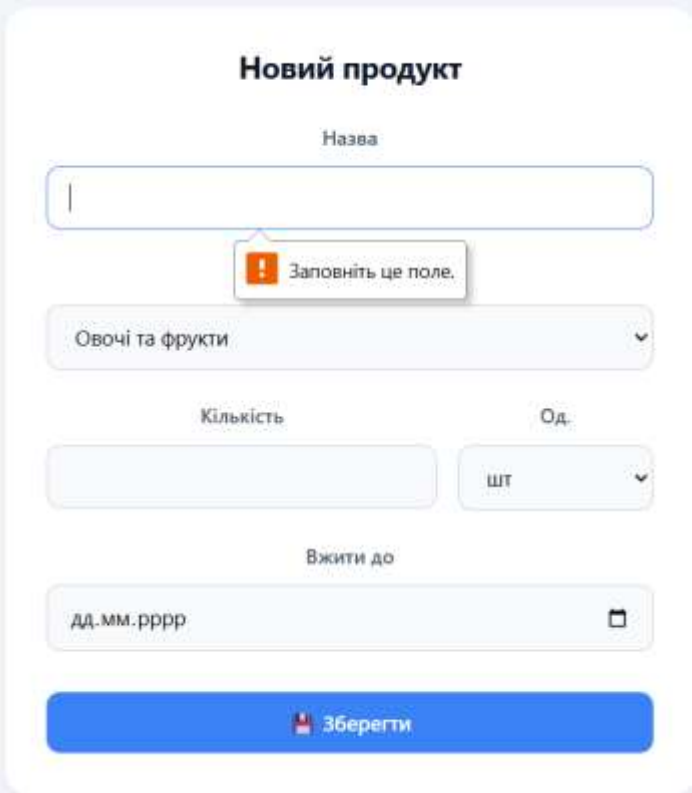

Рисунок 3.8 – Приклад спрацювання клієнтської валідації при спробі відправити порожню форму

Перевірка механізму аутентифікації та авторизації. Система використовує технологію Laravel Sanctum для генерації та перевірки Bearer-токенів [34]. Було проведено спробу отримати список продуктів через прямий HTTP-запит (GET /api/products) без передачі валідного токена у заголовку Authorization. Сервер коректно відхилив запит, повернувши статус помилки 401 Unauthorized, що підтверджує надійний захист доступу до чужих даних на рівні бекенду. На рисунку 3.9 можемо побачити приклад такої помилки.

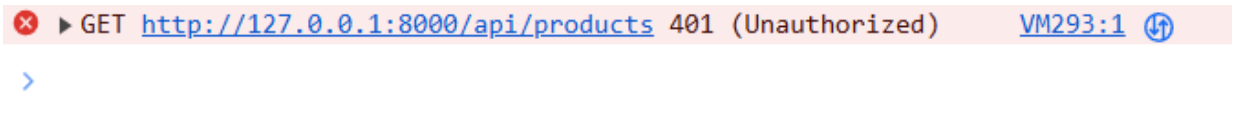

Рисунок 3.9 – Відхилення несанкціонованого запиту сервером

Захист від SQL-ін’єкцій. Було здійснено спробу впровадження шкідливого SQL-коду через форму авторизації та поле текстового пошуку. Оскільки на сервері використовується механізм об’єктно-реляційного відображення Eloquent ORM, усі вхідні дані автоматично екрануються за допомогою параметризованих запитів PHP Data Objects (PDO) [35]. Вразливість була повністю нейтралізована, а система сприйняла ввід як звичайний безпечний текст.

Захист від міжсайтового скриптингу. У форму додавання продукту, в поле найменування, було введено базовий шкідливий JavaScript-код: <script>alert('Test');</script>. Завдяки особливостям архітектури, бібліотека React автоматично екранує всі рядкові змінні перед їх виведенням у віртуальний DOM [36]. Скрипт не виконався у браузері користувача, а безпечно відобразився на екрані як звичайний текстовий рядок. На рисунку 3.10 видно картку продукту який додався як звичайний незважаючи на шкідливий код в назві.

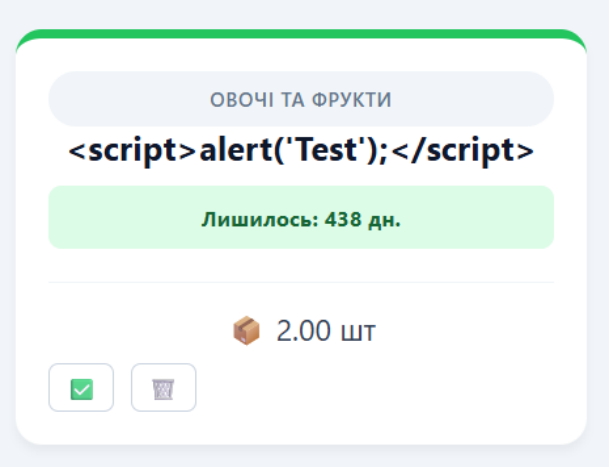

Рисунок 3.10 – Спроба виконання XSS-атаки через форму додавання продукту

Особливістю архітектури застосунку є залежність від зовнішніх мережевих сервісів: MyMemory API для машинного перекладу та TheMealDB API для доступу до кулінарної бази. Було проведено тестування обробки виняткових ситуацій при взаємодії з цими сервісами [37].

