

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Вебдодаток кулінарних рецептів»

Виконала студентка групи ПЗ-20-1
ЧЕРНОВА Іванна Андріївна

Керівник роботи:
ЧИЖМОТРИЯ Олексій Володимирович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЧЕРНОВА Іванна Андріївна**

Керівник роботи: **ЧИЖМОТРЯ Олексій Володимирович**

Тема роботи: **«Вебдодаток кулінарних рецептів»**,

затверджена Наказом закладу вищої освіти від **«23» лютого 2024 р., №74/с**

Вихідні дані для роботи: **методи та засоби розробки вебдодатків, аналіз проблематики додатку кулінарних рецептів, використання для розробки вебдодатку вебфреймворку Express.JS та платформи Node.JS.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.В.	27.02.2024р.	27.02.2024р.
2	Чижмотря О.В.	27.02.2024р.	27.02.2024р.
3	Чижмотря О.В.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплексу додатка кулінарних рецептів і пояснювальної записки. Пояснювальна записка до випускної роботи містить 94 сторінки, 68 ілюстрацій, 17 таблиць, 30 використаних джерел та 19 сторінок додатків.

Метою роботи є розробка вебдодатку кулінарних рецептів засобами Node.js та Express.js, що дозволить шукати, зберігати та додавати кулінарні рецепти.

У роботі визначено основні завдання на розробку системи, також проаналізовано аналоги створеного програмного продукту та обрано засоби реалізації. Проведено розробку структури бази даних, обрано зручний та сучасний стек технологій та розроблено концепцію програмного продукту. Також описана програмна реалізація, її результати тестування.

КЛЮЧОВІ СЛОВА: ВЕБДОДАТОК, ОДНОСТОРИНКОВИЙ ДОДАТОК, РЕЦЕПТИ, КУЛІНАРІЯ, NODE.JS, EXPRESS.JS, REACT.

					ІПЗ.КР.Б – 121 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Чернова І.А.			Вебдодаток кулінарних рецептів. Пояснювальна записка	Літ.	Арк.	Аркушів
Керівник		Чижмотря О.В.					3	94
Рецензент		Єфремов О.В.				Житомирська політехніка, група ІПЗ-20-1		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's thesis consists of a software package of culinary recipes and an explanatory note. The explanatory note to the final thesis contains 94 pages, 68 illustrations, 17 tables, 30 references and 19 pages of appendices.

The purpose of the work is to develop a web application for cooking recipes using Node.js and Express.js, which will allow you to search, store and add cooking recipes.

The paper defines the main tasks for the development of the system, analyses analogues of the created software product and selects the means of implementation. The database structure was developed, a convenient and modern technology stack was selected, and the concept of the software product was developed. The software implementation and its testing results are also described.

KEYWORDS: WEB APPLICATION, SINGLE-PAGE APPLICATION, RECIPES, COOKING, NODE.JS, EXPRESS.JS, REACT.

					ИПЗ.КР.Б – 121 – 24 – ИПЗ	Арк.
						4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ВЕБДОДАТКУ КУЛІНАРНИХ РЕЦЕПТІВ.....	9
1.1 Постановка задачі.....	9
1.2 Аналіз аналогів програмного продукту	11
1.3 Вибір архітектури вебдодатку для перегляду рецептів	16
Висновки до першого розділу.....	18
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБДОДАТКУ КУЛІНАРНИХ РЕЦЕПТІВ	20
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи	20
2.2 Розробка бази даних системи.....	27
2.3 Проектування та реалізація алгоритмів роботи системи	31
2.4 Реалізація функціоналу вебдодатку кулінарних рецептів	36
2.4.1 Реалізація основної архітектури вебдодатку	37
2.4.2 Реалізація функціоналу додавання відгуків до рецепту.....	42
Висновки до другого розділу	44
3.1 Порядок встановлення та налаштування параметрів додатку	45
3.2 Структура інтерфейсу вебдодатку та порядок роботи.....	46
3.3 Тестування роботи вебдодатку.....	67
Висновки до третього розділу.....	70
ВИСНОВКИ.....	71
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	73
ДОДАТКИ	76

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс

SPA – односторінковий застосунок

SQL – мова структурованих запитів

БД – база даних

UML – уніфікова мова моделювання програм

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						6

ВСТУП

Актуальність теми полягає у тому, що у сучасному та мінливому світі технології визначають не лише спосіб, яким ми взаємодіємо з інформацією, але й те, як ми втілюємо в життя наші щоденні потреби. Інтернет вже давно став невід’ємною складовою частиною повсякденного життя людства, де є можливість знайти відповіді на питання, розважитись та використовувати різноманітні сервіси. Відповідно до даної буденності, створення вебдодатку, призначеного для перегляду кулінарних рецептів, стає важливим кроком, аби забезпечити зручність та насолоду під час приготування страв.

Вебдодатки дозволяють швидко і легко знайти необхідний рецепт, поділитися ним з іншими користувачами, а також зберегти улюблені страви для подальшого використання. Актуальність обраної теми полягає у тому, що сучасні тенденції у галузі інформаційних технологій, такі як розробка вебзастосунків, сприяють підвищенню ефективності та зручності доступу до інформації у різних сферах діяльності, зокрема у кулінарії.

Метою випускної кваліфікаційної роботи є розробка вебдодатку для зберігання, організації та пошуку кулінарних рецептів.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Провести аналіз існуючих вебдодатків для кулінарних рецептів та визначити їх переваги та недоліки.
2. Розробити архітектуру вебдодатку, що забезпечить його функціональність та зручність у використанні.
3. Вибрати відповідні технології для реалізації вебдодатку.
4. Розробити інтерфейс користувача, що забезпечить зручний доступ до функціональних можливостей додатку.
5. Реалізувати функціональність вебдодатку, включаючи зберігання, організацію та пошук рецептів.
6. Реалізувати панель адміністратора для управління даними, якими наповнений вебсайт.
7. Провести тестування вебдодатку та усунути виявлені недоліки.

Об'єктом дослідження є процеси програмування, проєктування та впровадження вебдодатку кулінарних рецептів.

Предметом дослідження є методи та технології розробки вебдодатку для кулінарних рецептів, що забезпечить зручність використання.

Методи аналізу включають методи системного аналізу, об'єктно-орієнтованого проєктування, веб-програмування та дизайну. Інформаційна база дослідження складається з наукових статей, публікацій, а також існуючих вебдодатків для кулінарних рецептів.

Практичне значення створеного вебдодатку полягає у можливості використанні його для зберігання, організації та пошуку кулінарних рецептів. Вебдодаток може бути корисним як для індивідуальних кулінарів, так і для кулінарних спільнот, ресторанів та кулінарних шкіл з можливістю вдосконалення необхідного функціоналу.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ВЕБДОДАТКУ КУЛІНАРНИХ РЕЦЕПТІВ

1.1 Постановка задачі

Кожен з нас готує щодня. Комуś це вдається краще, іншим гірше. Проте, доступ до Інтернету з телефону, планшету чи ноутбуку робить можливим дізнатись новий рецепт на кулінарних веб-сайтах. Вони допомагають пришвидшити навчання кулінарії чи знайти нові смачні страви. Такі веб-додатки є значно зручнішими відносно кулінарних книг чи записників. Книгу можна загубити чи забути вдома, а веб-додаток легко відкрити у браузері смартфона.

Крім того, веб-додатки для кулінарних рецептів надають можливість експериментувати зі смаками та інгредієнтами. Вони можуть запропонувати вам рецепти з нестандартними поєднаннями, які ви, можливо, навіть не подумали спробувати. Таким чином, кожне приготування стає невеликим кулінарним викликом, який додає вам нові знання та задоволення від процесу готування.

Більше того, веб-додатки можуть функціонувати як освітні ресурси, де ви можете вивчати нові техніки готування, ознайомлюватися зі світовою кулінарією та вдосконалювати свої навички. Наприклад, деякі додатки можуть надавати відеоуроки з приготування, де ви можете побачити кожен крок процесу та отримати корисні поради від досвідчених шеф-кухарів. Така можливість навчання робить веб-додатки не лише корисними інструментами для готування, а й цінними джерелами знань у світі кулінарії.

За даними досліджень, 81% американців використовують Інтернет для пошуку рецептів та кулінарних порад. Вебдодатки для кулінарних рецептів дозволяють не лише зберігати рецепти у цифровому форматі, але й легко ділитися ними з іншими користувачами, що сприяє популяризації кулінарії та обміну культурними традиціями [1].

Основною метою створення вебдодатку для перегляду рецептів є спрощення щоденного приготування страв та внесення різноманіття у смаки. Реалізація даного

вебдодатку буде корисною людям, що прагнуть навчитись готувати та тим, хто потребує нових поєднань смаків.

У кваліфікаційній роботі необхідно створити вебдодаток, який буде корисний як для кулінарів, так і для гурманів. Додаток повинен надавати можливість легко знайти рецепт за певними критеріями, а також додавати та пропонувати власні рецепти.

Для реалізації поставленого завдання, основними етапами є:

1. Визначення вимог та функцій веб-додатку;
2. Аналіз схожих веб-додатків;
3. Створення цілісного дизайну вебдодатку:
 - a. Визначення загальної ідеї та ключових принципів дизайну;
 - b. Розробка окремих UI компонентів;
 - c. Створення дизайну для необхідних сторінок;
 - d. Забезпечення адаптивності щодо розмірів екранів та пристроїв;
4. Проєктування та розробка архітектури додатку:
 - a. Створення архітектури серверної частини вебдодатку;
 - b. Створення архітектури клієнтської частини вебдодатку;
5. Розробка основних компонентів, сторінок та функціоналу додатку:
 - a. Сторінки вебдодатку: головна, кухні світу, рецепти, окремий рецепт, про нас, вхід та реєстрація, кабінет авторизованого користувача, публічний профіль користувача;
 - b. Сторінки у панелі адміністратора: керування користувачами , кухнями світу, рецептами, коментарями;
 - c. Функціонал коректного зберігання інформації в базі даних;
 - d. Бекенд для взаємодії між клієнтською та серверною частинами.
 - e. Компоненти реєстрації та автентифікації користувачів;
 - f. Функціонал додавання та пропонування рецептів;
 - g. Функціонал відгуків для рецептів;
 - h. Функціонал пошуку рецептів;
 - i. Функціонал фільтрації та сортування рецептів;

j. Інтеграція бекенду у клієнтську частину додатку.

6. Тестування роботи додатку.

Результатом реалізації поставленого завдання є додаток для перегляду рецептів, що надає легкий та зручний доступ до рецептів, завдяки пошуку, сортуванню та фільтрації. Також, даний вебдодаток надає можливість поділитись власними рецептами та оцінити інші.

1.2 Аналіз аналогів програмного продукту

Для того щоб створити практичний та приємний у використанні продукт, необхідно було провести дослідження схожих вебдодатків кулінарних рецептів. Дане завдання включає загальний огляд дизайну, функціоналу вебсайту та зручності його застосування для кулінарних потреб. Таким чином, було легко визначити їх позитивні та негативні аспекти.

Для людей, що полюбляють готувати, завжди було важливо легко знайти певну чи нову для себе страву за вказаними критеріями. І при цьому мати доступ до рецепту, як і в книзі, чіткого та покрокового. Дані властивості притаманні усім далі розглянутим веб-додаткам, які при цьому мають дизайн, що кардинально відрізняється.

Першим аналогом розробленого додатку є кулінарний вебсайт «Клопотенко» (рис. 1.1) [2].

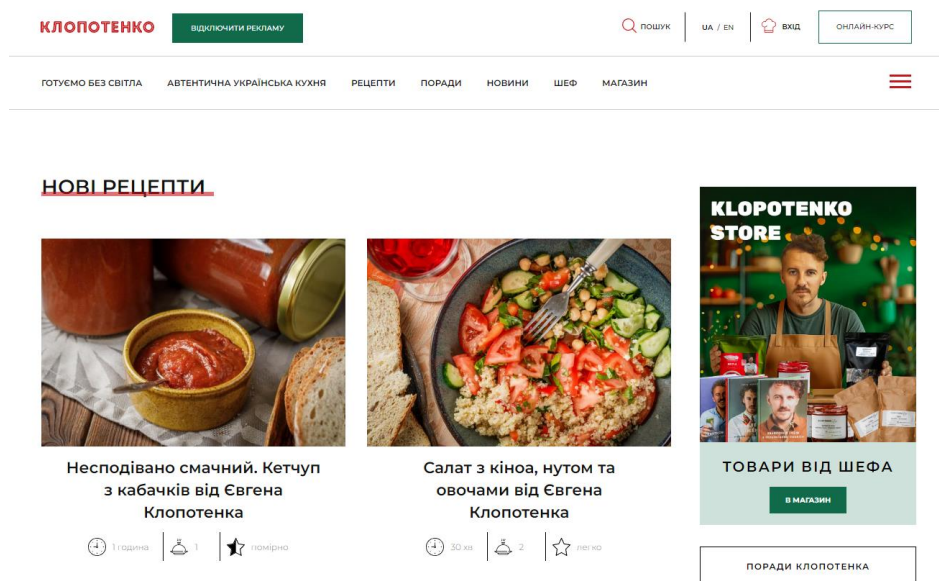


Рис. 1.1. Вебсайт «Клопотенко»

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						11

Даний сайт є одним з найпопулярніших сайтів з рецептами в Україні та містить авторські рецепти українського шеф-кухаря Євгенія Клопотенко. Він має мінімалістичний дизайн з акцентами червоного, що дозволяє сконцентруватись на головному – рецептах.

Основними перевагами вебсайту є:

- різноманітність категорій для перегляду рецептів;
- пошук за ключовими словами та назвою страви;
- система відгуків до рецептів та рейтингу, заснованого на вподобайках;
- після входу в обліковий запис можливо зберігати рецепти;
- фільтрація рецептів за інгредієнтами, категорією, кухнею та складністю.

Основними ж недоліками є:

- відсутність сортування рецептів;
- фільтрація надто складна у використанні, бо містить багато найменувань;
- відсутність сторінки з кухнями світу, за якою зручно було б переглядати рецепти.

Перейдемо до наступного аналогу. Allrecipes – сайт, орієнтований на приготування їжі, створений кулінарами для кулінарів, який дозволяє користувачам зберігати та обмінюватися рецептами. Даний вебсайт є популярним в США та має багато постійних користувачів (рис. 1.2) [3].

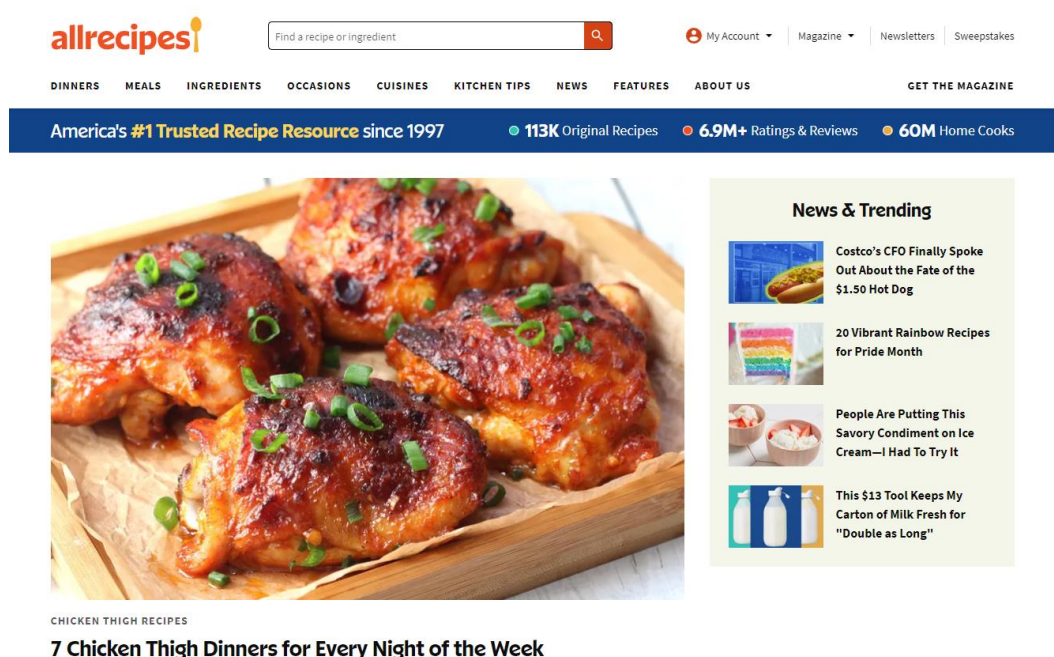


Рис. 1.2. Зовнішній вигляд сайту «Allrecipes»

					ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
						12

Дизайн вебдодатку має чіткі лінії та гармонійне поєднання білого, зеленого та помаранчевих акцентів. Інтерфейс досить зрозумілий, проте в деяких випадках містить надто багато елементів. Мова інтерфейсу – англійська.

Основні переваги вебсайту :

- велика кількість різних категорій рецептів;
- після авторизації можливо зберігати вподобані рецепти, створювати власні (приватні, публічні) та пропонувати їх для розміщення на вебсайті;
- присутній рейтинг рецепту, заснований на оцінці у відгуках.
- пошук за назвою чи певним інгредієнтом;
- окрема сторінка для кухонь світу.

Основні недоліки:

- перенасиченість інформацією;
- відсутність сортування та фільтрації;
- після додавання приватного чи публічного рецепту, його неможливо відредагувати;

«Epicurious» – це популярний кулінарний сайт з підпискою, що надає рецепти, кулінарні поради, статті та відеоінструкції. Він орієнтований на людей, які люблять готувати, пропонуючи різноманітний контент для покращення кулінарних навичок і натхнення (рис. 1.3) [4].

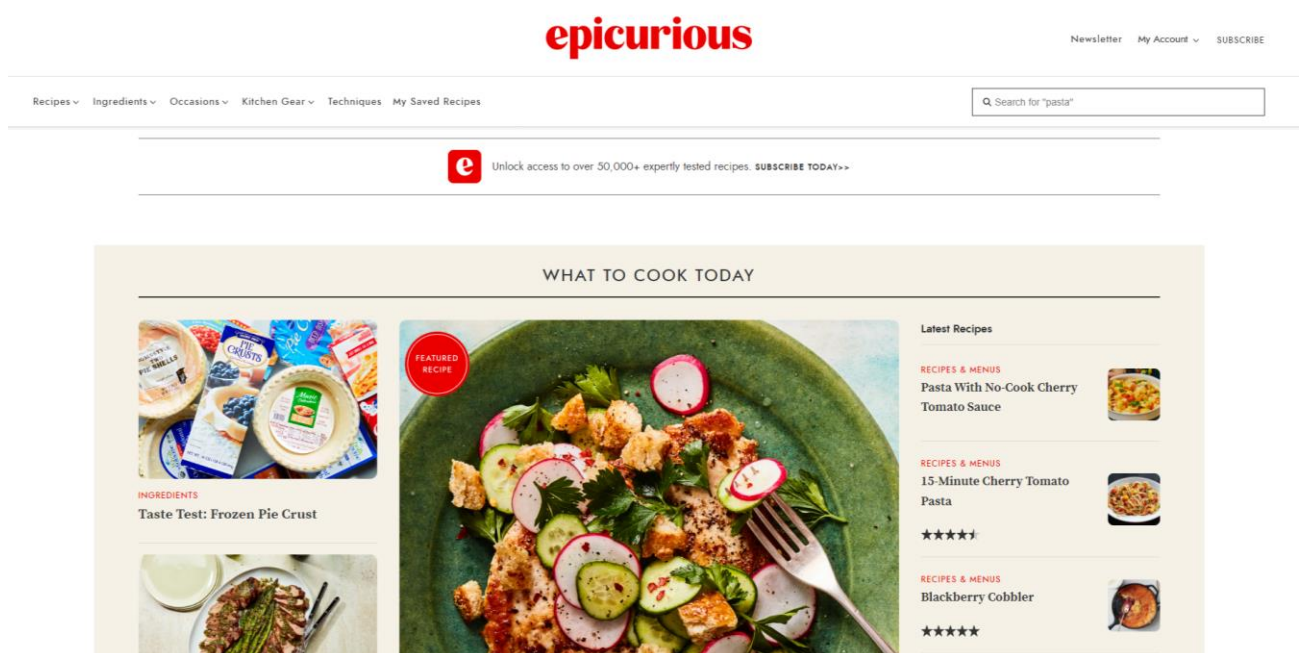


Рис. 1.3. Інтерфейс сайту «Epicurious»

Основними перевагами вебсайту є:

- великий асортимент рецептів різних кухонь і рівнів складності;
- зручні функції для підписників, що включають збереження рецептів, генератор списку покупок та голосове керування для роботи без рук.
- інтуїтивний інтерфейс та потужні фільтри;
- якісне наповнення та фотографії, приємний та мінімалістичний дизайн.

Основними недоліками є:

- платна підписка необхідна для використання, так як існує ліміт безкоштовних рецептів та статей;
- недосконалий рейтинг, бо відгуки можуть написати лише підписники;
- деякі рецепти на сайті можуть бути застарілими і не мати актуальних оновлень чи коригувань;

«Cookpad» – це міжнародна платформа для обміну рецептами, де користувачі можуть створювати, публікувати та зберігати свої кулінарні рецепти. Вона дозволяє кулінарам ділитися своїми ідеями та знаходити натхнення від інших користувачів (рис. 1.4) [5].

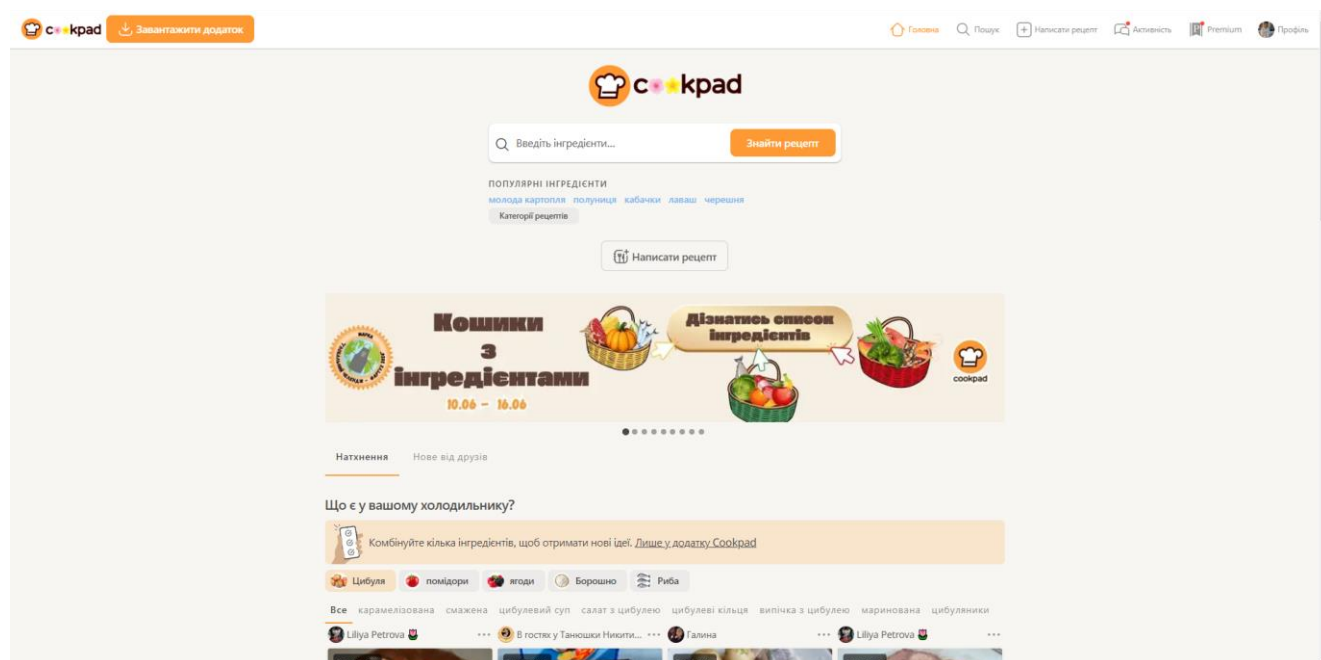


Рис. 1.4. Вебсайт «cookpad»

Основними перевагами вебсайту є:

- велика кількість активних користувачів, що забезпечує постійний потік нових рецептів і кулінарних ідей;

- зручний та простий у використанні інтерфейс, який дозволяє легко публікувати та знаходити рецепти;
- можливість шукати рецепти за конкретними інгредієнтами;
- функціонал коментування рецептів, слідкування за іншими користувачами та обміну повідомленнями.