Було змодельовано ситуацію, коли користувач ініціює алгоритм пошуку для специфічного інгредієнта, якого фізично не існує в базі TheMealDB (наприклад, випадковий набір символів). Застосунок не завершив роботу аварійно і не призвів до помилки рендерингу компонента (так званого «білого екрану»). Замість цього розроблений алгоритм коректно обробив порожню відповідь від зовнішнього API та вивів передбачене інформаційне повідомлення з проханням додати базові інгредієнти. Також було перевірено стійкість системи до мережевих затримок – під час очікування відповіді від стороннього сервера користувачу коректно демонструється візуальний індикатор завантаження.

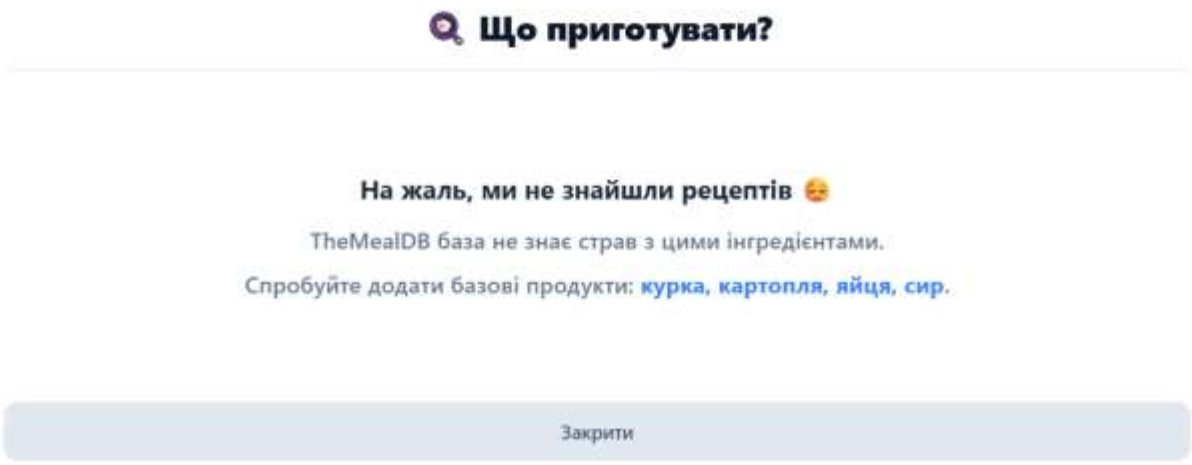

Рисунок 3.11 – Обробка системою відсутності рецептів для вказаних інгредієнтів

На фінальному етапі тестування було перевірено коректність відображення інтерфейсу на різних роздільних здатностях екрану (Responsive Design). Завдяки використанню сучасних підходів до верстки, гнучких CSS-сіток (Flexbox та CSS Grid) та медіа-запитів, застосунок є адаптивним і коректно масштабується від екранів мобільних пристроїв до широкоформатних моніторів [38].

За допомогою вбудованих інструментів аудиту Google Chrome DevTools було перевірено швидкість завантаження клієнтської частини. Завдяки

використанню сучасного збирача модулів Vite та автоматичній мініфікації вихідного коду, час початкового відмальовування складає менше однієї секунди [39], що забезпечує високий рівень користувацького досвіду та швидку реакцію системи на дії користувача.

3.4 Висновки до третього розділу

У третьому розділі було детально розглянуто процеси практичної реалізації, демонстрації функціоналу та тестування розробленого веб-застосунку для раціонального пошуку кулінарних рецептів. В ході роботи було продемонстровано клієнтську частину системи, побудовану за архітектурним патерном SPA з використанням бібліотеки React. Було успішно реалізовано інтуїтивно зрозумілий користувацький інтерфейс, який забезпечує зручне управління запасами, швидкий текстовий пошук продуктів та автоматичний візуальний контроль термінів придатності за допомогою динамічної зміни кольорів.

Окрему увагу було приділено опису програмної логіки. Зокрема, наведено фрагменти коду ключового компонента системи – багатоступеневого алгоритму інтелектуального підбору рецептів. Даний алгоритм вирішує складну задачу інтеграції локальних даних із зовнішньою кулінарною базою TheMealDB шляхом проміжного машинного перекладу інгредієнтів через MyMemory API. Для підтвердження практичної цінності продукту було змодельовано основні типові сценарії взаємодії користувача з системою. Розглянуті ситуації доводять, що застосунок повністю покриває реальні потреби: від повсякденного обліку продовольства до формування списків покупок та ведення гейміфікованої статистики ефективності.

Важливим завершальним етапом стало комплексне тестування розробленого програмного забезпечення. Функціональне тестування підтвердило правильність виконання всіх операцій з базою даних та надійність дворівневої системи валідації. Перевірка безпеки за міжнародними стандартами

OWASP довела стійкість застосунку до найпоширеніших загроз, зокрема SQL-ін’єкцій та Міжсайтового скриптингу (XSS-атак). Інтеграційне тестування та перевірка продуктивності показали, що система коректно обробляє виняткові ситуації при збоях сторонніх сервісів, є адаптивною до різних екранів та має високу швидкість відмальовування інтерфейсу. Таким чином, усі завдання, поставлені на етап розробки та тестування, виконано в повному обсязі, а програмний продукт є стабільним і готовим до експлуатації.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Психологічні чинники небезпеки

У структурі причин виробничого травматизму, аварійності та виникнення надзвичайних ситуацій техногенного характеру домінуючу позицію традиційно займає так званий «людський фактор». Сучасна наука про безпеку життєдіяльності та охорону праці розглядає людину як надзвичайно складну психофізіологічну систему, здатну адаптуватися до змін, але водночас вразливу до впливу негативних чинників виробничого та навколишнього середовища. Психологічні чинники небезпеки являють собою сукупність індивідуальних психологічних особливостей працівника та тимчасових станів його центральної нервової системи, які за певних умов здатні негативно впливати на безпеку трудового процесу, призводити до помилкових дій, зниження пильності та провокувати нещасні випадки. До фундаментальних різновидів таких чинників належать нервово-психічне перенапруження, інформаційне перевантаження, монотонність праці, емоційний стрес, перевтома, а також стан підвищеної тривожності та фрустрації [40].

Механізм впливу психологічних небезпек на організм людини найчастіше проявляється через розвиток стану втоми та перевтоми. Втома є природною фізіологічною реакцією організму на тривалу або інтенсивну роботу, проте за відсутності належного відпочинку вона переростає у хронічну перевтому. Цей стан супроводжується відчуттям млявості, притупленням уваги, збільшенням часу сенсомоторних реакцій та зниженням мотивації до дотримання правил техніки безпеки. Працівник у стані перевтоми схильний свідомо чи несвідомо ігнорувати використання засобів індивідуального захисту та спрощувати безпечні алгоритми виконання робіт, намагаючись заощадити власні фізичні зусилля. Не менш небезпечним фактором є монотонність праці, яка виникає внаслідок багаторазового повторення одноманітних дій або тривалого пасивного спостереження за приладами. Вона викликає стан емоційного

вигорання, підвищену сонливість та втрату концентрації, що різко підвищує ймовірність травматизму через автоматизм рухів і зниження критичного мислення [41].

В умовах сьогодення, враховуючи реалії повномасштабної війни в Україні, вплив психофізіологічних та психоемоційних чинників небезпеки на працездатність людини набув безпрецедентного і критичного значення. Постійна екзистенційна загроза, регулярні сигнали повітряної тривоги, порушення нормального режиму сну через нічні атаки, переживання за життя і здоров'я близьких та постійне перебування в агресивному, травматичному інформаційному полі призводять до глибокого виснаження резервів нервової системи працівників. Хронічний стрес, емоційна лабільність та прояви посттравматичного стресового розладу стають масовими явищами, які безпосередньо загрожують безпеці праці на будь-якому підприємстві. У стані гострого або хронічного дистресу в людини суттєво звужується обсяг оперативної пам'яті, порушується координація рухів, а також критично знижується здатність до адекватного та швидкого прийняття рішень в аварійних або нестандартних ситуаціях [42].

Окрім зовнішніх факторів, значну роль відіграють внутрішні психологічні установки особистості, зокрема схильність до невиправданого ризику, професійна деформація, недбалість, надмірна самовпевненість або навпаки – невпевненість у власних силах. Стан паніки або афекту, який може раптово виникнути під час аварій чи зовнішніх загроз, є вкрай небезпечним, оскільки він повністю блокує раціональне мислення людини. У такому стані індивід керується виключно базовими інстинктами, що спонукає його до неконтрольованих, хаотичних дій, які часто суперечать інструкціям з охорони праці і лише погіршують наслідки надзвичайної ситуації для самого працівника та його оточення [40].

Для мінімізації та нейтралізації негативного впливу психологічних чинників небезпеки на підприємствах повинна впроваджуватися системна превентивна робота. Вона обов'язково включає науково обґрунтовану

оптимізацію режимів праці та відпочинку, впровадження ергономічних вимог до організації робочих місць, а також проведення суворого психофізіологічного професійного відбору для видів робіт з підвищеною небезпекою. Вкрай важливою є роль якісних інструктажів з охорони праці та регулярних практичних тренінгів, які допомагають працівникам довести до автоматизму алгоритми дій в екстремальних умовах, що значно знижує рівень тривожності та запобігає виникненню паніки. Забезпечення психологічного супроводу персоналу, створення сприятливого, підтримуючого мікроклімату в колективі та розуміння керівництвом психоемоційного стану підлеглих сьогодні є невід'ємною і чи не найголовнішою складовою сучасної культури безпеки праці [42].

4.2 Психофізіологічне розвантаження для працівників

Психофізіологічне розвантаження є одним із найважливіших елементів сучасної системи охорони праці, спрямованим на збереження здоров'я, відновлення працездатності та профілактику професійних захворювань серед працівників. В умовах інтенсифікації виробничих процесів, високого ступеня автоматизації робочих місць та зростання частки інтелектуальної праці, нервово-емоційне напруження стає домінуючим фактором втоми. Психофізіологічне розвантаження являє собою комплекс цілеспрямованих організаційних, санітарно-гігієнічних та медико-біологічних заходів, які дозволяють швидко зняти накопичену втому, зменшити рівень стресу та відновити оптимальний функціональний стан центральної нервової системи впродовж робочої зміни [43].

Ефективним і науково обґрунтованим методом реалізації цих заходів на підприємствах є створення спеціальних кабінетів або кімнат психофізіологічного розвантаження. Їх облаштування суворо регламентується відповідними санітарними нормами та правилами. Інтер'єр таких приміщень проєктується з урахуванням принципів кольоротерапії: використовуються

м’які, пастельні тони, які заспокійливо діють на зоровий аналізатор і сприяють релаксації. Кімнати обладнуються зручними кріслами з відкидними спинками для забезпечення максимального розслаблення м'язового корсета. Вкрай важливою складовою є використання функціональної музики або звукотерапії – трансляція спеціально підібраних класичних композицій, звуків природи чи нейтральних мелодій, що сприяють гальмуванню процесів патологічного збудження в корі головного мозку [44].