Основними ж недоліками є:

- для авторизованих користувачів відсутня чітка система рейтингу, є тільки емоції та коментарі, а перевірені рецепти доступні за підпискою;
- недостатня суворість перевірки рецептів може призвести до публікації рецептів з неповною або некоректною інформацією;
- обмежені можливості для збереження та пошуку рецептів без підписки.

Для зручного порівняння вище описаних додатків була створена таблиця 1.1.

Таблиця 1.1

Порівняння аналогів веб-додатку

Характеристика	Klopotenko	Allrecipes	Epicurious	Cookpad
Пошук	Так	Так	Так	Так
Фільтрація рецептів	Так	Ні	Так	Так
Сортування рецептів	Ні	Ні	Так	Так
Збереження рецептів	Так	Так	Так	Так
Пропонування або додавання власного рецепту	Ні	Так	Ні	Так
Зручність використання	Висока	Середня	Висока	Середня
Достатній функціонал без підписки	Так	Так	Ні	Так
Дизайн інтерфейсу	Мінімалізм та акценти червоного	Яскравий, приємний для перегляду	Інтуїтивний мінімалізм	Простий та зрозумілий
Адаптивність	Так	Так	Так	Так

Аналіз чотирьох кулінарних вебсайтів показує, що вони мають як спільні риси, так і унікальні особливості, які впливають на їхню привабливість та зручність

для користувачів. Огляд дизайну, функціоналу та зручності використання допоміг визначити як позитивні, так і негативні аспекти кожного з них.

В цілому, кожен сайт надає можливість знайти рецепт за назвою чи інгредієнтом, відфільтрувати чи відсортувати рецепти. Два з них дають можливість додати чи запропонувати рецепт, що зручно для любителів та професіоналів у кулінарії. Також всі перелічені сайти концентрують увагу на змісті, тобто рецептах, та адаптуються в розмірі відповідно до пристрою.

Для тих, хто любить готувати, важливо мати можливість легко знаходити нові рецепти та отримувати чіткі, покрокові інструкції. Всі розглянуті вебсайти відповідають цим вимогам, хоча підхід до дизайну та функціоналу у кожного свій.

Отже, щоб покращити веб-додаток для перегляду рецептів, важливо удосконалити функціонал, зокрема фільтрації, сортування, а також дизайну та структурованості інтерфейсу. Це допоможе користувачам легко та зручно отримувати доступ до наявних рецептів.

1.3 Вибір архітектури вебдодатку для перегляду рецептів

Вебдодаток, що буде побудований на клієнт-серверній архітектурі, буде використовувати сучасні технології для забезпечення ефективної та надійної роботи. Серверна частина додатку буде реалізована у вигляді API за допомогою платформи Node.js та фреймворку Express.js, який забезпечує швидке та ефективне оброблення запитів від клієнтської частини. Фронтенд буде побудований з використанням фреймворку React.js, що дозволить створити інтерактивний та зручний інтерфейс для користувачів.

Архітектура клієнт-сервер є системою, яка розміщує, доставляє та керує більшістю ресурсів і послуг, які запитує клієнт. У цій моделі всі запити і послуги передаються через мережу, також відома як мережева модель обчислень або клієнт-серверна мережа [6].

Node.js є безкоштовним, відкритим, кросплатформним середовище виконання JavaScript, яке дозволяє розробникам створювати сервери, веб-додатки, інструменти командного рядка та скрипти [7].

Express.js є одним з найпопулярніших фреймворків для створення серверних додатків на Node.js. Він відомий своєю простотою та гнучкістю, що робить його ідеальним вибором для будь-якого вебдодатку. Завдяки широкому спектру плагінів і модулів, Express.js дозволяє легко розширювати функціональність додатку згідно з потребами проєкту [8].

React, у свою чергу, є потужним інструментом для створення інтерфейсів користувача. Він дозволяє розробникам будувати компоненти, які можна повторно використовувати та підтримувати, що спрощує процес створення складних інтерфейсів. Крім того, React забезпечує ефективне управління станом додатку, що дозволяє створювати інтерактивні та відповідні на події додатки [9].

MySQL є одним з найпоширеніших та надійних систем управління базами даних у світі. Вона відома своєю високою продуктивністю, швидкодією та розширюваністю. MySQL використовує стандартну мову SQL для роботи з даними, що дозволяє розробникам ефективно взаємодіяти з базою даних [10].

Основні переваги MySQL включають високий рівень надійності та безпеки, швидкодію та простоту використання. Вона підтримує широкий набір функцій, таких як транзакції, індексація та реплікація, що робить її ідеальним вибором для будь-якого проєкту.

Основною метою розробки вебдодатку є створення зручного та функціонального інтерфейсу для користувачів, що дозволить їм легко знаходити, зберігати та ділитися рецептами. Крім того, додаток буде мати можливість пошуку рецептів за різними критеріями, такими як вид страви, інгредієнти, кухня світу тощо.

Додатковою функціональністю буде можливість реєстрації користувачів та створення власних рецептів. Після реєстрації користувач зможе створювати власні рецепти, редагувати та видаляти їх, а також ділитися ними з іншими користувачами. Перед додаванням рецепту в систему власний користувач зможе перевірити його на відповідність вимогам адміністратора. Така система забезпечить якість контенту та запобігатиме додаванню неправдивої чи некоректної інформації.

Вибір фреймворку для серверної частини поставав між Express.js та Next.js.

Next.js – це React-фреймворк для створення веб-застосунків з функціями серверного рендерингу та статичної генерації сайтів. Він побудований на Node.js і розширює можливості React, але не є загальним Node.js фреймворком. Розроблений компанією Vercel, Next.js спрощує розробку та розгортання високопродуктивних веб-застосунків [11].

Вибір Express.js замість Next.js для створення серверної частини вебдодатку був обумовлений особливостями проєкту. Express.js є легковаговим та гнучким фреймворком, який дозволяє швидко розробляти серверні додатки на Node.js. Він ідеально підходить для створення API, що буде використовуватись клієнтською частиною додатку [12].

З іншого боку, Next.js зазвичай використовується для створення універсальних JavaScript-додатків, які включають як клієнтську, так і серверну частини. Однак, у випадку, коли не потрібно генерувати статичні HTML-сторінки на сервері або виконувати серверний рендеринг, використання Express.js може бути більш відповідним варіантом, оскільки воно дозволяє працювати з API більш прямолінійно та ефективно.

Отже, створення вебдодатку для пошуку та обміну рецептами є цікавим та корисним завданням. Використання сучасних технологій таких як Express.js, React та MySQL дозволить створити швидкий, надійний та зручний інструмент для всіх любителів кулінарії.

Висновки до першого розділу

У першому розділі було проведено аналіз предметної області, спрямований на визначення ключових вимог до вебдодатку кулінарних рецептів. Ця аналітична робота відобразила різноманітність функціоналу та можливостей існуючих платформ, що надають користувачам можливість зберігати, шукати та ділитися рецептами.

Було виконано детальний порівняльний аналіз кулінарних вебдодатків, було визначено різні принципи їхньої роботи, включаючи зручний пошук рецептів,

збереження улюблених, а також можливість додавання власних рецептів та обміну ними з іншими користувачами. Важливою частиною аналізу було встановлення основних вимог до майбутнього вебдодатку, який має надавати користувачам інтуїтивний та зручний інтерфейс для пошуку, збереження та додавання рецептів.

З урахуванням отриманих вимог та критеріїв, був складений перелік основних технологій та інструментів, що будуть використовуватися під час розробки вебдодатку кулінарних рецептів. Вибір цих технологій базується на їхній здатності забезпечити високу продуктивність, надійність та зручність використання для кінцевих користувачів.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБДОДАТКУ КУЛІНАРНИХ РЕЦЕПТІВ

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Вебдодаток кулінарних рецептів має інтерфейс, завдяки якому користувачі можуть переглядати всі головні сторінки, шукати, сортувати та фільтрувати рецепти. Існує два типи користувачів: гість та користувач (авторизований користувач). Гість може робити тільки раніше перераховане та перегляд профілів. Користувач, у свою чергу, може зберегти рецепти, писати відгуки та додавати рецепти, редагувати їх. Рецепти містять зображення, опис, список інгредієнтів та кроки виконання з можливим додаванням зображень. Адміністратор має доступ до дій користувача та до панелі адміністратора. Для розуміння взаємодії між елементами додатку було створено Діаграму варіантів використання (рис. 2.1).

Діаграма варіантів використання (Use Case Diagram) – це графічне зображення взаємодії користувачів із системою, яке показує всі можливі сценарії використання. Вона є компонентом мови моделювання UML. Ця діаграма ілюструє, як різні типи користувачів (актори) взаємодіють із системою через різні функції (варіанти використання). Вона допомагає зрозуміти вимоги до системи, ідентифікувати її функціональні можливості, а також служить основою для розробки більш детальних діаграм та модулів системи [13].

UML (Unified Modeling Language) – це стандартизована мова моделювання, яка використовується для візуалізації, специфікації, конструювання та документування компонентів програмного забезпечення. UML забезпечує набір графічних позначень для створення діаграм, які відображають різні аспекти програмних систем, включаючи структуру, поведінку та взаємодію між компонентами. Вона широко використовується в об'єктно-орієнтованому програмуванні для розробки та аналізу системи на різних етапах її життєвого циклу [14].

Таблиця 2.3

Опис прецеденту «Пропонування рецепту»

Назва варіанту використання	Додавання рецепту
Основні актори	Адміністратор, користувач
Короткий опис	Якщо користувач надає рецепту статус «запропонований», тоді адміністратор може перевірити його в панелі адміністратора та затвердити чи відхилити.

Таблиця 2.4

Опис прецеденту «Пошук рецептів»

Назва варіанту використання	Пошук рецептів
Основні актори	Адміністратор, користувач, гість
Короткий опис	У вебдодатку можна здійснювати пошук рецепту за назвою, інгредієнтом та списком інгредієнтів.

Таблиця 2.5

Опис прецеденту «Фільтрація та сортування рецептів»

Назва варіанту використання	Фільтрація та сортування рецептів
Основні актори	Адміністратор, користувач, гість
Короткий опис	Додаток надає можливість одночасно фільтрувати та сортувати рецепти за певними критеріями.

Таблиця 2.6

Опис прецеденту «Редагування особистих даних»

Назва варіанту використання	Редагування особистих даних
Основні актори	Адміністратор, користувач
Короткий опис	Користувач у власному кабінеті може редагувати інформацію про себе.

Таблиця 2.7

Опис прецеденту «Керування вебдодатком»

Назва варіанту використання	Керування вебдодатком
Основні актори	Адміністратор
Короткий опис	У панелі адміністратора можна виконувати такі дії: редагувати та видаляти користувачів; створювати, редагувати та видаляти кухні світу; редагувати, видаляти рецепти; затверджувати чи відхиляти запропоновані рецепти; затверджувати або видаляти відгуки користувачів.

Основна ціль розробки вебдодатку кулінарних рецептів полягає в створенні простого та зручного інструменту для пошуку, додавання, редагування та обміну рецептами. Він забезпечує широкий доступ до різноманітних рецептів та функціональність пошуку та фільтрації за різними критеріями. Користувачі можуть створювати власні рецепти, ділитися ними та залишати відгуки. Персоналізований досвід забезпечує налаштування власних профілів, зі створеними рецептами та зберігання улюблених. Адміністраторам доступні функції керування контентом, включаючи затвердження чи відміну запропонованих рецептів та управління відгуками.

Основною метою є полегшення процесу приготування рецептів і відкриття нових смакових вражень для користувачів, надаючи їм зручний інструмент для цього.

Вимоги користувачів

Вимоги гостей:

1. Можливість перегляду всіх основних сторінок додатку, включаючи кухні світу, рецепти, рецепт, пошук та публічні профілі зареєстрованих користувачів;
2. Можливість пошуку рецептів;
3. Можливість фільтрації та сортування рецептів;
4. Можливість реєстрації та входу у обліковий запис.

Вимоги авторизованих користувачів:

1. Всі можливості гостей;
2. Можливість додавання рецептів в збережені;
3. Можливість написання відгуків під рецептами;
4. Можливість перегляду та редагування власного облікового запису;
5. Можливість додавати власні рецепти та надавати їм різну видимість на сайті, а також запропонувати свій рецепт. Якщо адміністратор його затвердить, то даний рецепт з'явиться на сайті.

Вимоги адміністраторів:

1. Доступ до панелі адміністратора;

2. Можливість керування користувачами;
3. Можливість керування рецептами, а також затвердження запропонованих рецептів;
4. Можливість керування коментарями (затвердження, видалення).

Характеристика об'єкта комп'ютеризації

Користувач додатку буде мати можливість переглядати рецепти, зберігати вподобані. Крім того, зможе шукати рецепти та переглядати їх за прийомом їжі та різноманітними кухнями світу, писати коментарі під ними. Все це буде зручно робити завдяки зрозумілому та простому інтерфейсу додатку.

Функціональні вимоги:

- Авторизація у додатку: в системі повинна бути представлена можливість реєстрації користувача та присвоєння йому відповідної ролі (користувач, адміністратор).
- Можливість збереження інформації: додаток повинен зберігати інформацію і надавати можливість користувачу переглядати її, а адміністраторам керувати нею.
- Адаптивність: користувацька частина веб-додатку повинна бути реалізована для зручного користування зі смартфонів, планшетів та персональних комп'ютерів.

Нефункціональні вимоги:

- Сприйняття
 - час, необхідний для навчання адміністраторів – 30 хвилин, а для розуміння користувачем – 3 хвилини.
 - час відгуку для типових задач – не більше 5 секунд, для складних завдань – не більше 10 секунд.
 - інтерфейс має бути мінімалістичним, інтуїтивним та зручним.
- Надійність
 - доступність – час, що витрачається на обслуговування системи не повинно перевищувати 6% від загального часу роботи.
 - середній час безвідмовної роботи – 15 робочих днів.

- Продуктивність
 - система повинна підтримувати мінімум 100 користувачів, що переглядають веб-сайт, 5 одночасно працюючих адміністраторів, пов'язаних із загальною базою даних.
- Можливість експлуатації
 - масштабування – система повинна мати можливість збільшувати потужності (продуктивність), зі збільшенням користувачів та адміністраторів таким чином, щоб це не вплинуло негативно на її роботу.

Системні вимоги:

1. Вимоги до середовища виконання:

- Мінімум 2 Гб оперативної пам'яті.
- Процесор з тактовою частотою не менше 1.6 ГГц.
- Наявність доступу до мережі Інтернет.

2. Вимоги до серверу інформаційної системи:

- Наявність реляційної бази даних для забезпечення правильних зв'язків між таблицями та зберігання даних відповідно до їхньої структури.
- Підтримка технологій Node.js на сервері для реалізації вебдодатку.

Завдяки аналізу функціональних вимог стало можливо виокремити наступні сутності, які відображають загальну логіку роботи програмного продукту та надають можливість зрозуміти зв'язки як між основними компонентами клієнтської частини, так і між таблицями бази даних, які будуть далі розроблені.

Перейдемо до опису діаграми класів, їх полів та методів [15].

Клас **User** відповідає за усі персональні данні, що належать користувачу: нікнейм, ім'я, прізвище і т.д.

Клас **Admin** наслідує User та додає методи для керування іншими класами в панелі адміністратора.

Клас переліку **RoleName** відповідає за роль користувача: звичайний користувач чи адміністратор.

2.2 Розробка бази даних системи

Для збереження інформації у вебдодатку була використана СУБД MySQL, що була обрана за надійність, швидкодію та підтримку стандартної мови SQL. Для створення таблиць бази даних був використаний MySQL Workbench. Дане середовище є уніфікованим візуальним інструментом для безпосередньо візуального проектування, моделювання та створення баз даних [16].

База даних для вебдодатку містить дев'ять таблиць:

1. users – таблиця для збереження даних користувачів;
2. cuisines – таблиця для забезпечення збереження даних про кухні світу;
3. recipes – таблиця для збереження основних даних про рецепт;
4. ingredients – таблиця для забезпечення збереження інгредієнтів рецепту;
5. directions – таблиця для збереження кроків виконання рецепту;
6. reviews – таблиця для забезпечення збереження відгуків на рецепт;
7. categories – таблиця для збереження даних про категорії рецептів;
8. categories_recipes – таблиця для забезпечення збереження категорій відповідно до рецепту, а також рецептів до категорій;
9. saved_recipes – таблиця, що забезпечує збереження рецептів користувачем.

Опис структури таблиць бази даних вебдодатку кулінарних рецептів наведено у таблицях 2.8 – 2.16.

Для зберігання даних про користувачів та адміністраторів використовується таблиця «user». У дану таблицю вводяться усі дані про користувача. До них належать логін, електронна пошта, пароль, прізвище та ім'я, фото профілю, роль, кулінарний рівень, а також опис про користувача у профілі. Зареєстрованому користувачу надається роль 0, що означає роль користувача.

Таблиця 2.8

Структура таблиці «users»

Назва	Тип даних	ПК	ЗК	Опис поля
1	2	3	4	5
id	int	+	-	Код користувача

1	2	3	4	5
username	varchar (60)	-	-	Логін
email	varchar (60)	-	-	Електронна пошта
password	varchar (200)	-	-	Пароль
name	varchar (60)	-	-	Прізвище та ім'я
profilePic	varchar (300)	-	-	Назва фото користувача
isAdmin	tinyint (1)	-	-	Роль(користувач чи адмін)
level	enum ('Amateur', 'Home Cook', 'Expert')	-	-	Рівень (аматор, любитель, експерт)
about	varchar (200)	-	-	Опис користувача

У таблиці «cuisines» зберігаються всі дані про кухню світу.

Таблиця 2.9

Структура таблиці «cuisines»

Назва	Тип даних	ПК	ЗК	Опис поля
id	int	+	-	Код кухні світу
name	varchar (100)	-	-	Назва
text	varchar (500)	-	-	Опис
img	varchar (500)	-	-	Назва фото кухні світу

Таблиця «recipes» достатньо велика та зберігає всі дані про рецепт.

Таблиця 2.10

Структура таблиці «recipes»

Назва	Тип даних	ПК	ЗК	Опис поля
1	2	3	4	5
id	int	+	-	Код рецепту
title	varchar (100)	-	-	Назва
userId	int	-	+	Код користувача
createdAt	datetime	-	-	Дата та час створення
updatedAt	datetime	-	-	Дата та час оновлення
confirmedAt	datetime	-	-	Дата та час затвердження
cuisineId	int	-	+	Код кухні світу
currentRating	float	-	-	Поточний рейтинг

1	2	3	4	5
cookTime	int	-	-	Час приготування в хвилинах
servings	varchar (45)	-	-	Кількість порцій
difficulty	varchar (10)	-	-	Складність рецепту
intro	varchar (400)	-	-	Вступ рецепту
status	enum ('private', 'public', 'submitted', 'rejected', 'approved')	-	-	Статус рецепту (приватний, публічний, запропонований, відхилений, затверджений)
img	varchar (500)	-	-	Назва зображення рецепту
userName	varchar (60)	-	-	Ім'я користувача що створив рецепт
addInfo	varchar (400)	-	-	Додаткова інформація
tmeasure	varchar (45)	-	-	Одиниця вимірювання часу

У таблиці «ingredients» зберігаються інгредієнти, відповідно до рецепту.

Таблиця 2.11

Структура таблиці «ingredients»

Назва	Тип даних	ПК	ЗК	Опис поля
id	int	+	-	Код інгредієнту
recipeId	int	-	+	Код рецепту
text	varchar (500)	-	-	Текст інгредієнту

У таблиці «directions» зберігаються дані про кроки, також відповідно до рецепту.

Таблиця 2.12

Структура таблиці «directions»

Назва	Тип даних	ПК	ЗК	Опис поля
id	int	+	-	Код кроку
text	varchar (500)	-	-	Текст кроку
img	varchar (500)	-	-	Назва фото до кроку
recipeId	int	-	+	Код рецепту
orderNum	int	-	-	Черга виконання

У таблиці «reviews» зберігаються відгуки до рецептів.

Таблиця 2.13

Структура таблиці «reviews»

Назва	Тип даних	ПК	ЗК	Опис поля
id	int	+	-	Код відгуку
userId	int	-	+	Код користувача
recipeId	int	-	+	Код рецепту
rating	int	-	-	Оцінка
text	varchar (500)	-	-	Текст відгуку
createdAt	datetime	-	-	Дата і час створення
img	varchar (500)	-	-	Назва фото до відгуку
status	enum ('submitted', 'approved')	-	-	Статус (запропонований, затверджений)

Таблиця «categories» зберігає дані про категорії рецептів.

Таблиця 2.14

Структура таблиці «categories»

Назва	Тип даних	ПК	ЗК	Опис поля
id	int	+	-	Код кухні світу
name	varchar(100)	-	-	Назва

У таблиці «categories_recipes» здійснюється встановлення зв'язку «багато до багатьох» між категоріями та рецептами. До рецепту може належати багато категорій, а до категорії рецептів.

Таблиця 2.15

Структура таблиці «categories_recipes»

Назва	Тип даних	ПК	ЗК	Опис поля
id	int	+	-	Код кухні світу
categoryId	int	-	+	Код категорії
recipeId	int	-	+	Код рецепту

У таблиці «saved_recipes» зберігаються дані про те які рецепти зберіг користувач.

Структура таблиці «saved_recipes»

Назва	Тип даних	ПК	ЗК	Опис поля
id	int	+	-	Код кухні світу
userId	int	-	+	Код користувача
recipeId	int	-	+	Код рецепту

На рисунку 2.3. наведено схему бази даних. Дана візуалізація демонструє, яким чином виглядають таблиці та зв'язки між ними.