Окрім пасивного відпочинку в спеціально обладнаних приміщеннях, вагоме місце в системі психофізіологічного розвантаження посідають активні методи. До них належить виробнича гімнастика, яка включає фізкультпаузи та фізкультхвилинки. Такі заходи сприяють перерозподілу крові в організмі, зняттю напруги з робочих груп м’язів, покращенню мозкового кровообігу та усуненню застійних явищ, що є особливо критичним при малорухливій роботі за комп’ютером. Також активно застосовуються методи аутогенного тренування. Навчання працівників базовим навичкам саморегуляції дихання та свідомого розслаблення тіла дозволяє їм самостійно знижувати рівень емоційного напруження безпосередньо на робочому місці, що мінімізує ризик виникнення виробничих конфліктів та нещасних випадків через втрату уваги [44].

Особливої актуальності питання психофізіологічного розвантаження набуває в умовах воєнного стану. Хронічний стрес, спричинений зовнішніми загрозами, вимагає від роботодавців перегляду класичних режимів праці та відпочинку. Відповідно до сучасних рекомендацій у сфері охорони праці, на підприємствах необхідно впроваджувати гнучкі графіки регламентованих перерв, забезпечувати наявність безпечних і комфортних зон відпочинку безпосередньо в укриттях, а також надавати персоналу доступ до програм психологічної підтримки. Раціональна організація психофізіологічного розвантаження не лише виконує гуманістичну функцію захисту здоров'я людини, але й має виражений економічний ефект, оскільки достовірно зменшує

кількість помилок, мінімізує плинність кадрів та підвищує загальну продуктивність праці в умовах кризових ситуацій [45].

Розглядаючи питання психофізіологічного розвантаження в контексті розробки програмного забезпечення, зокрема під час створення комплексних вебзастосунків, варто відзначити специфіку праці ІТ-фахівців. Процес проєктування архітектури, написання програмного коду, налаштування серверної частини бази даних та тестування багатоступеневих алгоритмів вимагає максимальної концентрації уваги, інтенсивного логічного мислення та безперервної зорової фіксації на екрані монітора. Така діяльність неминуче призводить до специфічного профілю втоми: надмірного когнітивного навантаження, зорової перевтоми (астенопії), а також статичного напруження м’язів шийно-грудного відділу хребта через тривале перебування у сидячому положенні.

З огляду на це, впровадження профілактичних заходів для розробників інформаційних систем набуває виняткового значення. Особливу увагу на робочому місці програміста необхідно приділяти профілактиці зорової втоми шляхом регулярного виконання спеціальної гімнастики для очей, що дозволяє зняти спазм акомодації після тривалої роботи з текстовими редакторами та інтерфейсами. Крім того, періодична зміна виду діяльності з інтелектуальної на легку рухову активність допомагає відновити кровообіг головного мозку. Це безпосередньо впливає на здатність фахівця генерувати ефективні алгоритмічні рішення, оптимізувати бази даних та уникати критичних помилок під час написання коду. Таким чином, підтримка оптимального психофізіологічного стану розробника є не лише питанням збереження його здоров'я, але й фундаментальним фактором забезпечення високої якості, безпеки та надійності створюваного програмного продукту.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи проведена розробка веб-застосунку для раціонального пошуку кулінарних рецептів із використанням фреймворків React та Laravel.

В першому розділі роботи наведено результати аналізу предметної області, які показали, що сучасні системи обліку продуктів та пошуку рецептів часто є розрізненими, не мають зручного адаптивного інтерфейсу або перевантажені зайвим функціоналом і рекламою. Це підкреслює необхідність створення зручного, інтуїтивно зрозумілого веб-застосунку, який об'єднує ці функції. Вибір технологій для розробки був обґрунтованим. Використання бібліотеки React для клієнтської частини та PHP-фреймворку Laravel для серверної забезпечило високу продуктивність, надійність та зручність розробки. Застосування підходу SPA дозволило максимально покращити швидкість відгуку та загальний користувацький досвід.

В другому розділі роботи розглянуто архітектуру застосунку, побудовану на основі клієнт-серверної моделі та архітектурного патерну MVC, яка забезпечила модульність, масштабованість та безпеку системи. Використання системи об'єктно-реляційного відображення Eloquent ORM та СУБД MySQL спростили створення надійної структури бази даних та забезпечили ефективне управління даними. Централізація бізнес-логіки на сервері дозволила оптимізувати взаємодію між інтерфейсом, базою даних і зовнішніми API, що сприяло стабільності та зручності підтримки коду. Функціональні можливості системи, включаючи облік продуктів, контроль термінів придатності, генерацію статистики ефективності та підбір рецептів, були успішно спроєктовані.

В третьому розділі роботи продемонстровано роботу вебзастосунку для раціонального пошуку кулінарних рецептів. Тестування та валідація застосунку, проведені методами функціонального тестування та перевірки безпеки за стандартами OWASP, показали відсутність критичних помилок та високу стабільність роботи. Розроблений багатоступеневий алгоритм інтеграції

з MyMemory API та TheMealDB API довів свою ефективність у машинному перекладі та точному пошуку інгредієнтів. Виявлені під час тестування незначні недоліки були проаналізовані та виправлені, що забезпечило відповідність проєкту сучасним стандартам веб-розробки.

В розділі «Безпека життєдіяльності, основи охорони праці» підкреслюється важливість розуміння впливу психологічних чинників небезпеки на працівників в умовах воєнного стану, а також значення заходів з психофізіологічного розвантаження для збереження здоров’я, зниження рівня стресу та забезпечення безпеки праці на підприємстві.

У цілому, розроблений веб-застосунок відповідає поставленій меті та завданням, демонструючи високий рівень функціональності, зручності та потенціалу для подальшого розвитку.

ПЕРЕЛІК ДЖЕРЕЛ

1 UNEP Food Waste Index Report 2024 [Електронний ресурс] – Режим доступу: https://www.unep.org/resources/publication/food-waste-index-report-2024 (дата звернення: 07.05.2026).

2 SuperCook – Zero Waste Recipe Generator [Електронний ресурс] – Режим доступу: https://www.supercook.com/#/desktop (дата звернення: 07.05.2026).

3 Allrecipes: Food, Friends, and Recipe Inspiration. [Електронний ресурс] – Режим доступу: https://www.allrecipes.com/ (дата звернення: 07.05.2026).

4 KitchenAid. [Електронний ресурс] – Режим доступу: https://www.kitchenaid.com/major-appliances.html (дата звернення: 07.05.2026).

5 Laravel Documentation. [Електронний ресурс] – Режим доступу: https://laravel.com/docs/12.x (дата звернення: 07.05.2026).

6 React – A JavaScript library for building user interfaces. [Електронний ресурс] – Режим доступу: https://react.dev/ (дата звернення: 07.05.2026).

7 Richardson L., Amundsen M., Ruby S. RESTful Web APIs: Services for a Changing World. – O'Reilly Media, 2013. – 406 с.

8 Flanagan D. JavaScript: The Definitive Guide. 7th Edition. – O'Reilly Media, 2020. – 706 с.

9 Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 2nd Edition. – O'Reilly Media, 2020. – 338 с.

10 Stauffer M. Laravel: Up & Running: A Framework for Building Modern PHP Apps. 3rd Edition. – O'Reilly Media, 2023. – 582 с.

11 Spoonacular Food API Documentation. [Електронний ресурс]. – Режим доступу: https://spoonacular.com/food-api/docs (дата звернення: 07.05.2026).

12 Laravel Eloquent ORM Documentation. [Електронний ресурс]. – Режим доступу: https://laravel.com/docs/eloquent (дата звернення: 07.05.2026).

13 Blogger. “ Діаграми Прецедентів” [Електронний ресурс]. – Режим доступу: https://lvivqaclub.blogspot.com/2008/10/use-case-uml-diagram.html (дата звернення: 12.05.2026).

14 Introduction to web APIs. MDN Web Docs. [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction (дата звернення: 12.05.2026).

15 Microsoft REST API Guidelines. [Електронний ресурс]. – Режим доступу: https://github.com/microsoft/api-guidelines (дата звернення: 12.05.2026).

16 Побудова діаграм. Diagrams.net. [Електронний ресурс]. Режим доступу до ресурсу: https://app.diagrams.net/ (дата звернення: 12.05.2026).

17 10 Usability Heuristics for User Interface Design. Nielsen Norman Group. [Електронний ресурс]. – Режим доступу: https://www.nngroup.com/articles/ten-usability-heuristics/ (дата звернення: 16.05.2026).

18 W3C Schools. “ HTML Semantic Elements” [Електронний ресурс]. Режим доступу до ресурсу: https://www.w3schools.com/html/html5_semantic_elements.asp (дата звернення: 16.05.2026).

19 React Official Documentation: Describing the UI. [Електронний ресурс]. – Режим доступу: https://react.dev/learn/describing-the-ui (дата звернення: 16.05.2026).

20 React Router: Declarative Routing for React. [Електронний ресурс]. – Режим доступу: https://reactrouter.com/en/main (дата звернення: 17.05.2026).

21 Axios: Promise based HTTP client for the browser and node.js. [Електронний ресурс]. – Режим доступу: https://axios-http.com/docs/intro (дата звернення: 17.05.2026).

22 Laravel Documentation: The PHP Framework for Web Artisans. [Електронний ресурс]. – Режим доступу: https://laravel.com/docs/13.x (дата звернення: 17.05.2026).

23 Laravel Middleware: Filtering Requests. [Електронний ресурс]. – Режим доступу: https://laravel.com/docs/13.x/middleware (дата звернення: 19.05.2026).

24 Eloquent ORM: Getting Started. Laravel Official Documentation. [Електронний ресурс]. – Режим доступу: https://laravel.com/docs/13.x/eloquent (дата звернення: 19.05.2026).

25 Laravel HTTP Client (Guzzle Wrapper). [Електронний ресурс]. – Режим доступу: https://laravel.com/docs/13.x/http-client (дата звернення: 20.05.2026).

26 Validation in Laravel: Protecting Your Data. [Електронний ресурс]. – Режим доступу: https://laravel.com/docs/13.x/validation (дата звернення: 20.05.2026).

27 Software Logic and Algorithm Design. [Електронний ресурс]. – Режим доступу: https://www.khanacademy.org/computing/computer-science/algorithms (дата звернення: 20.05.2026).

28 Lucidchart: Intelligent Diagramming Environment. [Електронний ресурс]. – Режим доступу: https://www.lucidchart.com/ (дата звернення: 25.05.2026).

29 Figma: The Collaborative Interface Design Tool. [Електронний ресурс]. – Режим доступу: https://www.figma.com/ (дата звернення: 26.05.2026).