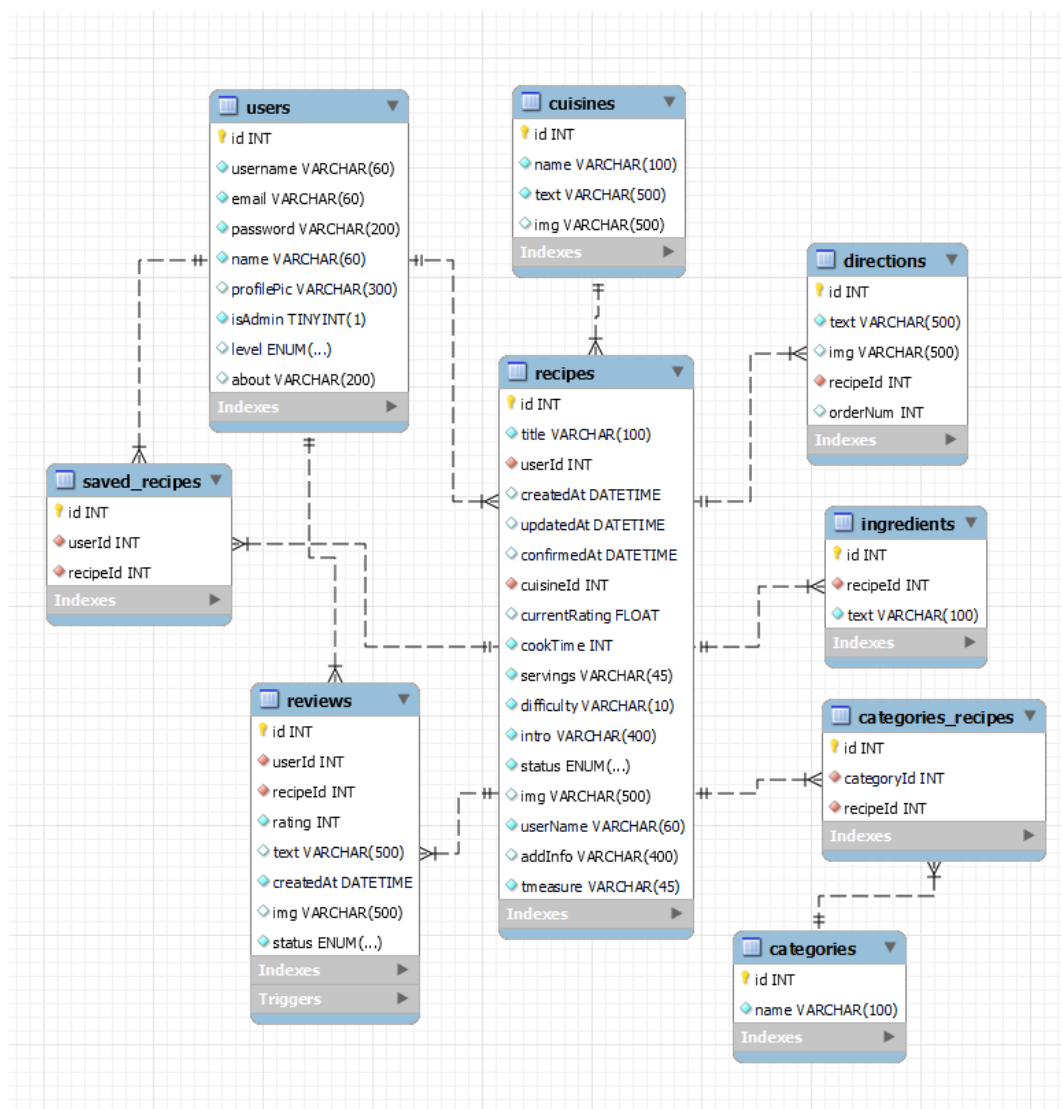


Рис. 2.3. Схема бази даних вебдодатку кулінарних рецептів

2.3 Проектування та реалізація алгоритмів роботи системи

Основним алгоритмом, який було обрано, став звичайний алгоритму односторінкового додатку (SPA) для клієнтської частини системи. Коли

користувач переходить на сторінку додатку або відкриває вебдодаток зі смартфона, сервер надсилає актуальну версію однієї сторінки (клієнтську частину). Далі, залежно від взаємодії користувача, клієнт надсилає запит на сервер. Після отримання відповіді від серверу, клієнт оновлює вигляд сторінки. Блок-схема загального алгоритму надана нижче (рис. 2.4) [17].

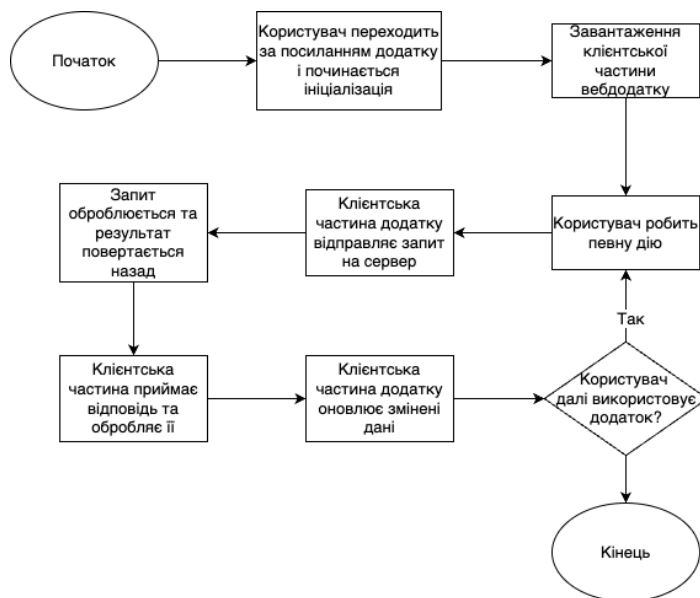


Рис. 2.4. Блок-схема алгоритму роботи клієнтської частини вебдодатку

Після визначення загального алгоритму, слід сформулювати алгоритм для окремих випадків. У даній ситуації зазвичай використовують діаграми активності.

Діаграма діяльності або активності є важливою поведінковою діаграмою в UML, та застосовується для опису динамічних аспектів системи. Вона, фактично, являє собою вдосконалену версію блок-схеми, яка моделює потік від однієї діяльності до іншої діяльності [18].

Почнемо з розгляду діаграми активності для авторизації та виконання дій користувача, що здійснив вхід до профілю.

Опис діаграми активності (Користувач):

1. Перегляд сторінки входу – користувач перейшов на сторінку входу до профілю.
2. Авторизація з логіном та паролем – користувач вводить власні дані на натискає на кнопку «Вхід».
3. Перевірка даних – система перевіряє чи існує користувач та чи співпадає пароль, якщо ні, то виводить повідомлення. Якщо так, відбувається вхід.

4. Вхід успішний – авторизація пройшла успішно, користувач тепер має доступ до додаткових функцій.
5. Перегляд рецептів – користувач переглядає рецепти та обирає необхідний.
6. Додавання в улюблені – одна з додаткових функцій, тож користувач може зберегти вподобані рецепти до улюблених.
7. Написання коментарів – тепер користувач може писати відгуки під рецептами.
8. Перегляд профілю – користувач має можливість переглянути власний профіль.
9. Додавання рецепту – користувач може додати власний рецепт, перейшовши зі свого профілю на «Додати рецепт» та виконавши подальші дії. Детальніше це буде описано в наступній діаграмі активності.
10. Пропонування рецепту – буде описано в наступній таблиці.
11. Редагування особистих даних – користувач може змінювати дані про себе та пароль.

Діаграма активності для Користувача представлена на рисунку 2.5.

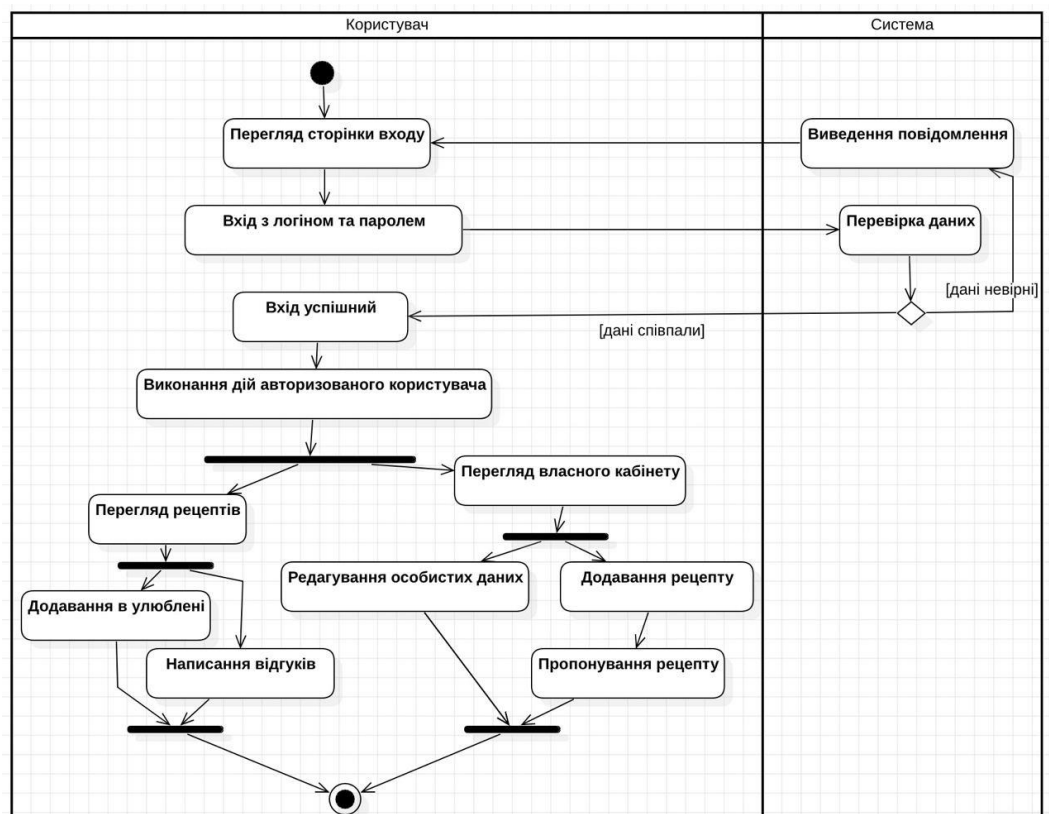


Рис. 2.5. Діаграма активності та відповідність станів та дій Користувача

Наступною стане діаграма активності для пропонування рецепту користувачем.

Опис діаграми активності (Авторизований користувач):

1. Перехід до власного профілю – користувач після авторизації переходить до власного профілю.
2. Додавання рецепту – користувач в профілі натискає «Додати рецепт» та перед ним з'являється форма.
3. Введення даних – користувач вводить дані про рецепт та надсилає їх.
4. Перевірка даних – система перевіряє дані і якщо якесь поле заповнено невірно, то виводить повідомлення. У протилежному випадку, дозволяє надсилання форми.
5. Виведення повідомлення – система показує повідомлення про те, що певні дані були введені невірно.
6. Пропонування рецепту – користувач обирає опцію пропонування рецепту.
7. Надсилання рецепту – відбувається відправлення рецепту адміністратору на перевірку змісту.
8. Перевірка рецепту – адміністратор перевіряє рецепт.
9. Чи відповідає рецепт усім критеріям? – адміністратор оцінює чи є цей рецепт правильним та визначає чи він є необхідним у веб-додатку.
10. Відмова у розміщенні – у випадку коли рецепт не має сенсу чи порушує правила, адміністратор може видалити його, але якщо рецепт потрібно змінити, то відхиляє його з можливістю повторної виправки.
11. Рецепт відображено у профілі та на сайті – якщо рецепт відповідає усім стандартам, то адміністратор додає його до наявних рецептів. Новостворений рецепт відображається в останніх доданих рецептах з авторством користувача, що його запропонував. Також схвалений рецепт відображається у профілі даного користувача.

Діаграма активності для Авторизованого користувача відображена на рисунку 2.6.

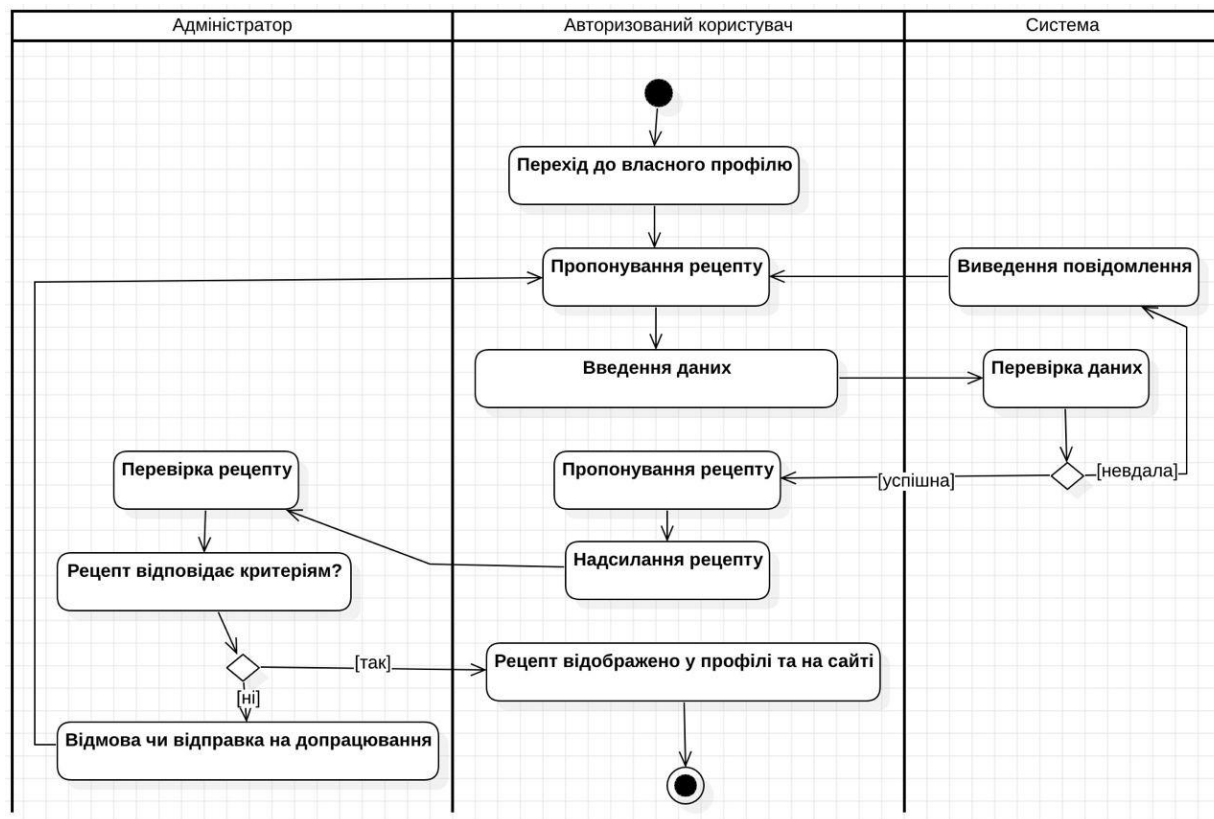


Рис. 2.6. Діаграма активності та відповідність станів та дій Авторизованого користувача та Адміністратора

1. Опис діаграми активності (Адміністратор):
2. Перегляд входу – адміністратор входить в додаток.
3. Авторизація з логіном та паролем – користувач вводить у форму свої дані.
4. Перевірка даних – система перевіряє чи такий користувач існує, якщо ні, то виводить повідомлення.
5. Вхід успішний – авторизація пройшла успішно.
6. Перехід по посиланню на панель адміністратора – користувач переходить за посиланням, де можливо переглянути панель адміністратора.
7. Перевірка рівня доступу – система перевіряє чи має існуючий користувач права адміністратора, якщо ні, виводить повідомлення, перенаправленням на сторінку помилки.
8. Авторизація успішна – користувач переходить до панелі адміністрування.
9. Доступ до дій панелі адміністратора – адміністратор може керувати користувачами, кухнями світу, рецептами та відгуками.

10. Керування користувачами – адміністратор має право змінювати дані про користувачів, надавати їм роль адміністратора, видаляти їх.
11. Керування кухнями світу – адміністратор має змогу додавати нові кухні світу, змінювати поточні та видаляти їх.
12. Керування рецептами – адміністратор має можливість затвердити запропоновані рецепти, змінити поточні, а також видаляти, якщо це необхідно.
13. Керування відгуками – у даному пункті меню адміністратор може затверджувати та видаляти відгуки.

Діаграма активності для актора Адміністратор зображена на рисунку 2.7.

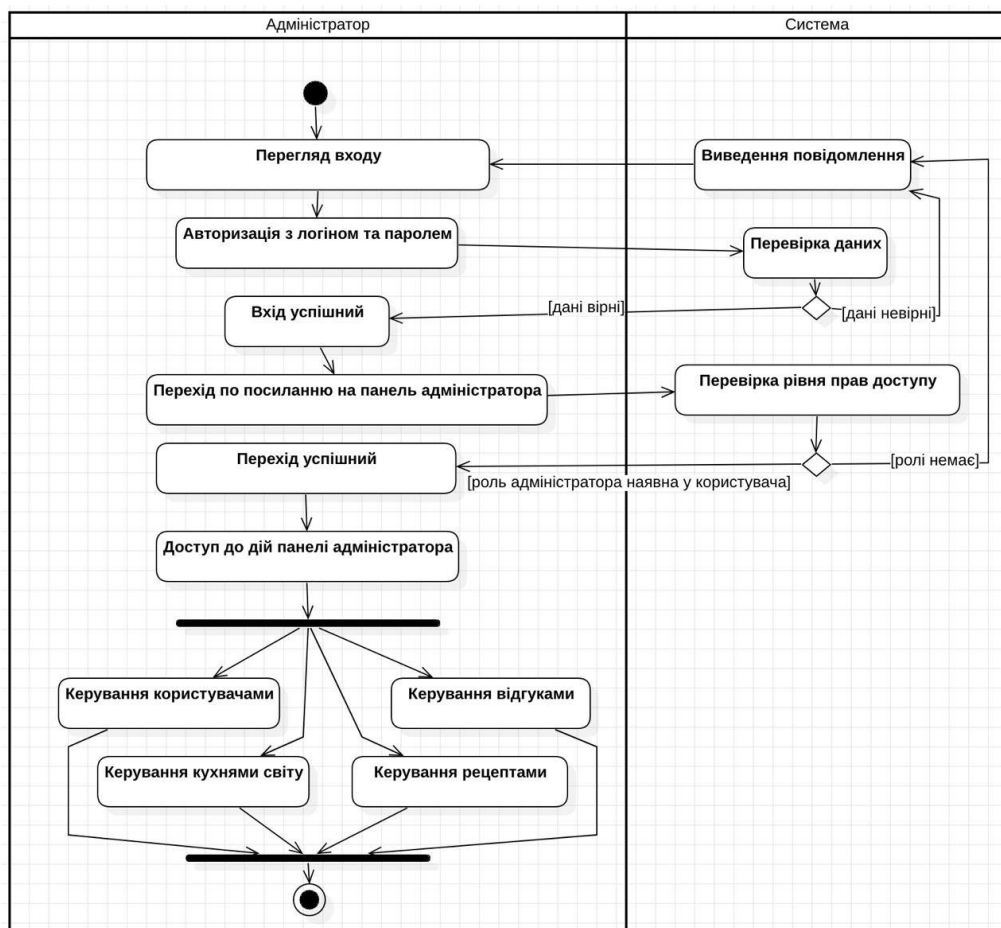


Рис. 2.7. Діаграма активності та відповідність станів та дій Адміністратора

2.4 Реалізація функціоналу вебдодатку кулінарних рецептів

Перейдемо до реалізації основи вебдодатку та функціоналу відгуків на рецепти. Врахуємо те, що базу даних вже було створено заздалегідь у MySQL Workbench. Для розробки було використане середовище Visual Studio Code [19].

2.4.1 Реалізація основної архітектури вебдодатку

Кулінарний додаток буде створений з клієнтської та серверної частин. Кожна з них буде самостійним додатком та розташується в каталозі «client» та «api» відповідно.

Серверна частина додатку (API) була розроблена з використанням платформи Node.js та фреймворку Express.js.

Спершу було ініціалізовано додаток Node.js, встановлено на нього Express.js та створено головний файл «index.js». Далі потрібно було імпортувати у файлі Express та написати:

```
const app = express();
app.listen(9900, () => {
  console.log("api work");
});
```

Перший рядок коду є початковою точкою входу для ініціалізації серверної частини на Express.js. Він ініціалізує новий екземпляр додатку Express, який використовується для налаштування маршрутів, обробки запитів та відправлення відповідей. Наступний рядок запускає сервер і налаштовує його на прослуховування вхідних HTTP-запитів на порт 9900 і після успішного запуску сервера виводить надпис у консоль.

Далі потрібно було встановити необхідні бібліотеки, що включають CORS.

CORS (Cross-Origin Resource Sharing) є пакетом для Node.js, що надає посередництво для Express, дозволяючи вмикати обмін ресурсами між різними джерелами [20].

Наданий нижче код дозволяє серверу: обробляти JSON об'єкти у тілі запиту, запити лише з http://localhost:5173 для безпеки, читати та обробляти cookie у запитах для збереження стану аутентифікації.

```
app.use(express.json());
app.use(
  cors({
    origin: "http://localhost:5173",
  })
);
app.use(cookieParser());
```

Після цього було створено необхідні папки та файли, утворено структуру API (рис. 2.8)

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						37

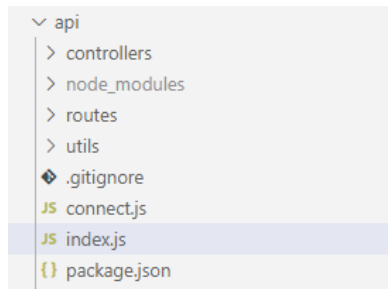


Рис. 2.8. Структура серверної частини веб-додатку

Дані папки містять маршрути (папка «routes»), контролери (папка «controllers») та додаткові функції в папці «utils». Крім того було створено файл «connect.js» для підключення до бази даних:

```
import mysql from "mysql2"

export const db = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: password,
  database: "yumbook"
})
```

Потім необхідно було створити контролер «auth.js», який буде забезпечувати функціонал реєстрації та автентифікації користувачів. Розглянемо метод реєстрації.

```
export const register = (req, res) => {
  //перевірка чи користувач з таким логіном існує
  const q = `SELECT * FROM users WHERE username = ?`;

  db.query(q, [req.body.username], (err, data) => {
    if (err) return res.status(500).json(err);
    if (data.length) return res.status(409).json("User already exists!");

    //створення нового користувача
    //хешування паролю
    const salt = bcrypt.genSaltSync(10);
    const hashedPassword = bcrypt.hashSync(req.body.password, salt);
    const q =
      "INSERT INTO users (username, email, password, name, level, profilePic)
      VALUE (?)";
    const img = req.body.img ? req.body.img : "NULL";
    const values = [
      req.body.username, req.body.email, hashedPassword, req.body.name,
      req.body.level, img];
    db.query(q, [values], (err, data) => {
      if (err) return res.status(500).json(err);
      return res.status(200).json("User successfully created");
    });
  });
};
```

При реєстрації спершу перевіряється чи не існує користувач з ідентичним логіном, якщо існує, то повертає помилку. У зворотньому випадку продовжує процес реєстрації, хешує отриманий пароль за допомогою бібліотеки Bcrypt та робить запит в базу даних на додавання даних про нового користувача. Якщо SQL-запит відпрацьовує без помилок, тоді у таблицю users додається користувач [21].

Далі створимо маршрут з аналогічною назвою до контролера та додамо код, який створює маршрут Express.js, а потім додає запит POST:

```
import express from "express";
import { register } from "../controllers/auth.js";

const router = express.Router();
router.post("/register", register);

export default router;
```

А у файлі «index.js» додамо раніше створений маршрут:

```
app.use("/api/auth", authRoutes);
```

І нарешті запустимо наш додаток (рис. 2.9):

```
○ yarn run v1.22.19
$ nodemon index.js
[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
api work
```

Рис. 2.9. Працюючий додаток Node.js

Postman – це інструмент для тестування API, що дозволяє створювати, відправляти та автоматизувати HTTP-запити, переглядати відповіді та документувати API [22].

Виконаємо запит до API за допомогою сайту Postman (рис. 2.10):

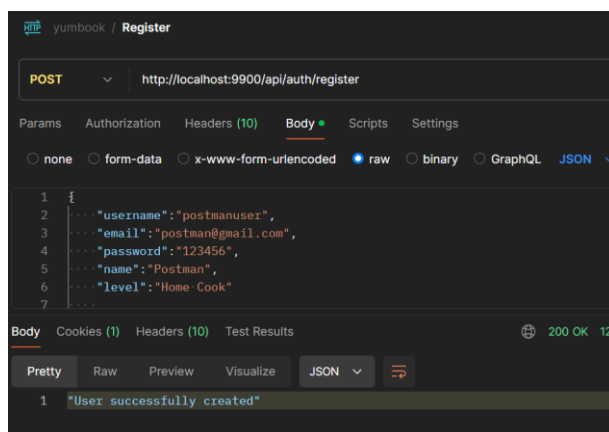


Рис. 2.10. Успішне створення користувача

Vite є сучасним інструментом розробки, який швидко набирає популярність завдяки швидкій компіляції коду та майже миттєвій гарячій заміні модулів. Він призначений для створення структур проєктів і їх групування [23].