30 Mermaid Live Editor. [Електронний ресурс]. – Режим доступу: https://mermaid.live/edit (дата звернення: 26.05.2026).

31 White/black/grey box-тестування. [Електронний ресурс]. – Режим доступу: https://qalight.ua/baza-znaniy/white-black-grey-box-testuvannya/ (дата звернення: 29.05.2026).

32 Validation – Laravel – The PHP Framework For Web Artisans. [Електронний ресурс]. – Режим доступу: https://laravel.com/docs/11.x/validation (дата звернення: 04.06.2026).

33 OWASP Top 10:2021. The Ten Most Critical Web Application Security Risks. OWASP Foundation. [Електронний ресурс]. – Режим доступу: https://owasp.org/Top10/2025/ (дата звернення: 04.06.2026).

34 Laravel Sanctum – Laravel – The PHP Framework For Web Artisans. [Електронний ресурс]. – Режим доступу: https://laravel.com/docs/11.x/sanctum (дата звернення: 04.06.2026).

35 Prepared statements and stored procedures. PHP Manual. [Електронний ресурс]. – Режим доступу: https://www.php.net/manual/en/pdo.prepared-statements.php (дата звернення: 04.06.2026).

36 Writing Markup with JSX. React Documentation. [Електронний ресурс]. – Режим доступу: https://react.dev/learn/writing-markup-with-jsx (дата звернення: 04.06.2026).

37 REST API HTTP Status Codes. RESTful API Tutorial. [Електронний ресурс]. – Режим доступу: https://restfulapi.net/http-status-codes/ (дата звернення: 05.06.2026).

38 Responsive design – Learn web development. MDN Web Docs. [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Responsive_Design (дата звернення: 05.06.2026).

39 First Contentful Paint (FCP). web.dev. [Електронний ресурс]. – Режим доступу: https://web.dev/articles/fcp (дата звернення: 05.06.2026).

40 Желібо Є. П., Заверуха Н. М., Зацарний В. В. Безпека життєдіяльності : навч. посіб. для студ. вищ. навч. закл. / за ред. Є. П. Желібо. 6-те вид. Київ : Каравела, 2018. 344 с.

41 Психологічна безпека та допомога в умовах війни : метод. рекомендації / Національна академія педагогічних наук України. Київ : Інститут психології імені Г. С. Костюка, 2022. 56 с.

42 Крушельницька Я. В. Фізіологія і психологія праці : навч. посіб. Київ : КНЕУ, 2003. 367 с.

43 Ткачук К. Н., Халімовський М. О., Зацарний В. В. Основи охорони праці : підручник. 2-ге вид., допов. та перероб. Київ : Основа, 2011. 480 с.

44 Гігієна праці : підручник / за ред. А. М. Шевченка. Київ : Інфотекс, 2000. 608 с.

45 Здоров’я та безпека на роботі в умовах воєнного стану: рекомендації роботодавцям щодо організації психосоціальної підтримки. [Електронний

ресурс]. – Режим доступу: https://pratsia.in.ua/faq.php (дата звернення: 14.06.2026).

46 Готович В. А., Ралік І. Р. Програмне забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі // Матеріали XI Міжнародної науково-практичної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“. – ТНТУ, 2022. – С. 126

47 Козак В. І., Готович В. А. Дослідження варіантів проектування інтерфейсу користувача в інформаційних інтерактивних аналітичних панелях // Матеріали XII Міжнародної науково-практичної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“. – ФОП Паляниця В. А., 2023. – С. 385–386

48 Готович В. А., Мачужак А. В. Застосування методології CI/CD для автоматизації процесів тестування та розгортання програмного забезпечення // XI Міжнародна науково-практична конференція молодих учених та студентів „Актуальні задачі сучасних технологій“, 7-8 грудня 2022 року. – Т. : ТНТУ, 2022. – С. 131–132. – (Комп’ютерно-інформаційні технології та системи зв’язку)

49 Готович В. А., Граб Д. В. Актуальність задачі розробки модуля інформаційної системи для управління ІТ-проєктами // Збірник тез доповідей XIII Міжнародної науково-практичної конференції молодих учених та студентів «АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ» – Тернопіль, 11-12 грудня 2024 року. с. 426-427

50 Гайдар А., Готович В. Розробка платформи для перевірки знань шляхом тестування // Матеріали IX науково-технічної конференції „Інформаційні моделі, системи та технології“. – ТНТУ, 2021. – С. 37

ДОДАТКИ

Додаток А

Код логіки веб-застосунку

Лістинг сервіс інтелектуального пошуку рецептів recipeService.js

```
import axios from 'axios';

export const translateIngredient = async (text) => {
    try {
        const response = await
axios.get(`https://api.mymemory.translated.net/get`, {
            params: { q: text, langpair: 'uk|en' }
        });
        return
response.data.responseData.translatedText.toLowerCase();
    } catch (error) {
        console.error("Помилка перекладу:", error);
        return null;
    }
};

export const findSmartRecipes = async (userProducts) => {
    const activeProducts = userProducts
        .filter(p => p.status === 'active')
        .sort((a, b) => new Date(a.expiry_date) - new
Date(b.expiry_date));

    if (activeProducts.length === 0) return [];

    let meals = [];

    for (let i = 0; i < Math.min(3, activeProducts.length); i++) {
        const engName = await
translateIngredient(activeProducts[i].name);
        if (!engName) continue;

        try {
            const res = await
axios.get(`https://www.themealdb.com/api/json/v1/1/filter.php?i=${
engName}`);
            if (res.data && res.data.meals) {
                // Отримуємо деталі перших 4 знайдених страв
                const mealDetails = await Promise.all(
                    res.data.meals.slice(0, 4).map(meal =>

axios.get(`https://www.themealdb.com/api/json/v1/1/lookup.php?i=${
meal.idMeal}`)
                    )
                );
                meals = [...meals, ...mealDetails.map(m =>
m.data.meals[0])];
```

```
            }
        } catch (error) {
            console.error("Помилка пошуку TheMealDB:", error);
        }
    }

    const analyzedRecipes = meals.map(meal => {
        let have = [];
        let missing = [];

        for (let i = 1; i <= 20; i++) {
            const ingredient = meal[`strIngredient${i}`];
            if (ingredient && ingredient.trim() !== '') {
                const isAvailable = activeProducts.some(p =>
p.name.toLowerCase().includes(ingredient.toLowerCase())
                );

                if (isAvailable) have.push(ingredient);
                else missing.push(ingredient);
            }
        }

        return {
            id: meal.idMeal,
            title: meal.strMeal,
            image: meal.strMealThumb,
            instructions: meal.strInstructions,
            ingredients: { have, missing }
        };
    });

    return analyzedRecipes;
};
```