Клієнтська частина вебдодатку, в свою чергу, була розроблена за допомогою фреймворку React. Після ініціалізації додатку React за допомогою Vite, з'являється основна структура додатку, а також файл «App.jsx». У React даний файл часто використовується як точка входу в додаток. Він містить основний компонент, що відображається на екрані при запуску. Це спрощує структуру проєкту та робить керування легше. Так як даний файл містить складну структуру, зазначено буде лише частину, повний лістинг наявний в Додатку А.

Наступний код повертає компонент App, що містить у собі маршрутизацію.

```
return (  
  <>  
    <div className="App">  
      <RouterProvider router={router} />  
    </div>  
  </>  
);
```

Було використано React Router для створення навігації. Лістинг роутера з маршрутами наведено нижче [24].

```
const router = createBrowserRouter([  
  {  
    path: "/",  
    element: <Layout />,  
    errorElement: <NotFoundPage />,  
    children: [  
      {  
        path: "/",  
        element: <Home />,  
      },  
    ],  
  },  
]);
```

У головному роутері знаходиться обгортка Layout, елемент що відображається при помилці та один підмаршрут – компонент Home, що відображає власний вміст на головній сторінці. Компонент Layout відповідає за загальну структуру сторінки та містить компоненти, що є стійкими під час використання додатку, наприклад: навігаційна панель, футер. Лістинг даного компоненту:

```
const Layout = () => {  
  return (  
    <QueryClientProvider client={queryClient}>
```

					ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
						40

```

        <div>
            <Navbar />
            <Outlet />
            <Footer />
        </div>
    </QueryClientProvider>
);
};

```

Отже, компонент Layout поєднує ці три елементи для створення єдиного інтерфейсу користувача для кожної сторінки веб-додатку. Це полегшує розробку, забезпечуючи однорідність і стандартизацію відображення сторінок.

Стилі CSS були реалізовані за допомогою препроцесора SCSS, який спрощує їх написання. Нижче наведено лістинг для стилю компоненту CuisineCard:

```

.cuisine_card {
    border: 2px solid #106B4A;
    border-radius: 10px;
    font-size: 18px;
    min-width: 250px;
    flex-grow: 2;
    flex: 1;
    @media (max-width:600px) {
        font-size: 16px;
        width: 100%;
    }
    &:hover {
        cursor: pointer;
        color: #0B7E55;
        border-color: black;
    }
    .inside{
        width: 100%;
        padding: 1rem;
    }
}

```

Для зручної авторизації було використано бібліотеку Zustand. Завдяки їй було створено глобальний стан useAuthStore, який містить поточного користувача та роль користувача, а також методи для взаємодії зі станом при вході та виході з облікового запису. Лістинг даного стану знаходиться в Додатку А.

Zustand - це невелике, швидке і масштабоване рішення для управління станом у React, яке не навантажує непотрібним кодом. У нього є зручний API на основі хуків [25].

2.4.2 Реалізація функціоналу додавання відгуків до рецепту

Створення функціоналу зручніше починати з API. Тому, давайте розглянемо функцію в контролері «review.js» для додавання відгуку у БД. Вона додає в базу даних відгук за масивом даних, що прийшли за запитом. А також встановлює відгуку статус «запропонований», час та дату додавання:

```
export const addReview = (req, res) => {
  const q =
    "INSERT INTO reviews (userId, recipeId, rating, text, createdAt, status,
img) VALUE (?)"
  const values = [
    req.user.id,
    req.body.recipeId,
    req.body.rating,
    req.body.text,
    moment(Date.now()).format("YYYY-MM-DD HH:mm:ss"),
    "Submitted",
    req.body.img,
  ]

  db.query(q, [values], (err, data) => {
    if (err) return res.status(500).json(err);
    return res.status(200).json("Ingredient has been created");
  });
};
```

Далі розглянемо додавання маршруту в «reviews.js». Новостворений маршрут використовує функцію-посередника verifyUser для перевірки чи користувач авторизований:

```
router.post("/add", verifyUser, addReview);
```

І також в «index.js» додамо маршрут з файлу «reviews.js»:

```
app.use("/api/reviews", reviewsRoutes);
```

Варто також переглянути код, що використовується для завантаження файлів на сервер у вебдодатку:

```
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, "../client/public/upload");
  },

  filename: function (req, file, cb) {
    cb(null, Date.now() + file.originalname);
  },
});

const upload = multer({ storage: storage });

app.post("/api/upload", upload.single("file"), (req, res) => {
```

```
const file = req.file;
res.status(200).json(file.filename);
});
```

Цей код використовує бібліотеку Multer для створення сховища файлів, яке визначає місце збереження файлів та генерує нові імена для завантажених файлів, щоб уникнути їх дублювання. Також створюється маршрут для обробки запиту на завантаження файлу [26].

У клієнтській частині розглянемо саме функції для додавання відгуку у компоненті AddReview.jsx. Спершу переглянемо функцію для виклику по натисканню кнопки, додавання рецепту на сервер:

```
const handleClick = async (e) => {
  e.preventDefault();
  if (!currentUser) return navigate("/signin");
  if (!text.trim()) return setErr("Будь ласка, введіть текст відгуку");
  if (rate === 0) return setErr("Будь ласка, вкажіть рейтинг");

  let imgUrl = "";
  if (file) imgUrl = await upload();

  try {
    await addReview(params.recipeId, rate, text, imgUrl || null);
    setText("");
    setFile(null);
    setRateWidth("0%");
    setRefetchR((prevState) => !prevState);
  } catch (error) {
    console.error("Failed to add review:", error);
  }
};
```

Функція handleClick викликається при натисканні користувачем кнопки "Надіслати" у формі відгуку. Вона перевіряє, чи користувач автентифікований, чи введений текст відгуку, чи встановлений рейтинг. Якщо всі перевірки успішні, вона виконує завантаження зображення, якщо воно прикріплене, та додає новий відгук на сервері за допомогою функції addReview. У разі успіху, поля форми очищаються, а також відбувається оновлення даних рецепту. У випадку помилки, вона відображається користувачу. Далі наведена функція addReview:

```
const addReview = async (recipeId, rate, text, imgUrl) => {
  try {
    const response = await makeRequest.post(
      `/reviews/add?userId=${currentUser.id}`,
      {
        recipeId: recipeId,
        rating: rate,
        text: text,
        img: imgUrl ,
      }
    );
  }
};
```

```

    }
  );
  return response.data;
} catch (error) {
  console.error("Failed to add review:", error);
  throw error;
}
};

```

Вона відправляє запит до API на додавання нового відгуку до бази даних. Вікликається POST запит на сервер з даними відгуку, такими як ідентифікатор рецепту (recipeId), оцінка (rate), текстовий відгук (text) та посилання на зображення (imageUrl). Якщо запит успішний, функція повертає дані відповіді, інакше виводить помилку у консоль та повертає її, щоб її можна було обробити вище по стеку викликів. Лістинги усіх згаданих файлів знаходяться у Додатку А.

Залишилась одна функція. Вона викликає запит до API на завантаження файлу, якщо її викликають у функції handleClick:

```

const upload = async () => {
  try {
    const formData = new FormData();
    formData.append("file", file);
    const res = await makeRequest.post("/upload", formData);
    return res.data;
  } catch (err) {
    console.log(err);
  }
};

```

Висновки до другого розділу

У другому розділі були визначені варіанти використання, описані їх прецеденти. Також були розглянуті вимоги користувачів, функціональні, нефункціональні та системні вимоги до вебдодатку. Було створено класову діаграму, що відображає загальну логіку роботи програмного продукту. На основі діаграми сутностей та загальних вимог, було спроектовано дев'ять таблиць для бази даних вебдодатку кулінарних рецептів, а також створено схему бази даних.

Крім того, було здійснено реалізацію ключової архітектури веб-додатку, з лістингами та описом, а також реалізацією функціоналу додавання відгуків.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБДОДАТКОМ КУЛІНАРНИХ РЕЦЕПТІВ

3.1 Порядок встановлення та налаштування параметрів додатку

Веб-додаток для кулінарних рецептів складається з ряду компонентів, які потребують окремого розгортання та налаштування. Щоб показати компоненти системи та процес розгортання, була створена діаграма розгортання (рис. 3.1) [27].

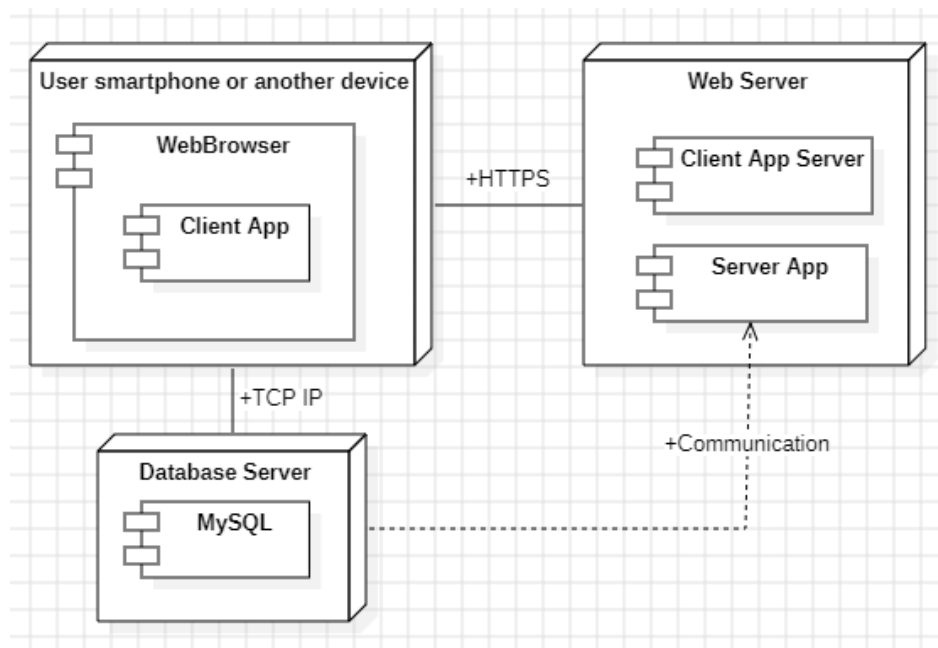


Рис. 3.1. Діаграма розгортання

Діаграма розгортання ілюструє систему, що включає три основні вузли: сервер, сервер бази даних та пристрій користувача (комп'ютер або інший пристрій). З'єднання між пристроєм користувача та веб-сервером здійснюється через протокол HTTPS. Протокол HTTP використовується для взаємодії між клієнтським та серверним додатками на сервері.

Протоколи HTTP/HTTPS функціонують на основі системи запитів/відповідей і використовують стандарт TCP для забезпечення надійної передачі даних. Взаємодія між сервером бази даних та веб-сервером здійснюється через протокол TCP/IP.

Для встановлення веб-додатка для кулінарних рецептів необхідно розгорнути та налаштувати всі компоненти системи. На сервері бази даних має бути встановлений сервер MySQL, створена база даних та налаштовані параметри

підключення. Клієнтському додатку потрібно лише наявність веб-браузера для кожного користувача. Для правильної роботи веб-сервера потрібно використовувати Node.js версії 20 або вище.

3.2 Структура інтерфейсу вебдодатку та порядок роботи

Користувачі можуть перейти у вебдодаток як зі смартфона, так і з планшета та комп'ютеру, бо дизайн було адаптовано під різні розміри екранів. Перейдемо до огляду додатку.

В першу чергу, користувачі побачать головну сторінку вебсайту (рис. 3.2). На ній вони можуть побачити чотири прийоми їжі, за якими можливо переглянути рецепти.

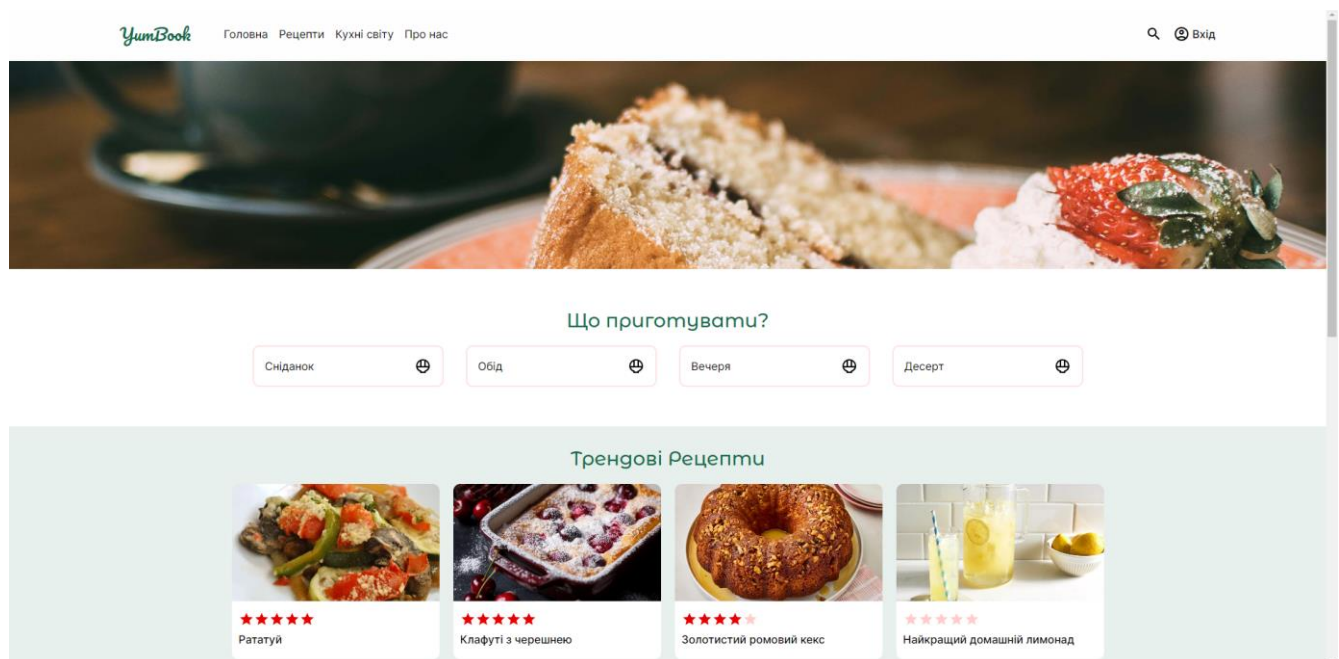


Рис. 3.2. Зовнішній вигляд головної сторінки при відкритті веб-додатку

У головної сторінки також є продовження, де знаходяться не тільки «Трендові Рецепти», а також «Останні Додані». Тому, користувачі зможуть знайти рецепт на будь-який смак та настрій (рис. 3.3) .

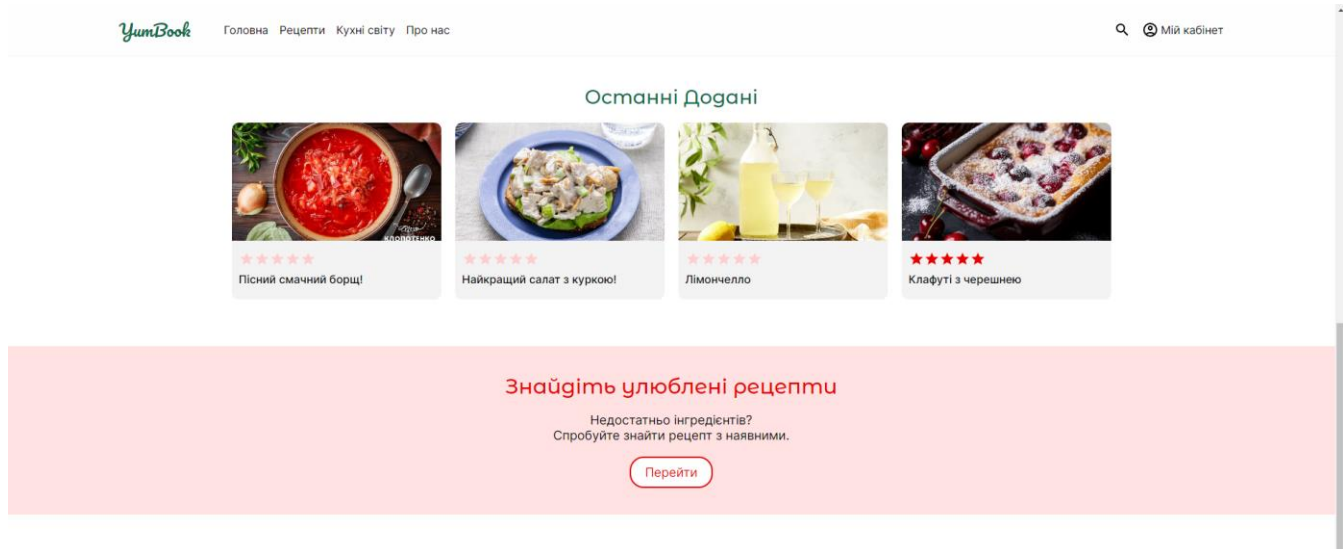


Рис. 3.3. Зовнішній вигляд головної сторінки після прокручування вниз

На рис. 3.3. також видно банер, при натисканні на кнопку «Перейти», відбудеться переміщення на сторінку пошуку рецептів. На дану сторінку можна також потрапити, натиснувши на навігаційній панелі на значок пошуку. Ось як виглядає пошук рецепту за назвою (рис. 3.4).

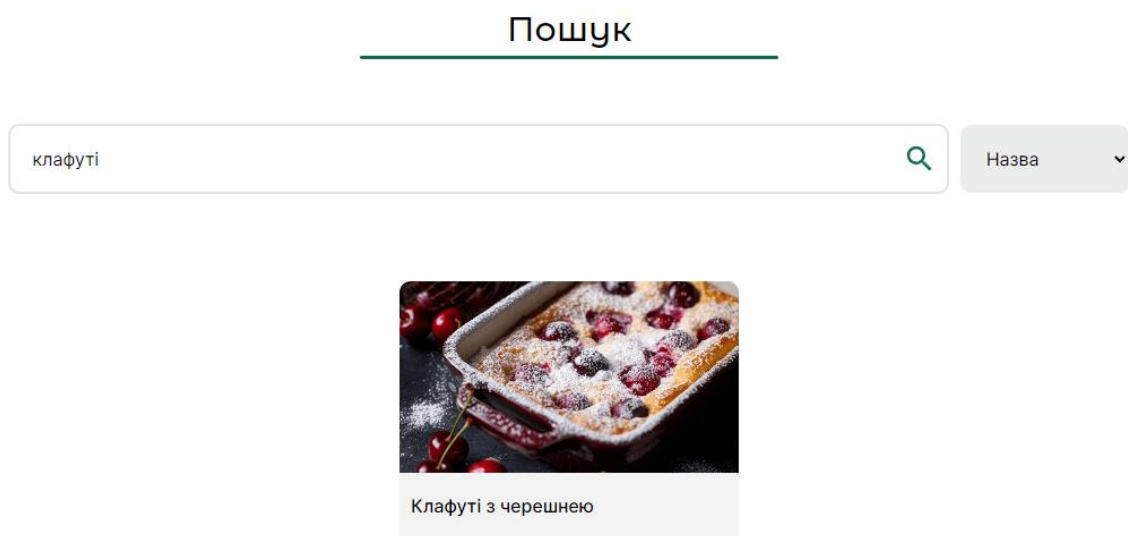


Рис. 3.4. Пошук рецептів за назвою

Можливо також змінити тип пошуку на інгредієнти і знайти рецепти за одним або декількома інгредієнтами. Це дозволить знайти рецепт, що відповідає конкретному випадку. Для пошуку за більше ніж одним інгредієнтом обов'язково потрібно вводити їх через кому. На рисунку 3.5 показано приклад пошуку за одним інгредієнтом.

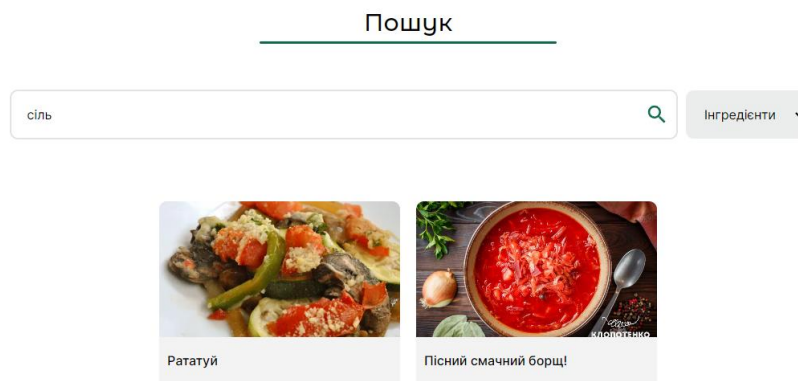


Рис. 3.5. Пошук рецептів за інгредієнтом

Пошук за кількома інгредієнтами розширює можливості пошуку, дозволяючи шукати рецепти на основі того, що наявне у користувача. Це значно підвищує зручність, особливо при наявності великої кількості рецептів. Даний вид пошуку зображений на рисунку 3.6.

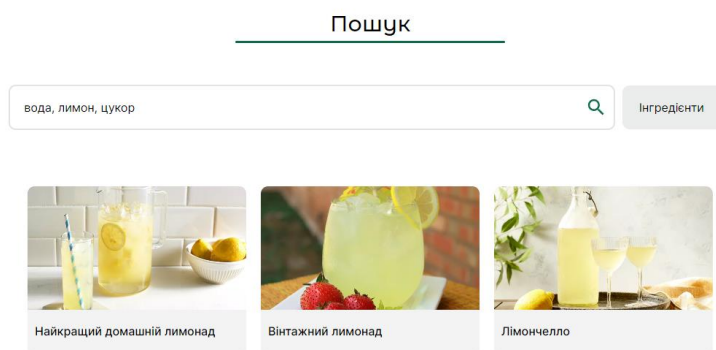


Рис. 3.6. Пошук рецептів за інгредієнтами

Потім переходимо на сторінку рецептів, натиснувши на пункт меню "Рецепти" у навігаційній панелі (рис. 3.7).

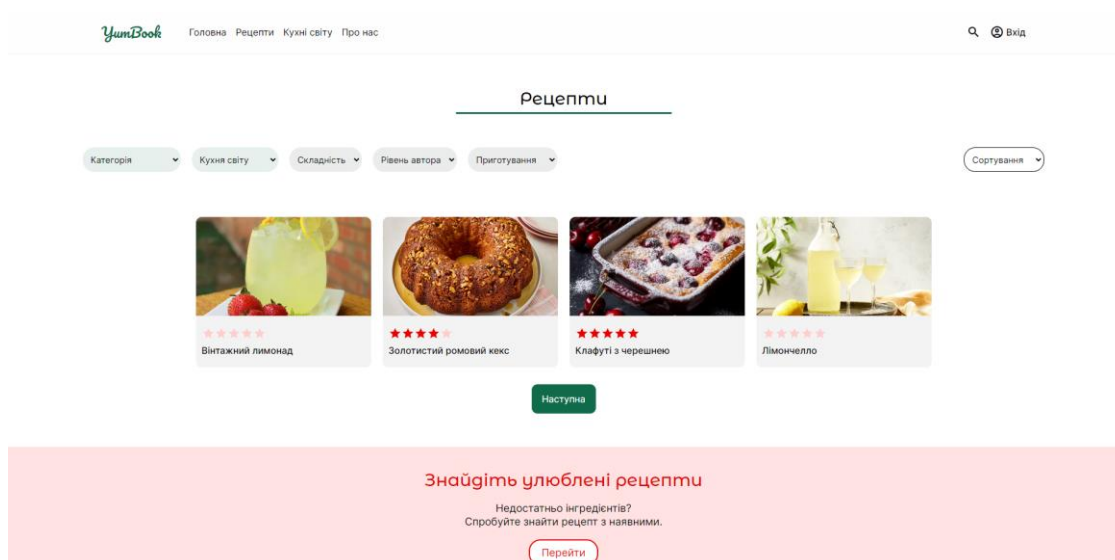


Рис. 3.7. Сторінка рецептів

На сторінці рецептів розміщені інструменти для фільтрації та сортування рецептів, які можна використовувати як окремо, так і у поєднанні. Фільтрацію рецептів показано на рисунку 3.8.

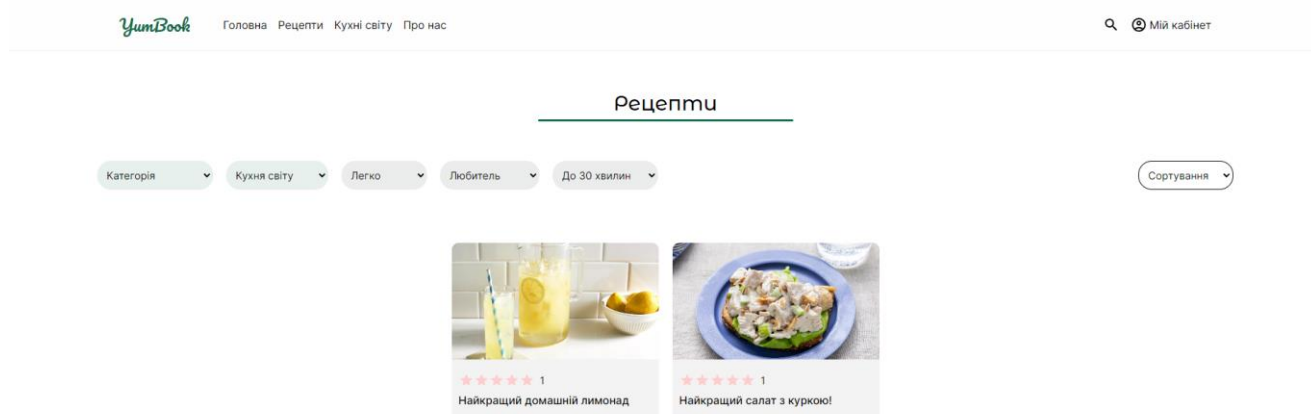


Рис. 3.8. Результат фільтрації за трьома критеріями

Відсортувати рецепти можливо за такими критеріями: за найкращим рейтингом, останніми, давно оновленими. Сортування за останніми оновленими зображено на рис. 3.9.

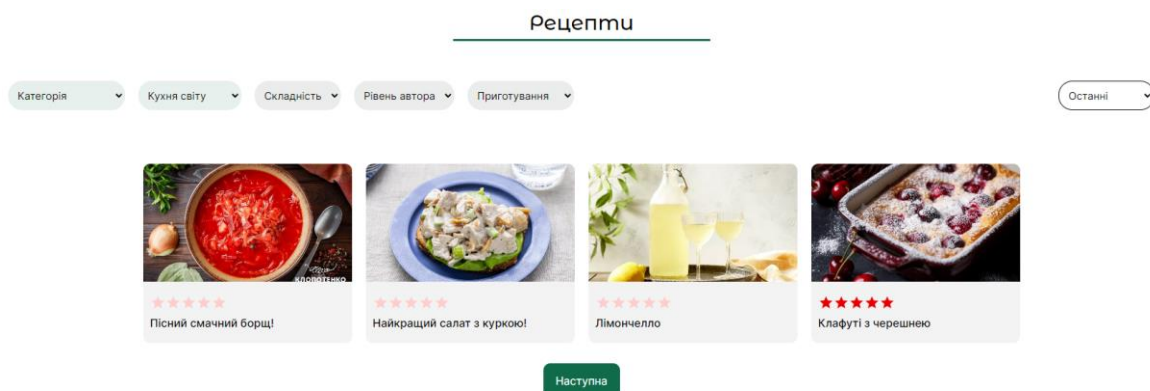


Рис. 3.9. Сортування за останніми

Тепер розглянемо одночасне фільтрування та сортування рецептів. Всі критерії застосовуються разом. Результат показано на рис. 3.10.

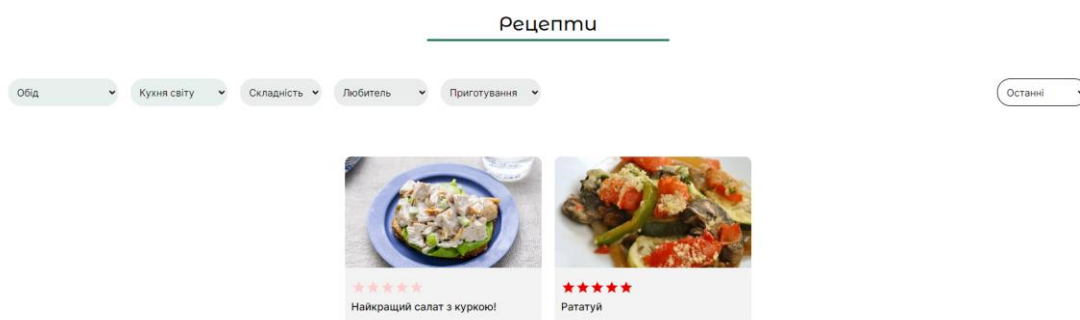


Рис. 3.10. Результат фільтрації та сортування

Натисканням на пункт меню "Рецепти" у навігаційній панелі, здійснюється перехід на сторінку кухонь світу (рис. 3.11).

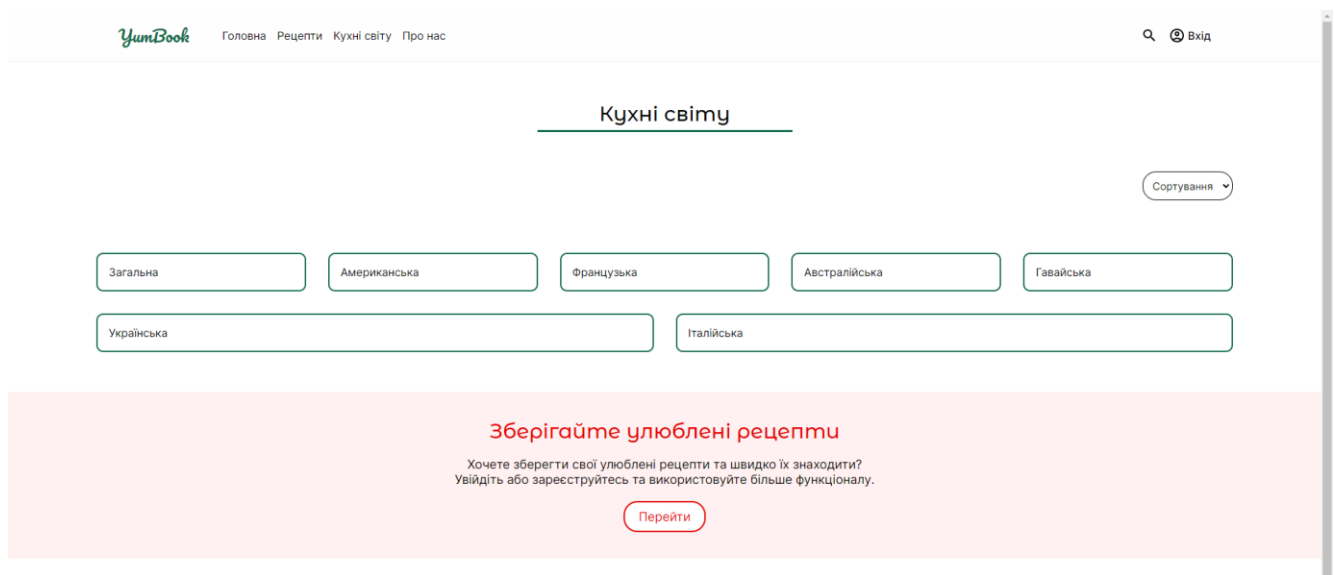


Рис. 3.11. Сторінка кухонь світу

На даній сторінці відображені всі кухні світу що у вебдодатку. Їх також можливо сортувати за назвою: від а до я, від я до а. Це дозволить швидше знайти необхідну кухню.

Обираємо загальну кухню та відбувається перенаправлення на сторінку відповідної кухні світу (рис. 3.12).

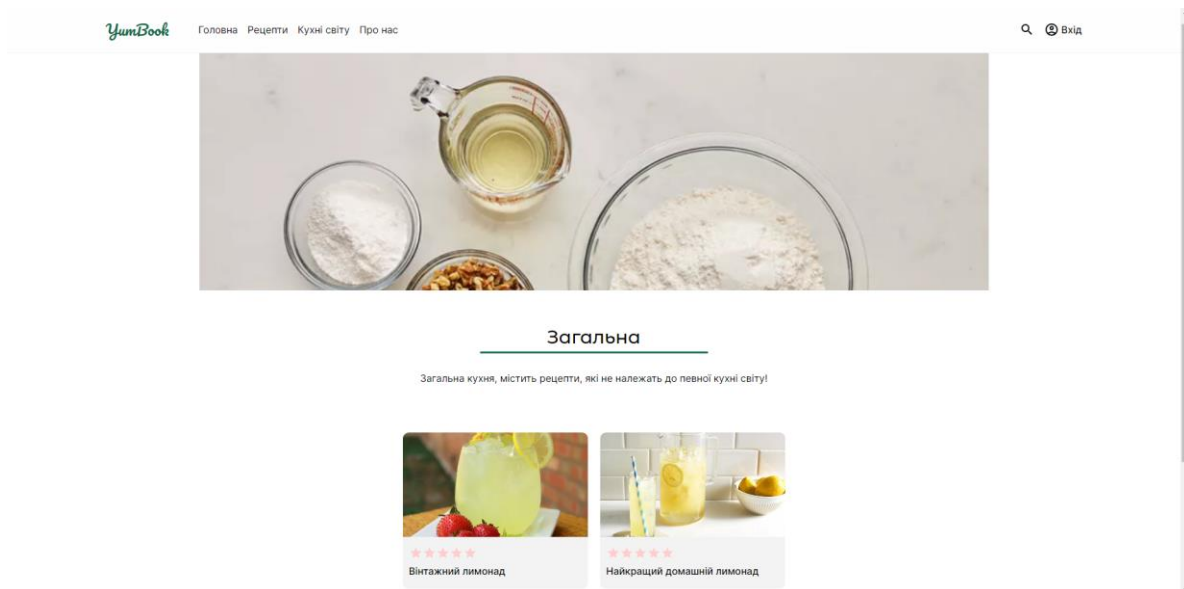


Рис. 3.12. Сторінка кухні світу

Повертаємось на сторінку рецептів та обираємо необхідний рецепт, відбувається перехід на сторінку окремого рецепту (рис. 3.13).

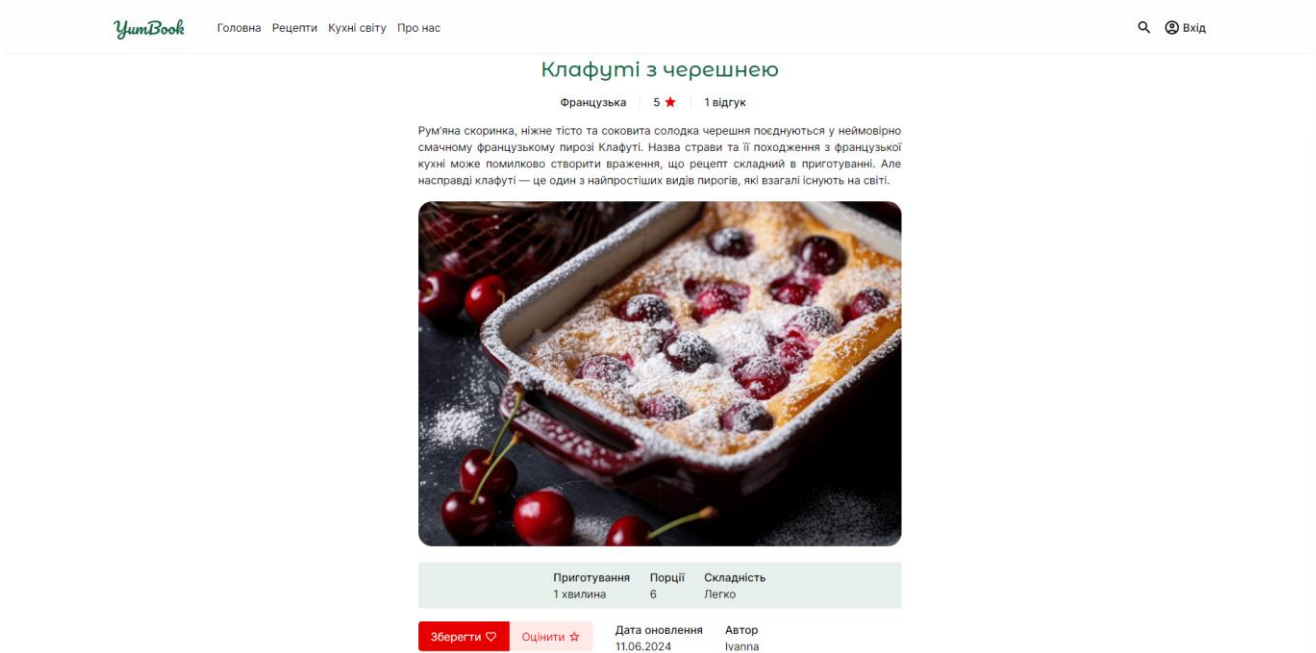


Рис. 3.13. Сторінка рецепту

Сторінка рецепту містить також список інгредієнтів та кроки приготування. Друге забезпечує розуміння користувачем послідовності приготування певного рецепту. Інтерфейс наявний на рисунку 3.14.

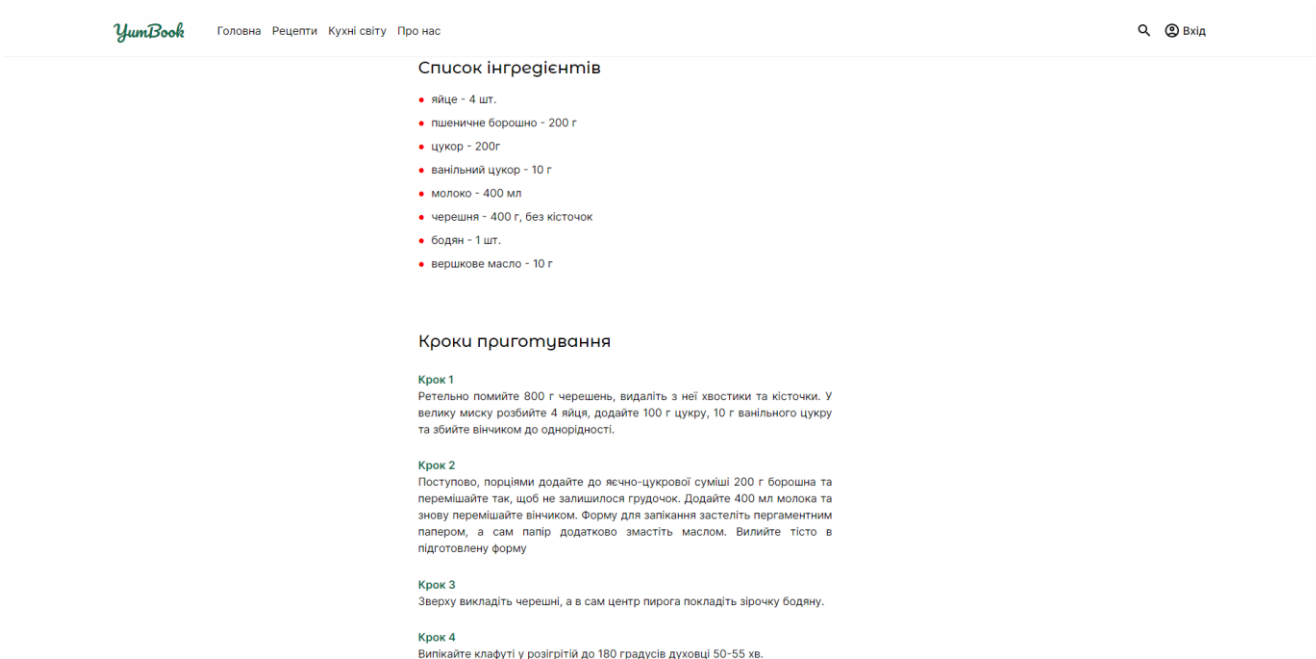


Рис. 3.14. Зовнішній вигляд списку інгредієнтів та кроків приготування

Після кроків знаходиться секція відгуків на рецепт. Тут можна переглянути відгуки від інших користувачів або спробувати залишити власний (рис. 3.15).

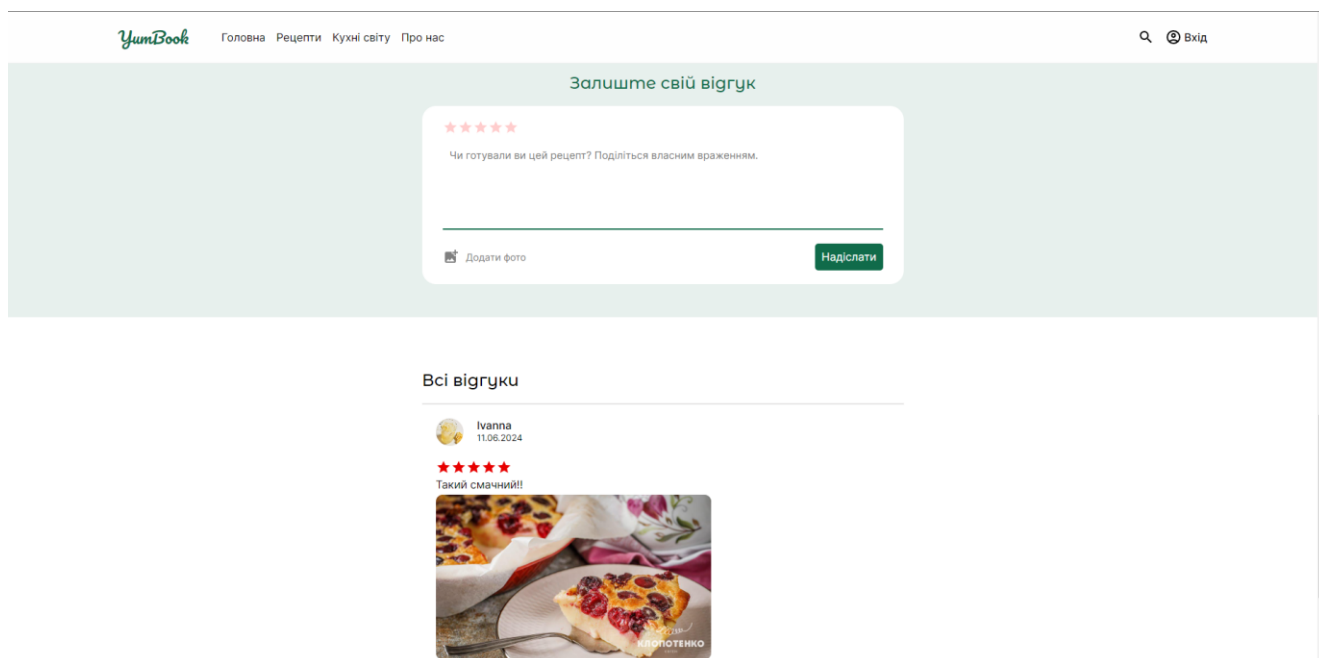


Рис. 3.15. Відгуки на рецепт

При спробі залишити відгук на рецепт, відбувається перенаправлення на сторінку входу у обліковий запис. Тобто, відгук може залишити тільки автентифікований користувач. Сторінка входу зображена на рисунку 3.16.

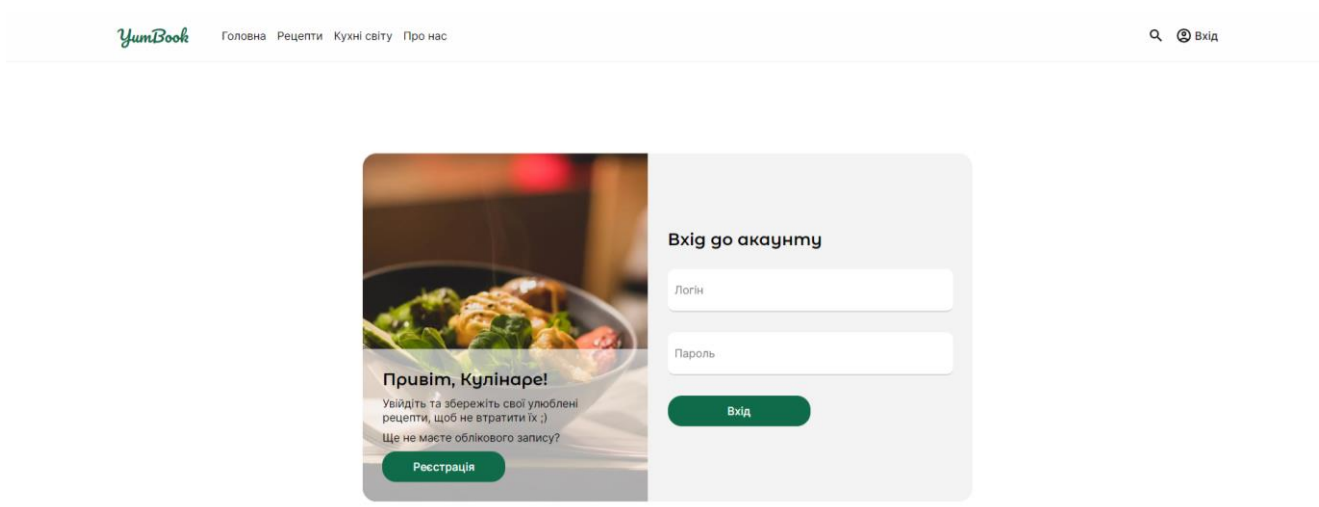
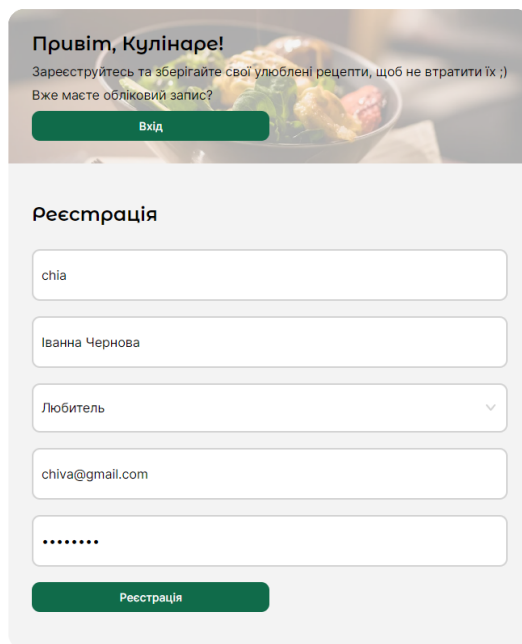


Рис. 3.16. Вигляд сторінки входу

Проте, якщо немає облікового запису, натискаємо на кнопку «Реєстрація». Відбувається перехід на сторінку реєстрації. Тоді, заповнимо форму необхідною інформацією про себе: логін, ім'я, рівень кулінарії, електронна пошта та пароль (рис. 3.17).



Привіт, Кулінаре!
Зареєструйтесь та зберігайте свої улюблені рецепти, щоб не втратити їх :)
Вже маєте обліковий запис?