Лістинг контролеру управління запасами та статистикою ProductController.php

```
    <?php

    namespace App\Http\Controllers;

    use App\Models\Product;
    use Illuminate\Http\Request;
    use Carbon\Carbon;

    class ProductController extends Controller
    {
        public function index(Request $request)
        {
            $products = Product::where('user_id', $request-
    >user()->id)
```

```
            ->orderBy('expiry_date', 'asc')
            ->get();
        $today = Carbon::today();
        foreach ($products as $product) {
            if ($product->status === 'active' &&
Carbon::parse($product->expiry_date)->lt($today)) {
                $product->update(['status' => 'spoiled']);
            }
        }

        return response()->json($products);
    }
    public function store(Request $request)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'category' => 'required|string',
            'quantity' => 'required|numeric|min:0.1',
            'unit' => 'required|string',
            'expiry_date' =>
'required|date|after_or_equal:today',
        ]);

        $product = $request->user()->products()->create([
            'name' => $validated['name'],
            'category' => $validated['category'],
            'quantity' => $validated['quantity'],
            'unit' => $validated['unit'],
            'expiry_date' => $validated['expiry_date'],
            'status' => 'active'
        ]);

        return response()->json($product, 201);
    }
    public function updateStatus(Request $request, $id)
    {
        $validated = $request->validate([
            'status' => 'required|in:eaten,thrown_away'
        ]);

        $product = Product::where('user_id', $request->user()-
>id)->findOrFail($id);
        $product->update(['status' => $validated['status']]);

        return response()->json(['message' => 'Статус успішно
оновлено', 'product' => $product]);
    }
    public function getStatistics(Request $request)
    {
        $userId = $request->user()->id;

        $eatenCount = Product::where('user_id', $userId)-
>where('status', 'eaten')->count();
```

```
            $thrownCount = Product::where('user_id', $userId)-
    >whereIn('status', ['thrown_away', 'spoiled'])->count();
            $totalProcessed = $eatenCount + $thrownCount;

            $efficiency = $totalProcessed > 0 ? round(($eatenCount
    / $totalProcessed) * 100) : 0;

            return response()->json([
                'eaten' => $eatenCount,
                'thrown_away' => $thrownCount,
                'efficiency_percent' => $efficiency
            ]);
        }
}
```