Вхід

Реєстрація

chia

Іванна Чернова

Любитель

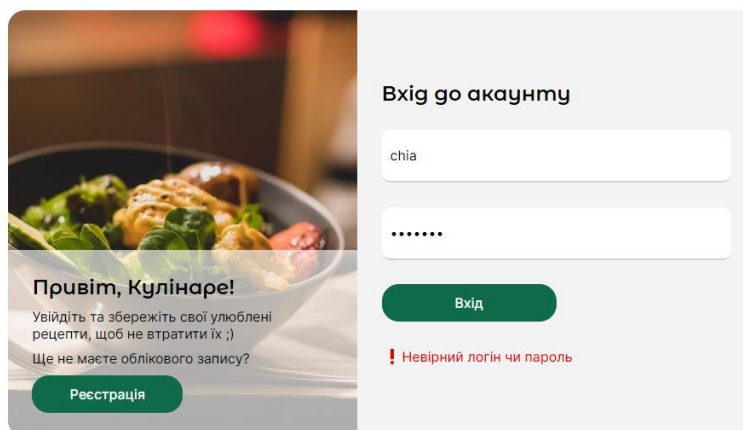
chiva@gmail.com

.....

Реєстрація

Рис. 3.17. Сторінка реєстрації з введеними даними

Після натискання на кнопку «Реєстрація», відбувається успішне створення облікового запису і ми зможемо здійснити вхід. Для цього потрібно вказати логін та пароль, що ми вказували при реєстрації. Якщо буде невірно введено одне зі значень, при натисканні на кнопку «Вхід», відобразиться помилка (рис. 3.18).



Вхід до акаунту

chia

.....

Вхід

Реєстрація

Привіт, Кулінаре!
Увійдіть та збережіть свої улюблені рецепти, щоб не втратити їх :)
Ще не маєте облікового запису?

! Невірний логін чи пароль

Рис. 3.18. Помилка при спробі входу з невірними даними

Коли буде введено вірні дані та натиснуто на кнопку, відбудеться перехід на головну сторінку, а в навігаційній панелі з'явиться пункт «Мій кабінет», при наведенні на який відображається меню (рис. 3.19).

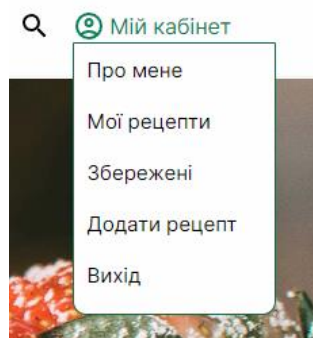


Рис. 3.19. Меню при наведенні

При натисканні «Про мене», відбувається перехід до кабінету та інформації про користувача, який можна побачити на рисунку 3.20.

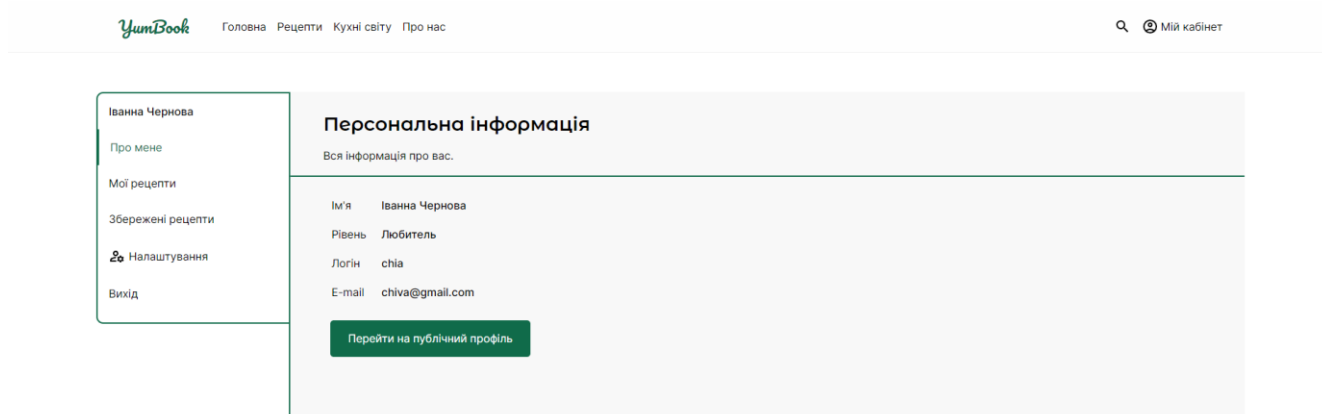


Рис. 3.20. Персональна інформація про користувача

Обираємо «Мої рецепти» і відображається відповідна сторінка, де знаходяться рецепти, що додав користувач. Сторінка з рецептами користувача наявна на рисунку 3.21.

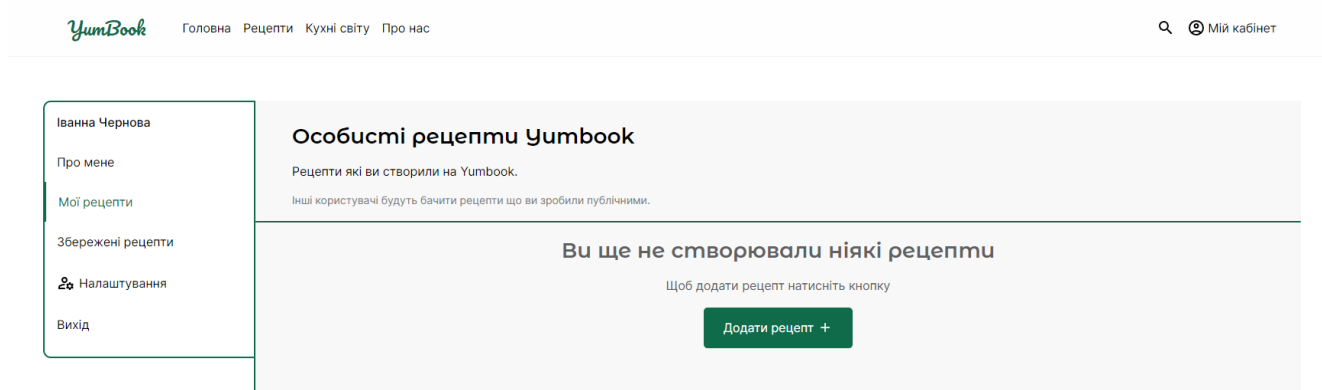


Рис. 3.21. Особисті рецепти користувача

Наразі на даній сторінці немає рецептів, тому вони і не відображаються. Користувач натискає на кнопку «Додати рецепт». Відображається сторінка для додавання рецепту. Вводяться основні дані про рецепт (рис. 3.22).

+ Додати рецепт
Завантажити власні рецепти - легко! Додавайте їх до обраного та діліться з друзями, родиною, чи навіть спільнотою Yumbook.

Назва
Домашній курячий суп

Вступ
Цей рецепт домашнього курячого супу вартий того, щоб його приготувати — він корисний для тіла та душі. Як виходить, що звичайна курка та овочі, притовані разом, можуть бути такими ситними?

Додайте фото рецепту
Додати фото

Оберіть кухню світу
Загальна

Оберіть категорії для рецепту
Сніданок x Обід x Вечеря x Супи x

Рис. 3.22. Форма додавання рецепту з введеними даними

Продовжується додавання даних. Слід зауважити, що якщо користувач хоче запропонувати для публікації рецепт, він повинен чітко дотримуватись всіх вказівок. Наприклад для додавання інгредієнтів існує певний шаблон. Потрібно писати інгредієнт в однині, а через тире – його кількість (рис. 3.23).

Сніданок x Обід x Вечеря x Супи x

Список інгредієнтів
Введіть один інгредієнт на рядок. Вкажіть назву інгредієнту в однині та його кількість (у шт. ст.л. ч.л. склянках), через тире.

цїла курка - 1 шт.

морква - 4 шт.

велика цибуля - 1 шт.

вода - щоб покрити

сіль та перець - за смаком

Додати інгредієнт

Кроки приготування
Поясніть, як приготувати ваш рецепт, включаючи температуру духовки, час приготування або випікання, розміри форм для випічки тощо.

Зберіть всі інгредієнти, поріжте навіпіл моркву, селеру та цибулю.

Рис. 3.23. Форма додавання рецепту з введеним списком інгредієнтів

Для кроків також потрібно притримуватись певних критеріїв, зазначених у формі. Крім того, до кожного кроку можливо додавати зображення, яке допоможе іншим користувачам краще зрозуміти яким чином його виконувати (рис. 3.24).

Рис. 3.24. Форма додавання рецепту з введеними кроками

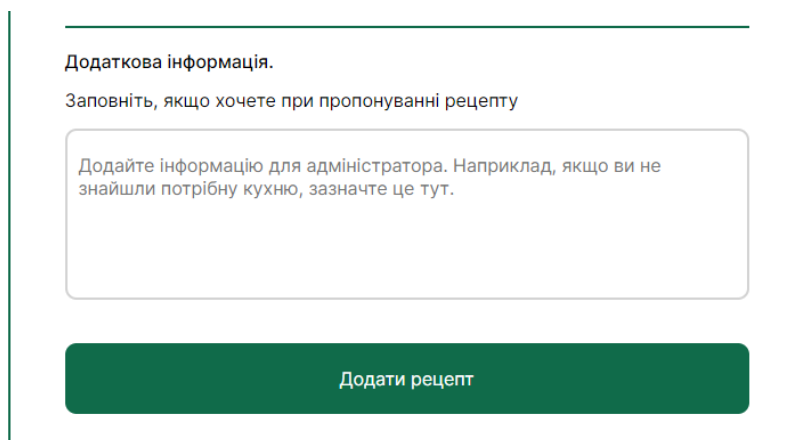
Слід зазначити, що на формі присутня перевірка обов'язкових полів за визначеними для кожного параметрами. Якщо прибрати з передостаннього кроку текст то отримаємо такий результат (рис. 3.25)

Рис. 3.25. Відображення помилки

Повернемо крок та введемо трохи додаткової інформації, такої як порції, час приготування, складність та статус для рецепту. Існує три статуси: видимий тільки користувачу, видимий у профілі, запропонований на публікацію (рис. 3.26).

Рис. 3.26. Форма додавання рецепту з введеними даними

Також наявне поле для додаткової інформації, в якому користувач може зазначити інформацію, якщо наприклад не знайшов необхідну кухню світу чи категорію (рис. 3.27).



Додаткова інформація.

Заповніть, якщо хочете при пропонуванні рецепту

Додайте інформацію для адміністратора. Наприклад, якщо ви не знайшли потрібну кухню, зазначте це тут.

Додати рецепт

Рис. 3.27. Форма додавання рецепту

Щоб додати рецепт, натискаємо на відповідну кнопку. Відбувається перехід на сторінку «Мої рецепти», де буде видно щойно доданий рецепт (рис. 3.28). Для того, рецепт опублікували, потрібно почекати поки адміністратор його перевірить та затвердить. Слід зазначити, що користувач може редагувати власні рецепти, доки вони не мають статус «Розміщений», форма виглядатиме аналогічно до додавання рецептів.

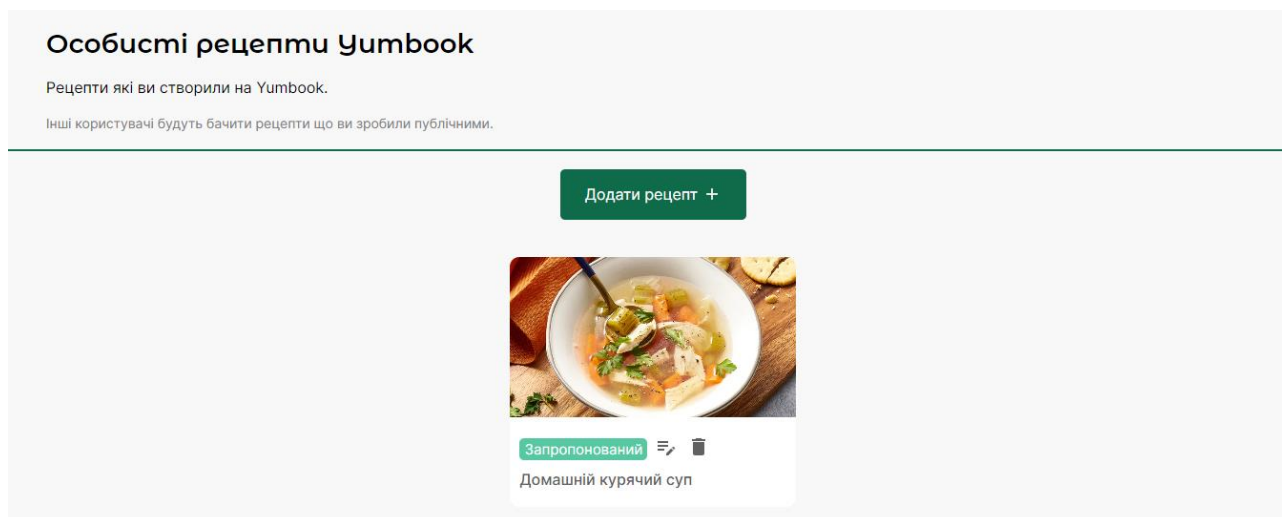


Рис. 3.28. Відображення щойно доданого рецепту

Далі переходимо в налаштування, де можливо змінити особисту інформацію, а також додати опис про себе та фото для публічного профілю. Інтерфейс даної сторінки зображений на рисунку 3.29.

Рис. 3.29. Сторінка налаштування

Переходимо на сторінку «Про мене» та натискаємо кнопку «Перейти на публічний профіль». Відображається профіль поточного користувача. Так як фото профілю відсутнє, відображається його стандартна заміна (рис. 3.30).

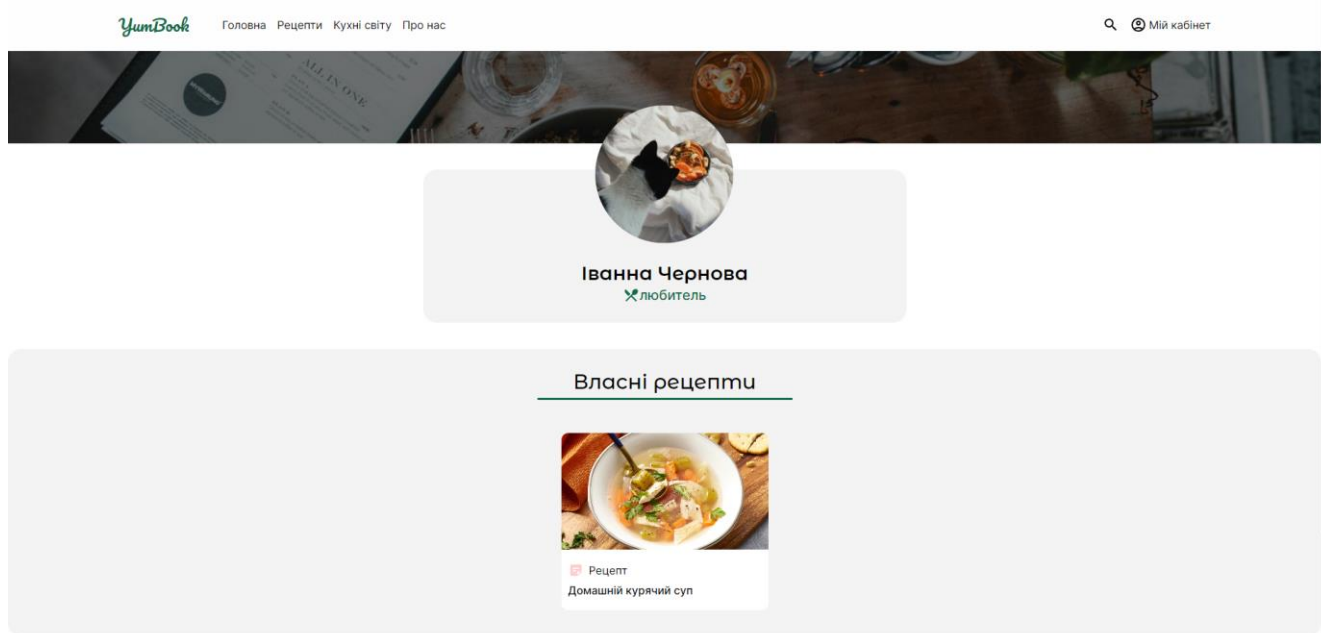


Рис. 3.30. Публічний профіль користувача

Після цього переходимо на сторінку рецептів та обираємо вподобаний з назвою «Клафуті з черешнею». Відображається сторінка рецепту, раніше зображена на рисунку 3.13. Натискаємо на кнопку «Зберегти» (рис. 3.31) та вона

змінює свій вигляд (рис. 3.32), що свідчить про те, що рецепт було додано до збережених.



Рис. 3.31. Кнопки до збереження рецепту

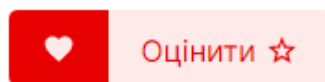


Рис. 3.32. Кнопки після збереження рецепту

Щоб пересвідчитись, перейдемо до збережених рецептів. Вона зображена на рис. 3.33.

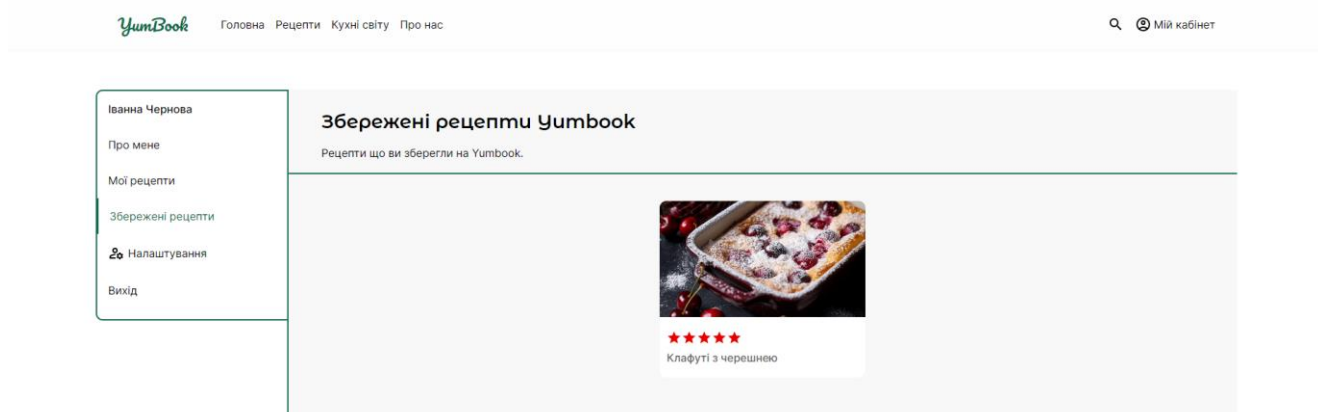


Рис. 3.33. Збережені рецепти користувача

Повертаємось на сторінку того ж рецепту та залишаємо відгук. Для цього заповнюємо текст та натискаємо кнопку, але з'являється помилка (рис. 3.34).

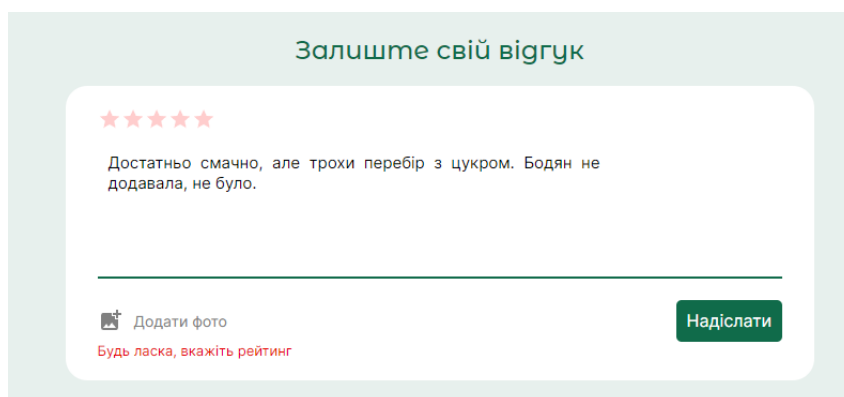


Рис. 3.34. Помилка при додаванні відгуку

Тоді зазначаємо рейтинг та натискаємо повторно на кнопку. Відгук відображається у відповідному місці (рис. 3.35), але має текст «Запропоновано вами», що означає те, що відгук повинен затвердити адміністратор.

Всі відгуки

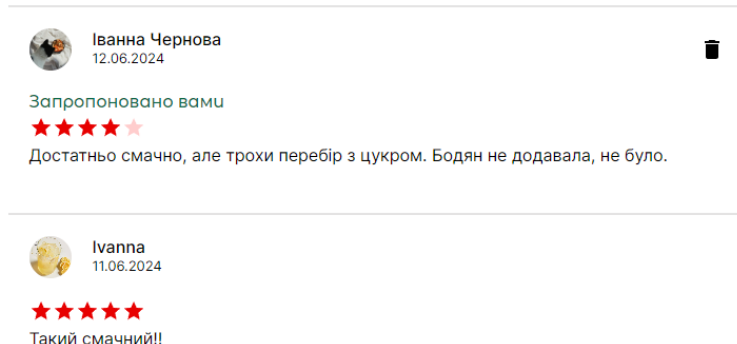


Рис. 3.35. Відображений щойно доданий відгук

Також ми можемо перейти на профілі інших, натиснувши на ім'я у коментарі. Після чого відобразиться профіль відповідного користувача (рис. 3.36).

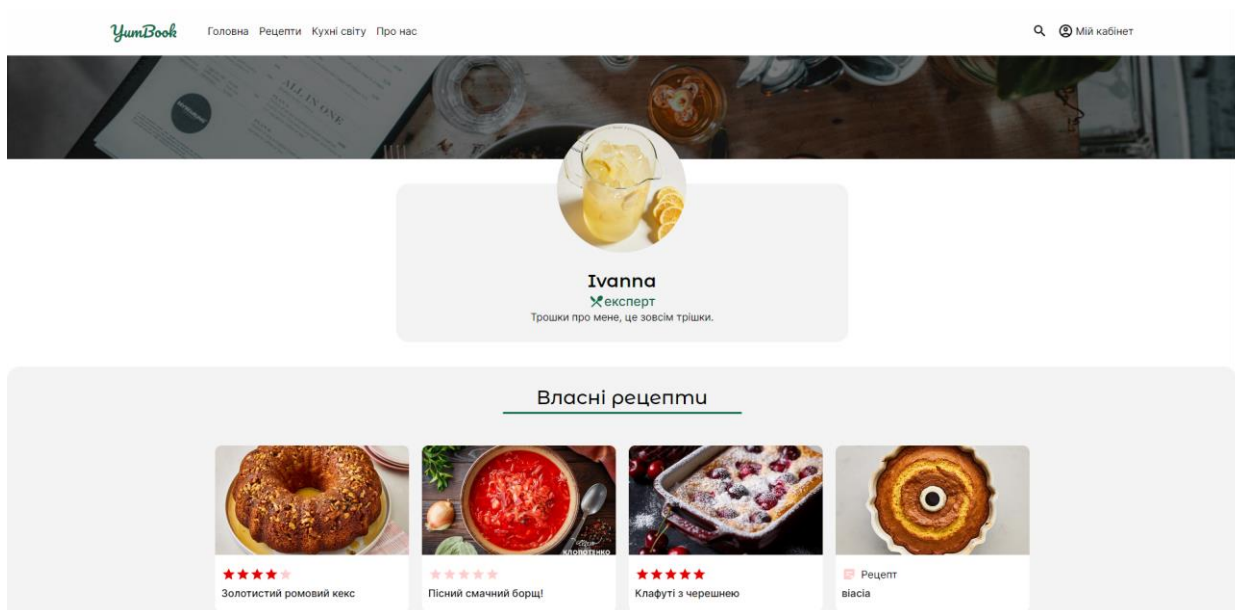


Рис. 3.36. Профіль іншого користувача

Тепер перейдемо до функціоналу адміністратора, для цього адміністратор, аналогічно до користувача повинен увійти у обліковий запис. Після входу, у кабінеті користувача, сторінці «Про мене» відобразиться посилання для переходу в панель адміністратора (рис. 3.37).

Персональна інформація

Вся інформація про вас.

Ім'я

Ivanna

Рівень

Експерт

Логін

first1

E-mail

first1@gmail.com

Адмін панель

Перейти на панель адміністратора

Перейти на публічний профіль

Рис. 3.37. Відображення посилання на панель адміністратора

Натискаємо на посилання, відбувається перехід на головну сторінку адміністративної панелі (рис. 3.38).

Панель адміністратора

Привіт, Ivanna!

Головна

Користувачі

Кухні світу

Рецепти

Відгуки

Yumbook

Вихід

Панель адміністратора

Користувачі

Кухні світу

Рецепти

Відгуки

Рис. 3.38. Головна сторінка панелі адміністратора

Переходимо до управління користувачами, натиснувши в меню «Користувачі». Відображається таблиця з усіма користувачами та діями, що можна зробити відносно них (рис. 3.39).

Панель адміністратора

Привіт, Ivanna!

Головна

Користувачі

Кухні світу

Рецепти

Відгуки

Yumbook

Вихід

Керування користувачами

id	Логін	E-mail	Ім'я	Роль	Рівень	Про користувача	Дії
1	first1	first1@gmail.com	Ivanna	Адміністратор	Expert	Трошки про мене, це зовсім трішки.	
2	second1	second1@gmail.com	Usef	Користувач	Amateur	ddddddd	
3	third1	third1@gmail.com	Hanna	Користувач	Amateur	Пусто	
4	jocli	joacliff@gmail.com	Joanna	Користувач	Home Cook	Привіт :)	
7	hanco	hanco@gmail.com	Hank	Користувач	Amateur	Пусто	
8	leco	leocorn@gmail.com	Leo	Користувач	Amateur	Пусто	
9	limay	linamay@gmail.com	Lina	Користувач	Amateur	Пусто	
10	paulg	paulg@gmail.com	Paul	Користувач	Amateur	Пусто	
11	fourth1	fourth1@gmail.com	Forton	Користувач	Amateur	Пусто	
12	lavander	lavander@mail.com	Amber	Користувач	Amateur	Пусто	

Наступна

Рис. 3.39. Сторінка керування користувачами

Якщо натиснути на значок видалення користувача, з'явиться вікно для підтвердження дії (рис. 3.40). Аналогічно станеться в керуванні кухнями світу, рецептами та відгуками.

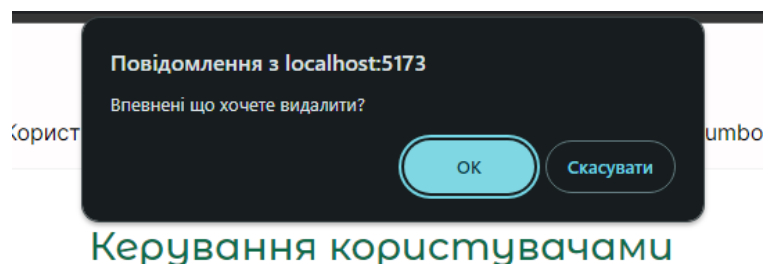


Рис. 3.40. Повідомлення для підтвердження видалення користувача

Скасуємо видалення та натиснемо на значок редагування користувача, відбувається перехід на відповідну сторінку. Змінимо роль користувача на адміністратора (рис. 3.41).

Рис. 3.41. Сторінка редагування користувача

При натисканні на кнопку «Оновити», з'являється повідомлення про успішне оновлення користувача (рис. 3.42). Аналогічно відбудеться і при оновленні даних про кухню світу та відгуки.

Рис. 3.42. Результат оновлення користувача

Зміни в таблиці користувачів зображені на рисунку 3.43.

4	jocli	joacliff@gmail.com	Joanna	Адміністратор
---	-------	--------------------	--------	---------------

Рис. 3.43. Зміни в таблиці користувачів

Перейдемо на сторінку управління кухнями світу через головне меню (рис. 3.44).

id	Назва	Опис	Зображення	Дії
1	Загальна	Загальна кухня, містить рецепти, які не належать до певної кухні світу!	1717947120928Golden-Rum-Cake (2).webp	
2	Американська	Текст про американську кухню.	1718113877178Golden-Rum-Cake.webp	
3	Французька	Французька кухня говорить чи каже "Bonjour!"	1717980666942fresh-broccoli-salad.webp	
4	Австралійська	Австралійська кухня цікава)	1718113893573apple-cake.webp	
5	Гавайська	Інопланетяни теж мають потребу у смачній їжі!	1718113977652best-lemonade-ever-step.webp	

Рис. 3.44. Сторінка керування кухнями світу

Створимо кухню світу, для цього натискаємо на «Додати», відображається створення кухні світу. Вводимо дані, додаємо фотографію та натискаємо на кнопку. Відображається повідомлення про успішне створення (рис. 3.45).

Назва

Грецька

Опис

Опис грецької кухні.

Додати фото

Додати

Кухня успішно створена

Рис. 3.45. Успішне створення кухні світу

Новостворена кухня також відображається у вебдодатку (рис. 3.46).

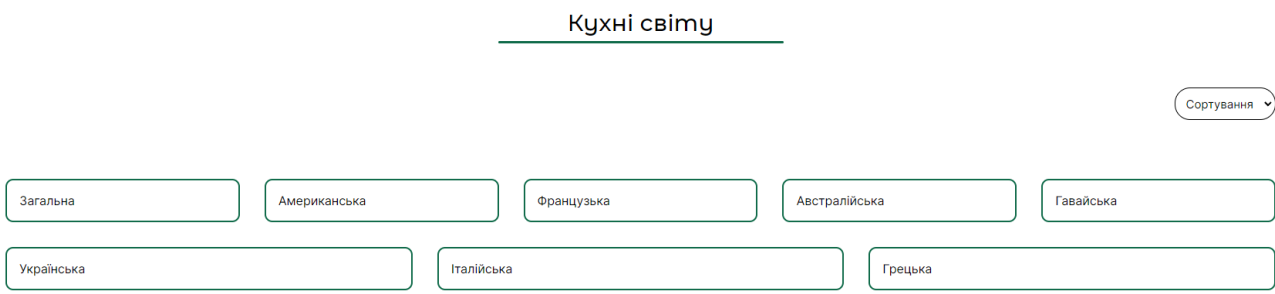


Рис. 3.46. Грецька кухня на сторінці кухонь світу

Перейдемо до сторінки керування рецептами, через меню в панелі адміністратора. Тут відображені всі рецепти незалежно від їх статусу (рис. 3.47).

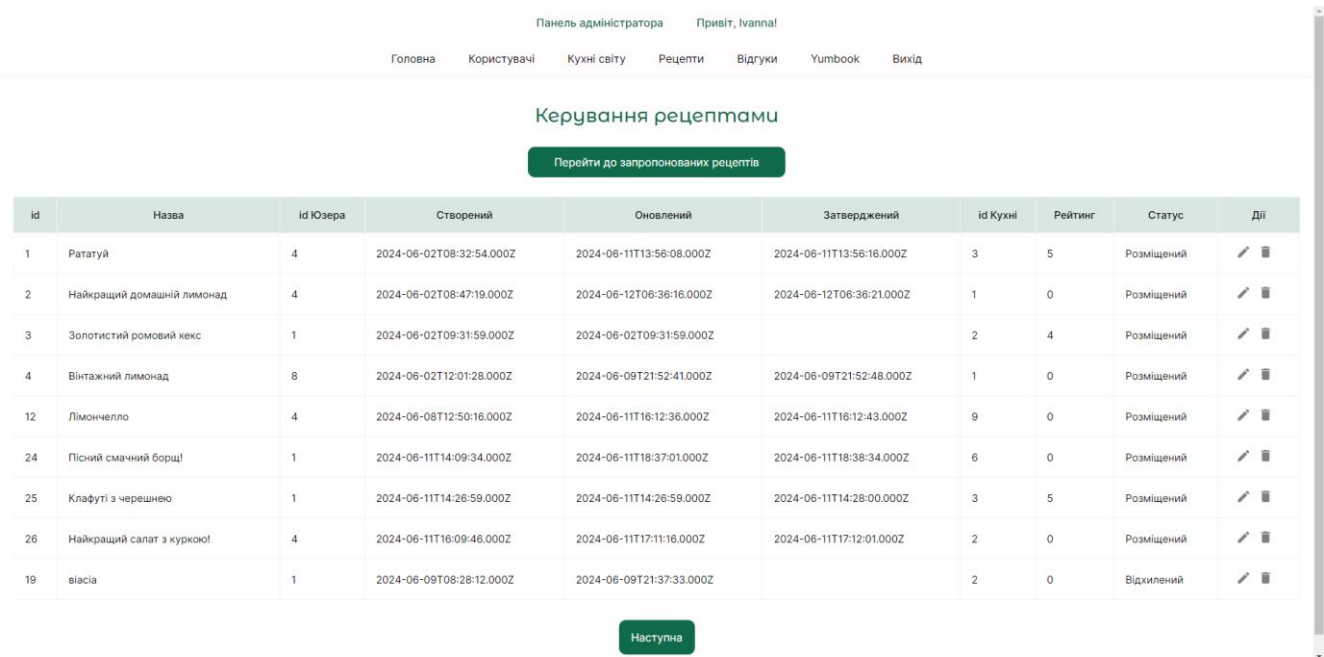


Рис. 3.47. Сторінка керування рецептами

Також можливо перейти на сторінку, де знаходяться лише запропоновані рецепти, натиснувши на відповідну кнопку. Тут відображається рецепт, раніше запропонований користувачем (рис. 3.48).

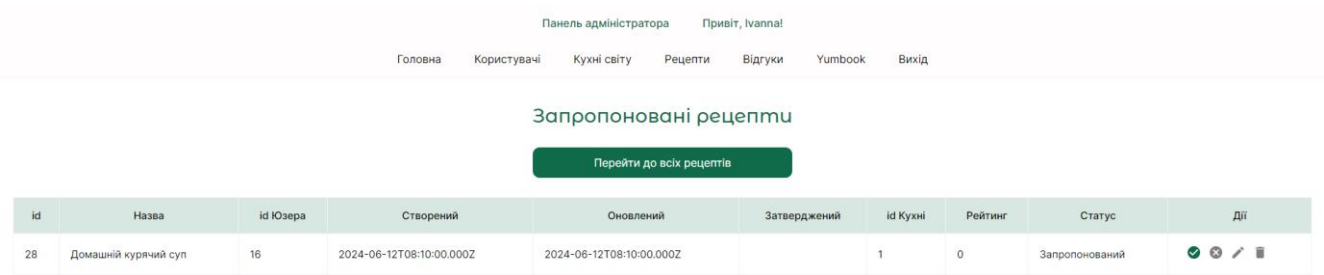


Рис. 3.48. Сторінка запропонованих рецептів

Даний рецепт можливо затвердити, натиснувши на значок, що на рисунку 3.48. має зелений колір. При натисканні з'явиться діалогове вікно для затвердження дії (рис. 3.49).

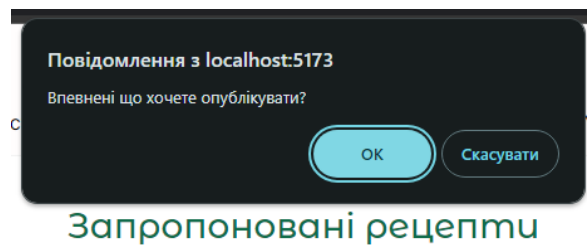


Рис. 3.49. Повідомлення для підтвердження публікації рецепту
Натиснемо «ОК», рецепт зникне зі списку затверджених (рис. 3.50).

Запропоновані рецепти

Перейти до всіх рецептів

id	Назва	id Юзера	Створений	Оновлений	Затверджений	id Кухні	Рейтинг	Статус	Дії
----	-------	----------	-----------	-----------	--------------	----------	---------	--------	-----

Рис. 3.50. Пустий список запропонованих рецептів

Рецепт можна також відхилити, все спрацює аналогічно до затвердження.

Перейдемо на сторінку керування відгуками, де відображаються всі запропоновані, затверджені відгуки та вся інформація про них (рис. 3.51).

Керування відгуками

id	id Юзера	id Рецепту	Оцінка	Текст	Створений	Статус	Зображення	Дії
1	1	3	4 ★	Чудовий рецепт. Вся родина була задоволена!	2024-06-02T08:32:54.000Z	Approved		
3	8	3	5 ★	Такий смачний, просто неперевершений!	2024-06-07T16:08:58.000Z	Approved		
4	4	3	3 ★	Не дуже сподобався, бо надто солодкий, але якщо додавати менше цукру, має бути смачно.	2024-06-07T16:48:55.000Z	Approved		
9	1	25	5 ★	Такий смачний!!	2024-06-11T14:28:26.000Z	Approved		
11	1	1	5 ★	Мені дуже сподобався рецепт, поки сезон, готуватиму!	2024-06-11T14:40:08.000Z	Approved		

Наступна

Рис. 3.51. Сторінка керування відгуками

Натиснемо кнопку «Наступна» та побачимо продовження таблиці, а також запропоновані відгуки (рис. 3.52). У даній таблиці затвердження працює ідентично до затвердження рецептів, тому його розглядати не будемо.

Керування відгуками									
id	id Юзера	id Рецепту	Оцінка	Текст	Створений	Статус	Зображення	Дії	
10	1	2	5 ★	Освіжас!	2024-06-11T14:39:02.000Z	Submitted			
12	1	24	5 ★	смачний	2024-06-11T19:05:12.000Z	Submitted			
13	16	25	4 ★	Достатньо смачно, але трохи перебір з цукром. Бодян не додавала, не було.	2024-06-12T09:12:01.000Z	Submitted			

Попередня

Рис. 3.52. Сторінка керування відгуками

Всі основні дії були розглянуті, але залишився важливий аспект — адаптивність інтерфейсу. Тож вийдемо з облікового запису та переглянемо вигляд головної сторінки зі смартфона (рис. 3.53).

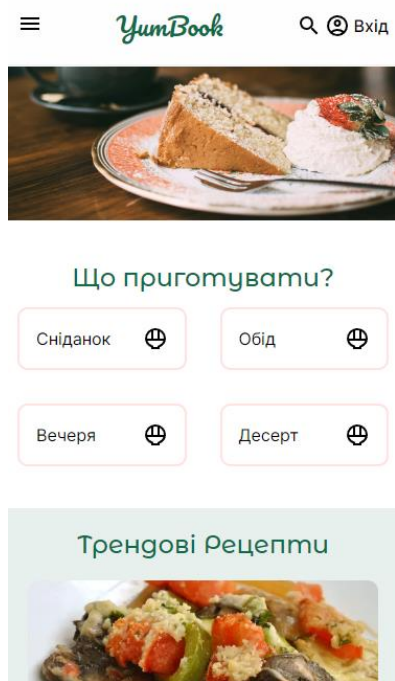


Рис. 3.53. Головна сторінка на смартфоні

При натисканні на значок меню, зліва на навігаційній панелі, з’являється меню, для зручної навігації по вебдодатку (рис. 3.54).

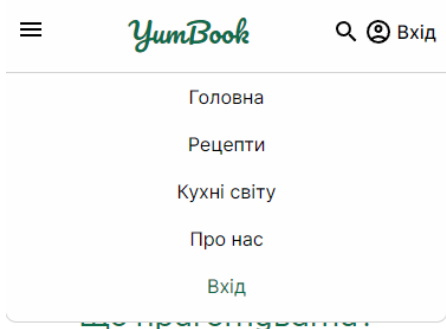


Рис. 3.54. Вигляд меню на смартфоні

3.3 Тестування роботи вебдодатку.

Тестування є ключовим етапом у забезпеченні якості та надійності додатку, допомагає виявляти і усувати помилки, недоліки та потенційні проблеми. Воно також дозволяє переконатися, що додаток відповідає вимогам і очікуванням користувачів.

Тестування необхідно проводити протягом всього процесу розробки, а також після завершення розробки для перевірки всього функціоналу. Після виявлення та виправлення помилок додаток тестується повторно. Існує багато технік тестування, які ефективно застосовуються у великих додатках. Для даного ж додатку використовувалися найпростіші техніки, оскільки він невеликий та не має складних проблем, які неможливо легко виявити. Тестування допомагає забезпечити високу якість програмного забезпечення і задовольнити вимоги користувачів. Воно спрямоване на перевірку правильності роботи окремих функцій, модулів або всієї системи [28].

Велику кількість помилок зручно виявляти за допомогою такої техніки, як "передбачення помилок", а для форм з числовими значеннями – техніки "аналізу граничних значень". У даному контексті використання передбачення помилок буде відмінним вибором, оскільки весь додаток був розроблений однією людиною, яка має чітке уявлення про потенційні недоліки, що можуть виникнути під час розробки [29].

Аналіз граничних значень базується на припущенні, що помилки найчастіше виникають на межах діапазону вхідних даних. Таким чином, якщо програма правильно працює для найменших і найбільших значень, вона, ймовірно, буде працювати правильно для інших значень в межах діапазону [30].

Протягом розробки вебдодатку було створено тест-кейси для перевірки коректної роботи основного функціоналу.

1. Авторизація у веб-додатку.

передумови: користувач хоче увійти у свій обліковий запис, коли він вже був зареєстрований.

кроки:

- 1) Користувач переходить на сторінку входу.
- 2) Користувач вводить логін та пароль.
- 3) Натискає кнопку «Вхід».

очікуваний результат: користувач успішно авторизується та зберігається у cookie зі своїм унікальним токеном.

фактичний результат: користувач успішно авторизується та зберігається у cookie зі своїм унікальним токеном.

2. Збереження рецепту

передумови: авторизований користувач хоче зберегти рецепт.

кроки:

- 1) Користувач переходить на рецепт.
- 2) Натискає на «Зберегти».
- 3) Переходить на збережені рецепти.

очікуваний результат: користувач бачить у збережених рецептах щойно доданий рецепт.

фактичний результат: користувач бачить у збережених рецептах щойно доданий рецепт.

3. Додавання відгуку на рецепт

передумови: авторизований користувач хоче додати відгук під рецептом

кроки:

- 1) Користувач переходить на певний рецепт.
- 2) Натискає «Оцінити»
- 3) Вводить у форму для відгуку текст
- 4) Встановлює оцінку
- 5) Натискає на кнопку «Надіслати»

очікуваний результат: користувач успішно додає відгук та бачить його у всіх відгуках, зазначеним як «Запропоновано вами».

фактичний результат: користувач успішно додає відгук та бачить його у всіх відгуках, зазначеним як «Запропоновано вами».

4. Редагування опису про користувача

передумови: авторизований користувач хоче відредагувати опис про себе.

кроки:

- 1) Користувач переходить у кабінет.
- 2) Переходить до налаштувань.
- 3) Змінює поле «Про себе».
- 4) Натискає кнопку «Оновити».
- 5) Переходить на свій публічний профіль.

очікуваний результат: користувач успішно змінив дані без помилок, зміни відображено у його публічному профілі.

фактичний результат: користувач успішно змінив дані без помилок, зміни відображено у його публічному профілі.

5. Додавання рецепту

передумови: авторизований користувач хоче додати та запропонувати власний рецепт.

кроки:

- 1) Користувач переходить у кабінет.
- 2) Переходить до «Мої рецепти».
- 3) Натискає «Додати рецепт».
- 4) Вводить всі необхідні дані та не допускається помилок.
- 5) Обирає «Запропонувати рецепт».
- 6) Натискає на кнопку «Додати рецепт».

очікуваний результат: користувач успішно додав рецепт, його видно у «Мої рецепти» зі статусом «Запропонований»

фактичний результат: користувач успішно додав рецепт, його видно у «Мої рецепти» зі статусом «Запропонований»

6. Фільтрація та сортування рецептів.

передумови: користувач хоче відфільтрувати рецепт за певними параметрами, а також відсортувати за останніми доданими.

кроки:

- 1) Користувач переходить на сторінку рецептів.

2) У фільтрації обирає необхідні параметри.

3) У сортуванні обирає «Останні».

очікуваний результат: користувач бачить рецепти, що відображені відповідно до сортування та фільтрації

фактичний результат: користувач бачить рецепти, що відображені відповідно до сортування та фільтрації.

7. Пошук рецептів за інгредієнтом

передумови: авторизований користувач хоче знайти рецепти за інгредієнтом.

кроки:

1) Користувач переходить на сторінку пошуку

2) Обирає метод пошуку – «Інгредієнти»

3) Вводить в поле назву інгредієнту.

очікуваний результат: користувач бачить рецепти, що відображені відповідно до вказаного інгредієнту.

фактичний результат: користувач бачить рецепти, що відображені відповідно до вказаного інгредієнту.

Тож підсумовуючи можна сказати що під час тестування доступного функціоналу додатку у нього не було виявлено жодних недоліків чи помилок у роботі.

Висновки до третього розділу

У третьому розділі було надано послідовність кроків щодо встановлення та налаштування веб-додатку, а також діаграму розгортання. Крім того, було розглянуто роботу як з основною частиною вебдодатку, так і з панеллю адміністратора. Було детально проаналізовано компоненти інтерфейсу програмного забезпечення, а також їх призначення та можливості.

Додаток пройшов процес тестування з метою виявлення помилок в роботі, з метою підвищення якості вебдодатку. Створено основні тест-кейси для полегшення пошуку недоліків. У результаті тестування не було виявлено помилок, що свідчить про справну роботу додатку.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						70

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра, було розроблено готовий до користування вебдодаток кулінарних рецептів.

Спершу, була поставлена задача на створення вебдодатку та проведено аналіз предметної області, а також виконано детальний порівняльний аналіз кулінарних вебдодатків. Важливою частиною аналізу стало встановлення основних вимог до майбутнього вебдодатку, який має надавати користувачам інтуїтивний та зручний інтерфейс для пошуку, збереження та додавання рецептів. З урахуванням отриманих вимог та критеріїв, був складений перелік основних технологій та інструментів, що будуть використовуватися під час розробки вебдодатку кулінарних рецептів. Для серверної частини використовувався Node.JS та Express.JS. Для клієнтського додатку – React, а для бази даних – MySQL.

Далі були визначені варіанти використання вебдодатку, функціональні та нефункціональні вимоги до додатку. Було створено діаграму класів, що відображає основну логіку роботи програмного продукту та зв'язки між компонентами системи. На основі діаграми сутностей та загальних вимог, було спроектовано таблиці для бази даних вебдодатку кулінарних рецептів та створено схему бази даних.

Було сформульовано основний алгоритм роботи системи, створено діаграми активності як для користувача, так і для адміністратора. Вебдодаток кулінарних рецептів був повністю спроектований та реалізований. Також була надана реалізація основних кроків початку створення вебдодатку та окремого функціоналу для відгуків до рецептів.

Була створена система, що відповідає визначеним вимогам до додатку. Дана система включає функціонал для спрощення пошуку, збереження та додавання рецептів та відгуків до них, що впливають на рейтингову систему та відображення рецептів у вебдодатку.

Крім того, було сформовано порядок встановлення та налаштування додатку. Вебдодаток пройшов процес тестування, під час якого були виявлені та виправлені недоліки.

Адаптивний інтерфейс вебдодатку робить його доступним при різних розширеннях екрану. Інтуїтивний інтерфейс дозволяє легко знаходити рецепти за власними потребами, зберігати їх для подальшого приготування, ділитись досвідом приготування рецептів, додавати власні рецепти та обмінюватись ними.