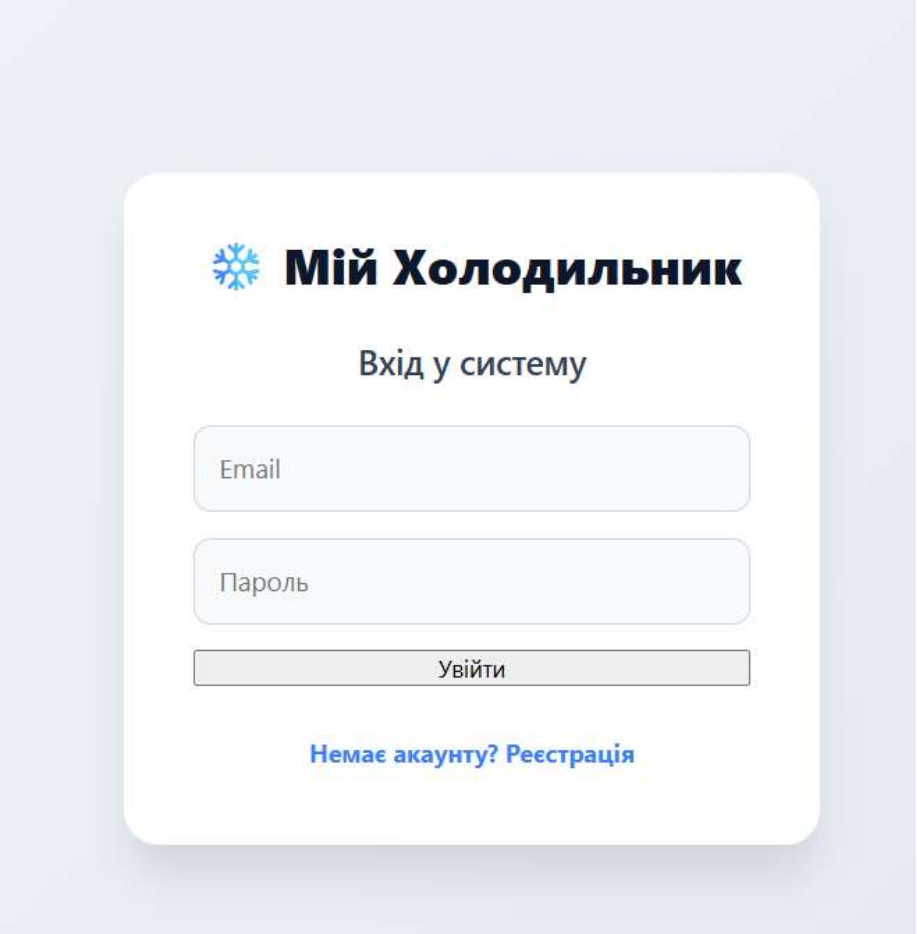

Рисунок 1.А – Інтерфейс екрану авторизації та реєстрації користувача

Лістинг контролеру авторизації та реєстрації користувачів

```
<?php

namespace App\Http\Controllers;
```

```
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\ValidationException;

class AuthController extends Controller
{
    public function register(Request $request)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
                                                    'email'       =>
    'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:8|confirmed',
        ]);

        $user = User::create([
            'name' => $validated['name'],
            'email' => $validated['email'],
            'password' => Hash::make($validated['password']),
        ]);

        $token = $user->createToken('auth_token')->plainTextToken;

        return response()->json([
            'message' => 'Реєстрація успішна',
            'access_token' => $token,
            'token_type' => 'Bearer',
            'user' => $user
        ], 201);
    }

    public function login(Request $request)
    {
        $request->validate([
            'email' => 'required|email',
            'password' => 'required',
        ]);

        $user = User::where('email', $request->email)->first();

          if (! $user || ! Hash::check($request->password, $user-
    >password)) {
            throw ValidationException::withMessages([
                'email' => ['Надані облікові дані є невірними.'],
            ]);
        }

        $user->tokens()->delete();

        $token = $user->createToken('auth_token')->plainTextToken;

        return response()->json([
```

```
            'message' => 'Авторизація успішна',
            'access_token' => $token,
            'token_type' => 'Bearer',
            'user' => $user
        ]);
    }

    public function logout(Request $request)
    {
        $request->user()->currentAccessToken()->delete();

        return response()->json([
            'message' => 'Успішний вихід із системи'
        ]);
    }
}
```

Додаток Б

Інформація про розроблену базу даних

Лістинг міграції створення таблиці продуктів

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::create('products', function (Blueprint $table) {
            $table->id();
                        $table->foreignId('user_id')->constrained()-
    >onDelete('cascade');

            $table->string('name');
            $table->string('category')->index();
            $table->decimal('quantity', 8, 2);
            $table->string('unit');
            $table->date('expiry_date');

                        $table->enum('status',   ['active',   'eaten',
    'thrown_away', 'spoiled'])->default('active');

            $table->timestamps();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('products');
    }
};
```

Лістинг моделі даних користувача із налаштуванням зв’язків ORM Eloquent

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
```

```
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
use Illuminate\Database\Eloquent\Relations\HasMany;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    protected $casts = [
        'email_verified_at' => 'datetime',
        'password' => 'hashed',
    ];

    public function products(): HasMany
    {
        return $this->hasMany(Product::class);
    }
}
```

Лістинг моделі даних продукту

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Product extends Model
{
    use HasFactory;

    protected $fillable = [
        'user_id',
        'name',
        'category',
        'quantity',
        'unit',
        'expiry_date',
```

```
        'status'
    ];

    public function user(): BelongsTo
    {
        return $this->belongsTo(User::class);
    }
}
```

Лістинг моделі даних продукту

```
CREATE TABLE `users` (
  `id` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `email` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `email_verified_at` timestamp NULL DEFAULT NULL,
  `password` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `remember_token` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT
   NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `users_email_unique` (`email`)
)          ENGINE=InnoDB          DEFAULT          CHARSET=utf8mb4
    COLLATE=utf8mb4_unicode_ci;

CREATE TABLE `products` (
  `id` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
  `user_id` bigint(20) UNSIGNED NOT NULL,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `category` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `quantity` decimal(8,2) NOT NULL,
  `unit` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `expiry_date` date NOT NULL,
   `status`  enum('active','eaten','thrown_away','spoiled')  COLLATE
    utf8mb4_unicode_ci NOT NULL DEFAULT 'active',
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `products_user_id_foreign` (`user_id`),
  KEY `products_category_index` (`category`),
  KEY `products_status_index` (`status`),
   CONSTRAINT  `products_user_id_foreign`  FOREIGN  KEY  (`user_id`)
    REFERENCES `users` (`id`) ON DELETE CASCADE
)          ENGINE=InnoDB          DEFAULT          CHARSET=utf8mb4
    COLLATE=utf8mb4_unicode_ci;

CREATE TABLE `personal_access_tokens` (
  `id` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
   `tokenable_type`  varchar(255)  COLLATE  utf8mb4_unicode_ci  NOT
    NULL,
  `tokenable_id` bigint(20) UNSIGNED NOT NULL,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
```

```
  `token` varchar(64) COLLATE utf8mb4_unicode_ci NOT NULL,
  `abilities` text COLLATE utf8mb4_unicode_ci,
  `last_used_at` timestamp NULL DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `personal_access_tokens_token_unique` (`token`),
   KEY  `personal_access_tokens_tokenable_type_tokenable_id_index`
    (`tokenable_type`,`tokenable_id`)
)          ENGINE=InnoDB          DEFAULT          CHARSET=utf8mb4
   COLLATE=utf8mb4_unicode_ci;
```