Вебдодаток може бути налаштований та використаний для індивідуальних кухарів, кулінарних спільнот та шкіл для збереження, організації, додавання та обміну рецептами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SURVEY REVEALS 81% OF CONSUMERS NOW COOK MORE THAN HALF OF THEIR MEALS AT HOME [Електронний ресурс] – Режим доступу до ресурсу: <https://www.prnewswire.com/news-releases/survey-reveals-81-of-consumers-now-cook-more-than-half-of-their-meals-at-home-302007657.html>
2. КЛОПОТЕНКО [Електронний ресурс] – Режим доступу до ресурсу: <https://klopotenko.com/>
3. Allrecipes [Електронний ресурс] – Режим доступу до ресурсу: <https://www.allrecipes.com/>
4. Epicurious [Електронний ресурс] – Режим доступу до ресурсу: <https://www.epicurious.com/>
5. Cookpad [Електронний ресурс] – Режим доступу до ресурсу: <https://cookpad.com/ua>
6. What is Client-Server Architecture? Everything You Should Know [Електронний ресурс] – Режим доступу до ресурсу: <https://www.simplilearn.com/what-is-client-server-architecture-article>
7. Node.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/en>
8. Express [Електронний ресурс] – Режим доступу до ресурсу: <https://expressjs.com/uk/>
9. React [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/>
10. MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mysql.com/>
11. Next.js by Vercel - The React Framework [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/>
12. Next JS Vs Express: An In-depth Comparison [Електронний ресурс] – Режим доступу до ресурсу: <https://www.openxcell.com/blog/next-js-vs-express/>
13. UML Use Case Diagram Tutorial [Електронний ресурс] – Режим доступу до ресурсу: <https://www.lucidchart.com/pages/uml-use-case-diagram>

14. Unified Modeling Language (UML) Diagrams [Электронный ресурс] – Режим доступа до ресурсу: <https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/>
15. The Ultimate Class Diagram Tutorial to Help Model Your Systems Easily [Электронный ресурс] – Режим доступа до ресурсу: <https://creately.com/blog/software-teams/class-diagram-tutorial/>
16. MySQL Workbench [Электронный ресурс] – Режим доступа до ресурсу: <https://www.mysql.com/products/workbench/>
17. SPA (Single-page application) [Электронный ресурс] – Режим доступа до ресурсу: <https://developer.mozilla.org/en-US/docs/Glossary/SPA>
18. What is Activity Diagram? [Электронный ресурс] – Режим доступа до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/>
19. Visual Studio Code - Code Editing. Redefined [Электронный ресурс] – Режим доступа до ресурсу: <https://code.visualstudio.com/>
20. CORS [Электронный ресурс] – Режим доступа до ресурсу: <https://expressjs.com/en/resources/middleware/cors.html>
21. How to Hash Passwords with bcrypt in Node.js [Электронный ресурс] – Режим доступа до ресурсу: <https://www.freecodecamp.org/news/how-to-hash-passwords-with-bcrypt-in-nodejs/>
22. Postman documentation overview [Электронный ресурс] – Режим доступа до ресурсу: <https://learning.postman.com/docs/introduction/overview/>
23. Vite [Электронный ресурс] – Режим доступа: <https://vitejs.dev/>
24. Reactrouter [Электронный ресурс] – Режим доступа до ресурсу: <https://reactrouter.com/en/main/routers/create-browser-router>
25. Zustand [Электронный ресурс] – Режим доступа до ресурсу: <https://docs.pmnd.rs/zustand/getting-started/introduction>
26. Multer [Электронный ресурс] – Режим доступа до ресурсу: <https://www.npmjs.com/package/multer>

27. Create a UML deployment diagram [Электронный ресурс] – Режим доступа до ресурсу: <https://support.microsoft.com/en-au/office/create-a-uml-deployment-diagram-60811688-d811-4387-9715-c1f02f30b125>
28. 5 Test Design Techniques QA Engineers Should Know - QA Madness [Электронный ресурс] – Режим доступа до ресурсу: <https://www.qamadness.com/5-test-design-techniques-qa-engineers-should-know/>
29. What is Error Guessing Technique in Software Testing? [Электронный ресурс] – Режим доступа до ресурсу: <https://testsigma.com/blog/error-guessing/>
30. What is Boundary Value Analysis? [Электронный ресурс] – Режим доступа до ресурсу: <https://www.getsoftwareservice.com/what-is-boundary-value-analysis/>

ДОДАТКИ

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						76

Лістинг index.js:

```

app.use(cookieParser());

const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, "../client/public/upload");
  },
  filename: function (req, file, cb) {
    cb(null, Date.now() + file.originalname);
  },
});

const upload = multer({ storage: storage });

app.post("/api/upload", upload.single("file"), (req, res) => {
  const file = req.file;
  res.status(200).json(file.filename);
});

app.use("/api/users", usersRoutes);

app.use("/api/auth", authRoutes);

app.use("/api/cuisines", cuisinesRoutes);
app.use("/api/recipes", recipesRoutes);
app.use("/api/ingredients", ingredientsRoutes);
app.use("/api/directions", directionsRoutes);
app.use("/api/saves", savesRoutes);
app.use("/api/search", searchRoutes);
app.use("/api/categories", categoriesRoutes);
app.use("/api/reviews", reviewsRoutes);
app.use("/api/admin", adminRoutes);

app.listen(9900, () => {
  console.log("api work");
});

```

Лістинг auth.js:

```

import { db } from "../connect.js";
import bcrypt from "bcryptjs";
import jwt from "jsonwebtoken";

export const register = (req, res) => {
  //перевірка чи користувач з таким логіном існує
  const q = `SELECT * FROM users WHERE username = ?`;

```

```

db.query(q, [req.body.username], (err, data) => {
  if (err) return res.status(500).json(err);
  if (data.length) return res.status(409).json("User already exists!");

  //створення нового користувача
  //хешування паролю
  const salt = bcrypt.genSaltSync(10);
  const hashedPassword = bcrypt.hashSync(req.body.password, salt);

  const q =
    "INSERT INTO users (username, email, password, name, level, profilePic)
  VALUE (?)";

  const img = req.body.img ? req.body.img : "NULL";

  const values = [
    req.body.username, req.body.email, hashedPassword, req.body.name,
    req.body.level, img,
  ];

  db.query(q, [values], (err, data) => {
    if (err) return res.status(500).json(err);
    return res.status(200).json("User successfully created");
  });
});

export const login = (req, res) => {
  const q = "SELECT * FROM users WHERE username = ?";
  db.query(q, [req.body.username], (err, data) => {
    if (err) return res.status(500).json(err);
    if (data.length === 0) return res.status(404).json("User not found");

    const checkPassword = bcrypt.compareSync(
      req.body.password,
      data[0].password
    );

    if (!checkPassword)
      return res.status(400).json("Wrong password or username");

    const token = jwt.sign(
      { id: data[0].id, isAdmin: data[0].isAdmin },
      "secretkey"
    );

    const { password, ...others } = data[0];

    const expiresIn = new Date(Date.now() + 604800000); // 1 тиждень у
    мілісекундах
  });
};

```

```

    res
      .cookie("accessToken", token, {
        httpOnly: true,
        expires: expiresIn,
      })
      .status(200)
      .json(others);
  });
};

export const logout = (req, res) => {
  res
    .clearCookie("accessToken", {
      secure: true,
      sameSite: "none",
    })
    .status(200)
    .json("User has been logged out");
};

```

Лістинг auth.js:

```

import express from "express";
import { register, login, logout } from "../controllers/auth.js";

const router = express.Router();

router.post("/register", register);
router.post("/login", login);
router.get("/logout", logout);

export default router;

```

Лістинг App.jsx:

```

const Layout = () => {
  return (
    <div >
      <Navbar />
      <Outlet className="outlet"/>
      <Footer />
    </div>
  );
};

const ProtectedRoute = ({ children }) => {
  const { currentUser } = useAuthStore();

  if (!currentUser) {
    return <Navigate to="/" />;
  }
}

```

```

    return children;
  };

  const ProtectedRouteAdmin = ({ children }) => {
    const { currentUser, isAdmin } = useAuthStore();

    if (!currentUser) {
      return <Navigate to="/" />;
    }

    if (!isAdmin) {
      return <Navigate to="/" />;
    }

    return children;
  };

  const router = createBrowserRouter([
    // Основний маршрут для головної сторінки
    {
      path: "/",
      element: <Layout />,
      errorElement: <NotFoundPage />,
      children: [

        {
          path: "/",
          element: <Home />,
        },
        {
          path: "/recipes",
          element: <Recipes />,
        },
        {
          path: "/recipes/:recipeId",
          element: <Recipe />,
        },
        {
          path: "/cuisines",
          element: <Cuisines />,
        },
        {
          path: "/cuisines/:cuisineId",
          element: <Cuisine />,
        },
        {
          path: "/about",
          element: <About />,
        },
        {
          path: "/signin",

```

					ПЗ.КР.Б – 121 – 24 – ПЗ

```

        <AboutMe />
      </div>
    ),
  },
  {
    path: "/profile/my_recipes",
    element: (
      <div className="profile_layout_container">
        <ProfileMenu />
        <MyRecipes />
      </div>
    ),
  },
  {
    path: "/profile/saved_recipes",
    element: (
      <div className="profile_layout_container">
        <ProfileMenu />
        <SavedRecipes />
      </div>
    ),
  },
  {
    path: "/profile/recipes_collections",
    element: (
      <div className="profile_layout_container">
        <ProfileMenu />
        <RecipesCollections />
      </div>
    ),
  },
  {
    path: "/profile/settings",
    element: (
      <div className="profile_layout_container">
        <ProfileMenu />
        <Settings />
      </div>
    ),
  },
],
},
],
{
  path: "/not_authorized",
  element: <NotFound text={"Тут нічого"} />,
},
{
  path: "/admin",
  errorElement: <NotFoundPage />,
}

```

```

    element: (
      <ProtectedRouteAdmin>
        <Outlet />
      </ProtectedRouteAdmin>
    ),
    children: [
      {
        path: "/admin/recipes/edit/:recipeId",
        element: (
          <div className="admin_layout_container">
            <AdminMenu />
            <EditRecipe type={"admin"} />
          </div>
        ),
      },
      {
        path: "/admin/",
        element: (
          <div className="admin_layout_container">
            <AdminMenu />
            <AdminMain />
          </div>
        ),
      },
      {
        path: "/admin/users",
        element: (
          <div className="admin_layout_container">
            <AdminMenu />
            <AdminUsers />
          </div>
        ),
      },
      {
        path: "/admin/users/edit/:userId",
        element: (
          <div className="admin_layout_container">
            <AdminMenu />
            <AdminUsersEdit />
          </div>
        ),
      },
      {
        path: "/admin/cuisines",
        element: (
          <div className="admin_layout_container">
            <AdminMenu />
            <AdminCuisines />
          </div>
        ),
      },
    ],
  },

```

```

{
  path: "/admin/cuisines/edit/:cuisineId",
  element: (
    <div className="admin_layout_container">
      <AdminMenu />
      <AdminCuisinesEdit />
    </div>
  ),
},
{
  path: "/admin/cuisines/add",
  element: (
    <div className="admin_layout_container">
      <AdminMenu />
      <AdminCuisinesAdd />
    </div>
  ),
},
{
  path: "/admin/recipes",
  element: (
    <div className="admin_layout_container">
      <AdminMenu />
      <AdminRecipes />
    </div>
  ),
},
{
  path: "/admin/recipes/submitted",
  element: (
    <div className="admin_layout_container">
      <AdminMenu />
      <AdminSubmittedRecipes />
    </div>
  ),
},
{
  path: "/admin/reviews",
  element: (
    <div className="admin_layout_container">
      <AdminMenu />
      <AdminReviews />
    </div>
  ),
},
],
},
]);

function App() {

```

```

    return (
      <>
        <div className="App">
          <RouterProvider router={router} />
        </div>
      </>
    );
  }

  export default App;

```

Лістинг authStore.jsx:

```

import { create } from "zustand";
import axios from "axios";
import Cookies from "js-cookie";

const useAuthStore = create((set) => ({
  currentUser: JSON.parse(localStorage.getItem("user")) || null,
  isAdmin: Cookies.get("isAdmin") === "true",
  setCurrentUser: (user) => {
    if (user) {
      const { isAdmin, ...userWithoutAdmin } = user;
      set({ currentUser: userWithoutAdmin, isAdmin: isAdmin === 1 });
      localStorage.setItem("user", JSON.stringify(userWithoutAdmin));
      Cookies.set("isAdmin", isAdmin === 1 ? "true" : "false");
    } else {
      set({ currentUser: null, isAdmin: false });
      localStorage.removeItem("user");
      Cookies.remove("isAdmin");
    }
  },
  login: async (inputs) => {
    try {
      localStorage.clear();
      const res = await axios.post(
        "http://localhost:9900/api/auth/login",
        inputs,
        {
          withCredentials: true,
        }
      );
      const { isAdmin, ...userWithoutAdmin } = res.data;
      set({ currentUser: userWithoutAdmin, isAdmin: isAdmin === 1 });
      localStorage.setItem("user", JSON.stringify(userWithoutAdmin));
      Cookies.set("isAdmin", isAdmin === 1 ? "true" : "false");
    } catch (error) {
      console.error("Failed to login:", error);
      throw error;
    }
  },
});

```

```

logout: async () => {
  try {
    localStorage.clear();
    await axios.get("http://localhost:9900/api/auth/logout", {
      withCredentials: true,
    });
    set({ currentUser: null, isAdmin: false });
    Cookies.remove("isAdmin");
  } catch (error) {
    console.error("Failed to logout:", error);
  }
},
}}));

export default useAuthStore;

```

Лістинг review.js:

```

import { db } from "../connect.js";
import moment from "moment";

export const addReview = (req, res) => {
  const q =
    "INSERT INTO reviews (userId, recipeId, rating, text, createdAt, status, img) VALUE (?)";
  const values = [
    req.user.id,
    req.body.recipeId,
    req.body.rating,
    req.body.text,
    moment(Date.now()).format("YYYY-MM-DD HH:mm:ss"),
    "Submitted",
    req.body.img,
  ];

  db.query(q, [values], (err, data) => {
    if (err) return res.status(500).json(err);
    return res.status(200).json("Ingredient has been created");
  });
};

export const getReviewsByRecipe = (req, res) => {
  const { status, recipeId } = req.query;

  let q = `
    SELECT r.*, u.profilePic, u.name
    FROM reviews r
    JOIN users u ON r.userId = u.id
    WHERE r.recipeId = ?
  `;

  let values = [+recipeId];

```

```

    if (status) {
      q += " AND r.status = ?";
      values.push(status);
    }
    db.query(q, values, (err, data) => {
      if (err) return res.status(500).json(err);
      return res.status(200).json(data);
    });
  });
};

export const getReviewsByUser = (req, res) => {
  const { status, userId } = req.query;

  let q = `
    SELECT r.*, u.profilePic, u.name
    FROM reviews r
    JOIN users u ON r.userId = u.id
    WHERE r.userId = ?
  `;
  let values = [+userId];

  if (status) {
    q += " AND r.status = ?";
    values.push(status);
  }

  db.query(q, values, (err, data) => {
    if (err) return res.status(500).json(err);
    return res.status(200).json(data);
  });
};

export const getReviewsByUserAndRecipe = (req, res) => {
  const { userId, recipeId } = req.query;
  let q = "";
  let values = [];
  if (userId) {
    q = `
      SELECT r.*, u.profilePic, u.name
      FROM reviews r
      JOIN users u ON r.userId = u.id
      WHERE r.userId = ? AND r.recipeId = ? AND r.status = 'Submitted'
      UNION
      SELECT r.*, u.profilePic, u.name
      FROM reviews r
      JOIN users u ON r.userId = u.id
      WHERE r.recipeId = ? AND r.status = 'Approved'
      ORDER BY CASE WHEN status = 'Submitted' THEN 0 ELSE 1 END, createdAt ASC
    `;
    values = [+userId, +recipeId, +recipeId];
  }
};

```

```

    } else {
      q = `
        SELECT r.*, u.profilePic, u.name
        FROM reviews r
        JOIN users u ON r.userId = u.id
        WHERE r.recipeId = ? AND r.status = 'Approved'
        ORDER BY createdAt ASC
      `;
      values = [+recipeId];
    }

    db.query(q, values, (err, data) => {
      if (err) return res.status(500).json(err);
      return res.status(200).json(data);
    });
  });
};

export const deleteOwnReview = (req, res) => {
  const { reviewId } = req.query;

  const q = "DELETE FROM reviews WHERE id = ? AND userId = ?";
  const values = [+reviewId, req.user.id];

  db.query(q, values, (err, data) => {
    if (err) return res.status(500).json(err);
    if (data.affectedRows === 0)
      return res
        .status(404)
        .json(
          "Review not found or you do not have permission to delete this review."
        );
    return res.status(200).json("Review has been deleted");
  });
};
};

```

```

export const deleteReview = async (req, res) => {
  const reviewId = +req.params.reviewId;

  const q = "DELETE FROM reviews WHERE id = ?";

  db.query(q, [reviewId], (err, result) => {
    if (err) {
      return res.status(500).json({
        message: "Error deleting review",
        error: err,
      });
    }

    if (result.affectedRows > 0) {
      return res.status(200).json({ message: "Review deleted successfully" });
    } else {
      return res.status(404).json({ message: "Review not found" });
    }
  });
};

export const getReviewsApprovedCountByRecipe = (req, res) => {
  const { recipeId } = req.query;

  if (!recipeId) {
    return res.status(400).json({ error: "Recipe ID is required" });
  }

  const q = `
    SELECT COUNT(*) as count
    FROM reviews
    WHERE recipeId = ? AND status = 'Approved'
  `;

  db.query(q, [recipeId], (err, data) => {
    if (err) {
      return res.status(500).json(err);
    }
    return res.status(200).json(data[0]);
  });
};

export const getReviewsAdmin = (req, res) => {
  const { sort, page = 1, pageSize = 10 } = req.query;

  let countQuery = "SELECT COUNT(*) AS total FROM reviews";
  let dataQuery = "SELECT * FROM reviews";
  let sortQuery = "";
  let values = [];

```

```

// Додаємо умову сортування, якщо вона присутня
if (sort) {
  const [sortField, sortOrder] = sort.split(" ");
  if (
    sortField &&
    sortOrder &&
    (sortOrder.toLowerCase() === "asc" || sortOrder.toLowerCase() === "desc")
  ) {
    sortQuery = ` ORDER BY ${sortField} ${sortOrder.toUpperCase()}`;
  }
}

db.query(countQuery, (err, countResult) => {
  if (err) return res.status(500).json(err);

  const totalReviews = countResult[0].total;
  const totalPages = Math.ceil(totalReviews / pageSize);

  const limitValue = parseInt(pageSize);
  const offsetValue = (parseInt(page) - 1) * limitValue;

  dataQuery += `${sortQuery} LIMIT ? OFFSET ?`;
  values.push(limitValue, offsetValue);

  db.query(dataQuery, values, (err, dataResult) => {
    if (err) return res.status(500).json(err);

    return res.status(200).json({
      total: totalReviews,
      totalPages: totalPages,
      reviews: dataResult,
    });
  });
});

export const approveReview = async (req, res) => {
  const reviewId = +req.params.reviewId;

  const q = "UPDATE reviews SET status = ? WHERE id = ?";

  db.query(q, ["Approved", reviewId], (err, result) => {
    if (err) {
      return res.status(500).json({
        message: "Error approving review",
        error: err,
      });
    }
  })
}

```

```

if (result.affectedRows > 0) {
    return res.status(200).json({ message: "Review approved successfully" });
} else {
    return res.status(404).json({ message: "Review not found" });
}
});
};

```

Лістинг reviews.js:

```

import express from "express";
import { verifyAdmin, verifyUser } from "../utils/verifyToken.js";
import {
    addReview,
    approveReview,
    deleteOwnReview,
    deleteReview,
    getReviewsAdmin,
    getReviewsApprovedCountByRecipe,
    getReviewsByRecipe,
    getReviewsByUser,
    getReviewsByUserAndRecipe,
} from "../controllers/review.js";

const router = express.Router();

router.get("/byrecipe", getReviewsByRecipe);
router.get("/byuser", verifyUser, getReviewsByUser);
router.get("/byusernrecipe", getReviewsByUserAndRecipe);
router.get("/getcountapproved", getReviewsApprovedCountByRecipe);

router.get("/getadmin", getReviewsAdmin);
router.post("/add", verifyUser, addReview);

router.patch("/approve/:reviewId", verifyAdmin, approveReview);

router.delete("/userdelete", verifyUser, deleteOwnReview);
router.delete("/:reviewId", verifyAdmin, deleteReview);

export default router;

```

Лістинг AddReview.jsx:

```

export default function AddReview({ setRefetchR, recipeId }) {
    const starsArr = [1, 2, 3, 4, 5];

    const { currentUser } = useAuthStore();

    const [file, setFile] = useState(null);
    const [text, setText] = useState("");

```

```

const [rate, setRate] = useState(0);
const [rateWidth, setRateWidth] = useState("");

const navigate = useNavigate();
const params = useParams();

useEffect(() => {
  const width = (100 * +rate) / 5;
  setRateWidth(width.toString() + "%");
}, [rate]);

const upload = async () => {
  try {
    const formData = new FormData();
    formData.append("file", file);
    const res = await makeRequest.post("/upload", formData);
    return res.data;
  } catch (err) {
    console.log(err);
  }
};

const [err, setErr] = useState(""); // Додайте стейт для зберігання помилок

const handleClick = async (e) => {
  e.preventDefault();
  if (!currentUser) return navigate("/signin"); // Перевіряємо, чи є поточний користувач, якщо ні, перенаправляємо його на сторінку входу

  if (!text.trim()) return setErr("Будь ласка, введіть текст відгуку"); // Перевіряємо, чи не порожній текст

  if (rate === 0) return setErr("Будь ласка, вкажіть рейтинг"); // Перевіряємо, чи встановлений рейтинг

  let imgUrl = "";
  if (file) imgUrl = await upload(); // Якщо є файл, виконуємо завантаження

  try {
    await addReview(params.recipeId, rate, text, imgUrl || null); // Додаємо відгук
    setText("");
    setFile(null);
    setRateWidth("0%");
    setRefetchR((prevState) => !prevState);
  } catch (error) {
    console.error("Failed to add review:", error);
  }
};

const addReview = async (recipeId, rate, text, imgUrl) => {

```

```

    try {
      const response = await makeRequest.post(
        `/reviews/add?userId=${currentUser.id}`,
        {
          recipeId: recipeId,
          rating: rate,
          text: text,
          img: imgUrl, // Якщо imgUrl існує, передаємо його, інакше передаємо
null
        }
      );
      return response.data;
    } catch (error) {
      console.error("Failed to add review:", error);
      throw error;
    }
  };

  return (
    <div className="addreview_container">
      <div className="inside">
        <div className="rate">
          /* Дайте оцінку рецепту від 1 до 5 */
          <div className="stars">
            <div className="stars__nofill">
              {starsArr.map((star, index) => (
                <StarIcon
                  className="icon"
                  key={index}
                  style={{ color: "#FFCD CD", fontSize: "22px" }}
                  onClick={(e) => {
                    e.stopPropagation();
                    setRate(index + 1);
                  }}
                />
              ))}
            <div className="stars__filled" style={{ width: rateWidth }}>
              {starsArr.map((star, index) => (
                <StarIcon
                  className="icon"
                  key={index}
                  style={{ color: "#E70000", fontSize: "22px" }}
                  onClick={(e) => {
                    e.stopPropagation();
                    setRate(index + 1);
                  }}
                />
              ))}
            </div>
          </div>
        </div>
      </div>
    </div>
  );

```

```

    </div>
    <div className="top">
      <div className="left">
        <textarea
          cols="65"
          rows="4"
          minLength={3}
          maxLength={500}
          placeholder={`Чи готували ви цей рецепт? Поділіться власним
враженням.`}
          onChange={(e) => setText(e.target.value)}
          value={text}
        />
      </div>

      <div className="right">
        {file && (
          <img className="file" alt="" src={URL.createObjectURL(file)} />
        )}
      </div>
    </div>
    <hr />
    <div className="bottom">
      <div className="left">
        <input
          type="file"
          id="file"
          style={{ display: "none" }}
          onChange={(e) => setFile(e.target.files[0])}
        />
        <label htmlFor="file">
          <div className="item">
            <AddPhotoAlternateIcon className="icon" />
            <span>Додати фото</span>
          </div>
        </label>
      </div>
      <div className="right">
        <button onClick={handleClick}>Надіслати</button>
      </div>
    </div>
    {err && <div className="error">{err}</div>}
  </div>
</div>
);
}

```